Hierarchical Uncertainty Exploration via Feedforward Posterior Trees

Elias Nehme
Electrical and Computer Engineering
Technion–Israel Institute of Technology
seliasne@campus.technion.ac.il

Rotem Mulayoff
CISPA Helmholtz Center
for Information Security
rotem.mulayof@gmail.com

Tomer Michaeli
Electrical and Computer Engineering
Technion–Israel Institute of Technology
tomer.m@ee.technion.ac.il

Abstract

When solving ill-posed inverse problems, one often desires to explore the space of potential solutions rather than be presented with a single plausible reconstruction. Valuable insights into these feasible solutions and their associated probabilities are embedded in the posterior distribution. However, when confronted with data of high dimensionality (such as images), visualizing this distribution becomes a formidable challenge, necessitating the application of effective summarization techniques before user examination. In this work, we introduce a new approach for visualizing posteriors across multiple levels of granularity using *tree*-valued predictions. Our method predicts a tree-valued hierarchical summarization of the posterior distribution for any input measurement, in a single forward pass of a neural network. We showcase the efficacy of our approach across diverse datasets and image restoration challenges, highlighting its prowess in uncertainty quantification and visualization. Our findings reveal that our method performs comparably to a baseline that hierarchically clusters samples from a diffusion-based posterior sampler, yet achieves this with orders of magnitude greater speed. Code and examples are available at our webpage.

1 Introduction

Communicating prediction uncertainty is a crucial element in advancing the reliability of machine learning models. This is particularly important in the realm of imaging inverse problems, in which a given input can typically be associated with a multitude of plausible solutions. In such cases, it is advantageous to equip users with efficient tools for exploring and visualizing the set of admissible solutions. Such tools may be of high value especially in safety-critical domains, such as scientific and medical image analysis [1–4], where inaccuracies in predictions could impact human lives.

Information about the plausible solutions and their respective likelihoods is encapsulated within the posterior distribution. However, high-dimensional posteriors are challenging to visualize. One popular approach for communicating uncertainty is to generate samples from the posterior [5–14]. Yet, in complex domains with high uncertainty levels, users may need to sift through hundreds of posterior samples per input to be able to confidently validate or refute suspicions about the unobserved ground-truth image [15]. Several methods have been proposed for generating a concise set of samples that highlight posterior diversity [15, 16]. However, choosing this set to be small is often insufficient for summarizing the posterior, while choosing it to be large hinders the user's ability to properly

38th Conference on Neural Information Processing Systems (NeurIPS 2024).

 https://doi.org/10.52202/079017-3975

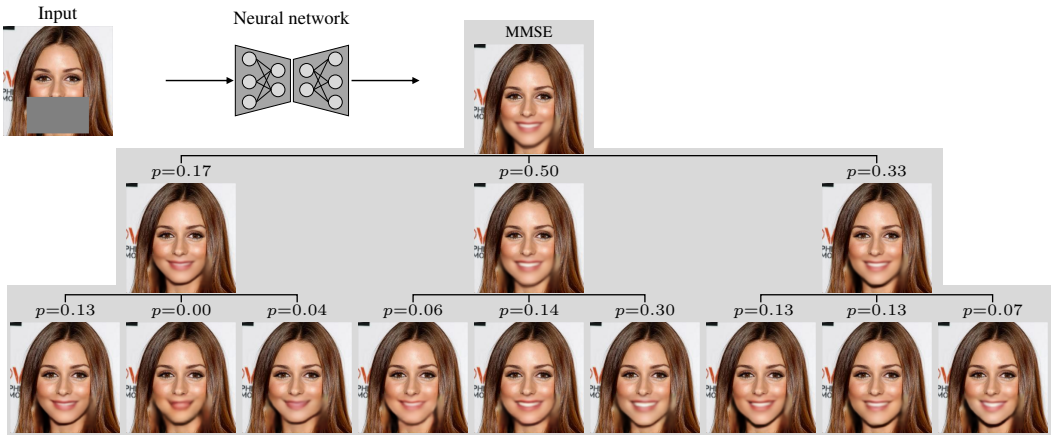


Figure 1: **Hierarchical decomposition of the minimum-MSE predictor into prototypes in the task of mouth inpainting**. The predicted tree explores the different options of bigger/smaller lips, mouth opening/closing, round/square jawline, etc.

inspect all solutions in the set. Other methods proposed to visualize uncertainty by allowing the user to navigate along the principal components (PCs) of the posterior [17–20]. While more interactive in nature, the projection onto a low dimensional principal subspace commonly accounts for only a small fraction of the reconstruction error.

In this work, we propose to model posterior uncertainty through *hierarchical* clustering, by using tree-structured outputs. In stark contrast to all existing methods, our approach supports efficient user interaction, allowing examination of only a small number of hypotheses en route to accepting or refuting a hypothesis about the unknown image. Specifically, our model receives a degraded measurement and outputs a hierarchy of predictions that cluster the solution set at multiple levels of granularity, each accompanied by their probability. This allows us to visualize the likelihood of different posterior modes, facilitating an informed visual exploration of posterior uncertainty (Fig. 1). Our method, which we coin *posterior trees*, is a generic technique for uncertainty visualization that is seamlessly transferable across tasks and datasets. Compared to the popular practice of training point estimators with mean square error (MSE) minimization, our method is an ideal drop-in replacement. This is because it adds a minor additional computational burden, while simultaneously accompanying the point estimate with its decomposition to the constituting clusters, revealing the underlying modes of variation and disclosing posterior uncertainty.

We showcase our method on multiple inverse problems in imaging, demonstrating the practical benefit of tree-valued predictions. We also quantitatively compare our learned posterior trees to a two-step baseline that generates samples using a diffusion-based posterior sampler and then applies hierarchical K-means clustering. Our approach achieves a comparable performance across tasks and datasets while enjoying a significantly faster runtime.

2 Related Work

The common avenue for probing data uncertainty (also known as aleatoric uncertainty [6–8, 21–23]) involves posterior sampling using conditional generative models like [5–14, 24–27], with the dominance of score-based/diffusion models [28–31] in recent years. While posterior sampling theoretically offers a sizable solution set per input, allowing useful uncertainty estimates through summarization methods like PCA [18] or K-means, its implementation proves impractically slow for contemporary state-of-the-art models. Despite attempts to enhance speed [32–35], the strategy remains plagued by extended run times.

In an effort to expedite inference, some studies proposed approximating posteriors with simpler distributions considering pixel correlations, such as correlated Gaussians [36–39]. More recently, two approaches emerged, directly outputting the top principal components (PCs) of the posterior distribution without imposing distributional assumptions [17, 19]. Additionally, subsequent work by

Yair et al. [20] visualized the *projected* posterior distribution onto the subspace spanned by the top PCs, unveiling its multi-modal nature at heightened uncertainty levels.

Multiple choice learning (MCL) [40–50] was proposed as a framework that can provide a multi-modal set of predictions when confronted with ambiguous inputs. The idea was originally proposed by Guzman-Rivera et al. [40] using structured support vector machines, and was later introduced to deep models in [42, 43]. The main working principle of MCL is to train an architecture with multiple output heads using the *oracle* loss, such that for a given input, only the head that is closest to the desired output gets updated. This winner-takes-all strategy encourages each head to specialize in a different mode of the output space, effectively predicting the centers of a Voronoi tessellation of all possible outputs [45]. Later works such as [48, 50] proposed to complement each prediction in the set with its likelihood. Nonetheless, existing multi-hypothesis methods are mainly tailored to classification settings, and are still limited in their ability to organize the predicted set of hypotheses effectively. Categorizing the solution set *hierarchically*, as we propose here, can significantly accelerate uncertainty exploration. For example, it allows iterative user interaction, focusing user efforts on the examination of a small number of hypotheses (not necessarily the most likely ones) to confirm/refute a suspicion about the signal. Moreover, the hierarchical structure can regularize the number of prototypes devoted to high-density modes, highlighting rare cases.

3 Method

Our goal is to predict a clean signal $\boldsymbol{x} \in \mathcal{X} \subseteq \mathbb{R}^{d_x}$, given some degraded measurements $\boldsymbol{y} \in \mathcal{Y} \subseteq \mathbb{R}^{d_y}$. We assume that $\boldsymbol{x}$ and $\boldsymbol{y}$ are realizations of random vectors $\mathbf{x}$ and $\mathbf{y}$ with an unknown joint distribution $p_{\mathbf{x},\mathbf{y}}(\boldsymbol{x}, \boldsymbol{y})$, from which we have a training set of *i.i.d.* samples $\mathcal{D} = \{(\boldsymbol{x}_i, \boldsymbol{y}_i)\}_{i=1}^N$. The objective of our predicted trees is therefore to visualize the uncertainty in the posterior distribution $p_{\mathbf{x}|\mathbf{y}}(\boldsymbol{x}|\boldsymbol{y})$ via a hierarchical set of a few prototypes.

3.1 Multiple Output Prediction

Many image restoration methods output a single prediction $\hat{\boldsymbol{x}}$ for any given input $\boldsymbol{y}$. A common choice is to aim for the posterior mean $\hat{\boldsymbol{x}} = \mathbb{E}[\mathbf{x}|\mathbf{y} = \boldsymbol{y}]$, which is the predictor that minimizes the MSE. However, a single prediction does not convey to the user the uncertainty in the restoration, especially if the solution set is multi-modal. When confronted with such ambiguous tasks, it is more natural to predict a small set of prototypical restorations $\{\hat{\boldsymbol{x}}_1, \dots, \hat{\boldsymbol{x}}_K\}$ that portray the different options. In MCL, this is achieved by minimizing the so-called oracle/winner-takes-all loss given by

$$\mathcal{L}_{\text{MCL}}\left(\boldsymbol{x}, \{\hat{\boldsymbol{x}}_i\}_{i=1}^K\right) = \min_{i=1,\dots,K} \ell(\boldsymbol{x}, \hat{\boldsymbol{x}}_i), \tag{1}$$

where $\ell(\cdot,\cdot)$ is a loss function that measures prediction distance. Note that $\mathcal{L}_{\text{MCL}}$ is minimized as long as one restoration in the predicted set is close to $\boldsymbol{x}$. This encourages prediction diversity, such that each restoration specializes in a different mode of the posterior. Rupprecht et al. [45] proposed a probabilistic interpretation of MCL, showing that the optimal minimizers of Eq. (1) form a centroidal Voronoi tessellation (CVT) of the posterior, where $\{\hat{\boldsymbol{x}}_i^\star\}_{i=1}^K$ are given by the conditional means/centroids of the resulting K Voronoi cells (see Appendix E). In a discrete setting where we are given samples from $p_{\mathbf{x}|\mathbf{y}}(\boldsymbol{x}|\boldsymbol{y})$, CVT with the Euclidean distance is equivalent to K-means clustering.

Our goal here is to generalize the concept of predicting a set of restorations. Given an input $\boldsymbol{y}$, instead of predicting an unordered set $\{\hat{\boldsymbol{x}}_1(\boldsymbol{y}), \dots, \hat{\boldsymbol{x}}_K(\boldsymbol{y})\}$, we propose to output an input-adaptive tree $\mathcal{T}(\boldsymbol{y})$ representing a hierarchical clustering of the posterior $p_{\mathbf{x}|\mathbf{y}}(\boldsymbol{x}|\boldsymbol{y})$. Specifically, we want the tree nodes to correspond to a hierarchical CVT of the posterior (analogous to a hierarchical K-means process in the discrete setting), where the children of each node constitute a CVT for the cluster represented by the node. Such a tree can organize the different restorations in a manner that facilitates their navigation by a user and can also accompany each node with its relative cluster probability.

3.2 Posterior Trees

As discussed above, training a model to predict a tree of degree K and depth $d = 1$ with the oracle loss, results in a CVT of the posterior $p_{\mathbf{x}|\mathbf{y}}(\boldsymbol{x}|\boldsymbol{y})$ for every $\boldsymbol{y}$. Let $\{\mathcal{A}_k(\boldsymbol{y})\}_{k=1}^K$ denote the resulting Voronoi cells at $d = 1$ for some input $\mathbf{y} = \boldsymbol{y}$, such that $\cup_{k=1}^K \overline{\mathcal{A}_k(\boldsymbol{y})} = \mathcal{X}$, with $\mathcal{X} \in \mathbb{R}^{d_x}$ being the

 https://doi.org/10.52202/079017-3975

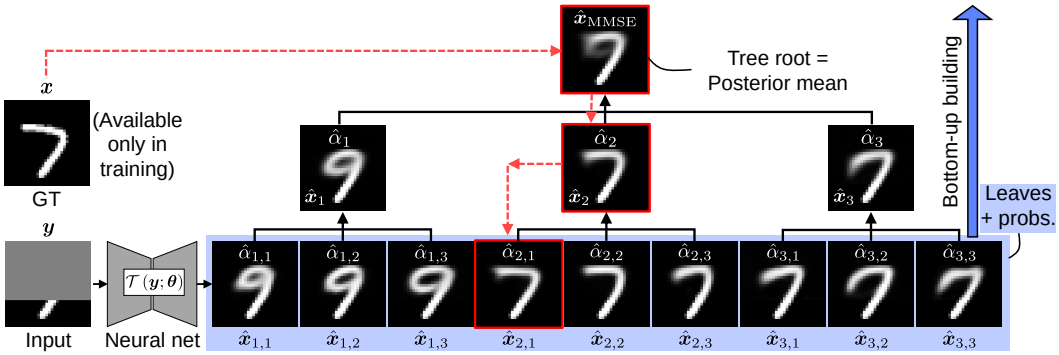


Figure 2: **Method overview**. Our model $\mathcal{T}(\boldsymbol{y};\boldsymbol{\theta})$ receives a degraded image $\boldsymbol{y}$ and predicts $\{\hat{\boldsymbol{x}}_{k_1,\ldots,k_d}\}_{k_1,\ldots,k_d=1}^{K}$, the bottom K^d leaves, and their probabilities $\{\hat{\alpha}_{k_1,\ldots,k_d}\}_{k_1,\ldots,k_d=1}^{K}$ (faint blue box; illustrated here for $K=3$ and $d=2$). Next, the tree is iteratively constructed from the bottom up using weighted averaging, until we reach the root node which is the minimum MSE predictor $\hat{\boldsymbol{x}}_{\text{MMSE}}$. During training, starting from the root, the ground truth $\boldsymbol{x}$ is propagated through the tree until it reaches the leaves (dashed red lines). At tree level d, x is compared to its immediate K children nodes, and the MSE loss to the nearest child is added to the loss trajectory.

output space. Training with ℓ_2 loss, *i.e.* $\ell(\hat{\boldsymbol{x}},\boldsymbol{x})=\|\hat{\boldsymbol{x}}-\boldsymbol{x}\|_2^2$, the optimal K predicted leaves and probabilities are given by [45, 50]

$$\begin{aligned}\hat{\boldsymbol{x}}_k^\star(\boldsymbol{y}) &= \mathbb{E}\left[\mathbf{x}|\mathbf{y}=\boldsymbol{y},\mathbf{x}\in\mathcal{A}_k(\boldsymbol{y})\right],\\ \hat{\alpha}_k^\star(\boldsymbol{y}) &= \mathbb{P}\left(\mathbf{x}\in\mathcal{A}_k(\boldsymbol{y})|\mathbf{y}=\boldsymbol{y}\right), \quad k=1,\ldots,K.\end{aligned} \tag{2}$$

We now want to extend the tessellation to the children of each node at the next level. Let $\{\mathcal{A}_{k,q}(\boldsymbol{y})\}_{q=1}^{K}$ denote the Voronoi cells at $d=2$, partitioning/tessellating the Voronoi cell $\mathcal{A}_k(\boldsymbol{y})$ from level 1 into K sub cells such that $\cup_{q=1}^{K}\overline{\mathcal{A}_{k,q}(\boldsymbol{y})}=\mathcal{A}_k(\boldsymbol{y})$. We want the K^2 leaves and their associated probabilities outputted by our model to be given by

$$\begin{aligned}\hat{\boldsymbol{x}}_{k,q}^\star(\boldsymbol{y}) &= \mathbb{E}\left[\mathbf{x}|\mathbf{y}=\boldsymbol{y},\mathbf{x}\in\mathcal{A}_{k,q}(\boldsymbol{y})\right],\\ \hat{\alpha}_{k,q}^\star(\boldsymbol{y}) &= \mathbb{P}\left(\mathbf{x}\in\mathcal{A}_{k,q}(\boldsymbol{y})|\mathbf{y}=\boldsymbol{y}\right), \quad k,q=1,\ldots,K.\end{aligned} \tag{3}$$

For simplicity of notation, in the following, we omit the dependence of the Voronoi cells on $\boldsymbol{y}$, and write $\mathcal{A}_k/\mathcal{A}_{k,q}$ instead. Equations (2) and (3) expose an interesting connection between tree levels. First, recall that since $\{\mathcal{A}_{k,q}\}_{k,q}$ is a tessellation, it satisfies $\cup_{q=1}^{K}\overline{\mathcal{A}_{k,q}}=\mathcal{A}_k$, and $\mathcal{A}_{k_1,q_1}\cap\mathcal{A}_{k_2,q_2}=\emptyset$ for $(k_1,q_1)\neq(k_2,q_2)$. Invoking the law of total expectation we can write

$$\mathbb{E}\left[\mathbf{x}|\mathbf{y}=\boldsymbol{y},\mathbf{x}\in\mathcal{A}_k\right]=\sum_{q=1}^{K}\mathbb{E}\left[\mathbf{x}|\mathbf{y}=\boldsymbol{y},\mathbf{x}\in\mathcal{A}_k,\mathbf{x}\in\mathcal{A}_{k,q}\right]\mathbb{P}\left(\mathbf{x}\in\mathcal{A}_{k,q}|\mathbf{y}=\boldsymbol{y},\mathbf{x}\in\mathcal{A}_k\right). \tag{4}$$

Equation (4) can be simplified by noting that $\mathcal{A}_k\cap\mathcal{A}_{k,q}=\mathcal{A}_{k,q}$, hence the expectation in the summand is given by $\mathbb{E}[\mathbf{x}|\mathbf{y}=\boldsymbol{y},\mathbf{x}\in\mathcal{A}_{k,q}]$. Moreover, we further simplify the probability in Eq. (4) by writing

$$\begin{aligned}\mathbb{P}\left(\mathbf{x}\in\mathcal{A}_{k,q}|\mathbf{y}=\boldsymbol{y},\mathbf{x}\in\mathcal{A}_k\right) &= \frac{\mathbb{P}\left(\mathbf{x}\in\mathcal{A}_k|\mathbf{y}=\boldsymbol{y},\mathbf{x}\in\mathcal{A}_{k,q}\right)\mathbb{P}(\mathbf{x}\in\mathcal{A}_{k,q}|\mathbf{y}=\boldsymbol{y})}{\mathbb{P}(\mathbf{x}\in\mathcal{A}_k|\mathbf{y}=\boldsymbol{y})}\\ &= \frac{\mathbb{P}(\mathbf{x}\in\mathcal{A}_{k,q}|\mathbf{y}=\boldsymbol{y})}{\sum_{j=1}^{K}\mathbb{P}(\mathbf{x}\in\mathcal{A}_{k,j}|\mathbf{y}=\boldsymbol{y})},\end{aligned} \tag{5}$$

where the first equality follows from Bayes' rule and the second equality is by the law of total probability and the fact that $\mathcal{A}_{k,q}\subseteq\mathcal{A}_k$, which implies that $\mathbb{P}(\mathbf{x}\in\mathcal{A}_k|\mathbf{y}=\boldsymbol{y},\mathbf{x}\in\mathcal{A}_{k,q})=1$.

Substituting Eq. (5) and the simplified expectation in Eq. (4) with the notations of Eqs. (2) and (3), we obtain that

$$\hat{\boldsymbol{x}}_k^\star(\boldsymbol{y}) = \sum_{q=1}^{K} \hat{\boldsymbol{x}}_{k,q}^\star(\boldsymbol{y}) \frac{\hat{\alpha}_{k,q}^\star(\boldsymbol{y})}{\sum_{j=1}^{K} \hat{\alpha}_{k,j}^\star(\boldsymbol{y})}, \tag{6}$$

$$\hat{\alpha}_k^\star(\boldsymbol{y}) = \sum_{q=1}^{K} \hat{\alpha}_{k,q}^\star(\boldsymbol{y}), \quad k = 1, \ldots, K, \tag{7}$$

where Eq. (7) is by the law of total probability. Note that Eqs. (6) and (7) expose the redundancy in predicting both $\{\hat{\boldsymbol{x}}_k^\star(\boldsymbol{y}), \hat{\alpha}_k^\star(\boldsymbol{y})\}$ and $\{\hat{\boldsymbol{x}}_{k,q}^\star(\boldsymbol{y}), \hat{\alpha}_{k,q}^\star(\boldsymbol{y})\}$, as given the latter, we can compute the former in closed form. Furthermore, although our derivation assumed a tree of depth $d = 2$, the result trivially generalizes to a tree of depth d. Given the leaves and their probabilities at level d, $\{\hat{\boldsymbol{x}}_{k_1,\ldots,k_d}^\star(\boldsymbol{y}), \hat{\alpha}_{k_1,\ldots,k_d}^\star(\boldsymbol{y})\}_{k_1,\ldots,k_d=1}^{K}$, we can compose their parents at the upper $d-1$ levels, by employing Eqs. (6) and (7) iteratively from the bottom up.

Here, this is precisely the property we exploit to output an input-adaptive tree $\mathcal{T}(\boldsymbol{y})$ of degree K and depth d, using a *single* model. Specifically, we train a model $\mathcal{T}(\boldsymbol{y};\boldsymbol{\theta}) \triangleq (\hat{\boldsymbol{X}}_d(\boldsymbol{y};\boldsymbol{\theta}), \hat{\boldsymbol{\alpha}}_d(\boldsymbol{y};\boldsymbol{\theta}))$ that outputs K^d prediction leaves $\hat{\boldsymbol{X}}_d(\boldsymbol{y};\boldsymbol{\theta}) = \{\hat{\boldsymbol{x}}_n(\boldsymbol{y};\boldsymbol{\theta})\}_{n=1}^{K^d}$ and their accompanying probabilities $\hat{\boldsymbol{\alpha}}_d(\boldsymbol{y};\boldsymbol{\theta}) = \{\hat{\alpha}_n(\boldsymbol{y};\boldsymbol{\theta})\}_{n=1}^{K^d}$ (see Fig. 2). Next, starting from the bottom leaves at depth d, the rest of the tree is composed level by level, recursively employing Eqs. (6) and (7) from the bottom up.

During training, the tree hierarchy is enforced through our loss function. Given some input $\boldsymbol{y}_i$, the model $\mathcal{T}(\boldsymbol{y};\boldsymbol{\theta})$ outputs are used as explained earlier to construct the full tree, where the root of the tree approximates the posterior mean $\mathbb{E}[\mathbf{x}|\mathbf{y} = \boldsymbol{y}_i]$. Next, the label $\boldsymbol{x}_i$ is first compared to the tree root with an ℓ_2 loss to ensure our decomposition recovers the MMSE estimator. At each successive level, the label $\boldsymbol{x}_i$ is compared to the children of the parent with minimal loss, and the closest child is added to the loss trajectory (Fig. 2). This process is repeated until we arrive at the bottom leaves.

Note that the described training scheme serves two purposes. First, it induces the desired tree structure by employing an amortized hierarchical version of the oracle loss in Eq. (1). Second, the loss employed on the root node enables predicting branch probability, mitigating the need for supervising the probabilities with explicit targets. See Appendix D for more details on our training algorithm and hierarchical oracle loss function.

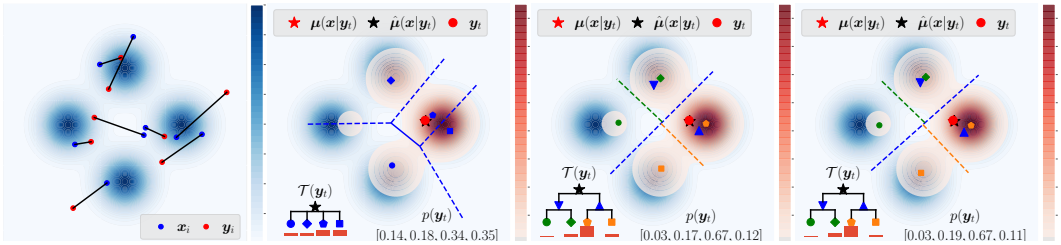
3.3 Architecture

The approach presented in Section 3.2 is general, and can be used to augment any architecture outputting K^d images and probabilities. Here, we choose to adapt the U-Net architecture [51] as a generic established choice for image-to-image regression tasks. The number of output images is K^d to accommodate a tree of degree K and depth d. In case all features are shared in the architecture, this amounts to changing the number of output filters at the last layer. However, in challenging tasks such as image inpainting, we found that sharing all parameters leads to reduced prediction diversity. In such cases, to trade off feature sharing and output diversity, we use group convolutions [52] in the decoder, such that each prediction has a disjoint set of channels learned separately from other outputs. As for the skip connections from the encoder, the concatenated features are interleaved equally per level such that each prediction has an equal share. In addition to the output images, our architecture also predicts K^d probabilities (see Figs. 2 and A2). This is achieved by (global) average pooling all feature maps from the decoder, and feeding their concatenation to an additional lightweight *multi-layer perceptron* (MLP), with four linear layers and a $\mathrm{softmax}$ at the output.

3.4 Preventing Tree Collapse

Vanilla training with the oracle loss is known to suffer from "hypotheses collapse", where some of the predictions are implausible [40, 42, 43, 45, 46, 50, 53, 54]. This phenomenon occurs when some predictions are initialized worse than others in stochastic training, and therefore receive little to no gradient updates due to not being chosen in the oracle loss. This results in degenerate outputs that are not encouraged to produce meaningful results. Previous works proposed various regularizations to remedy this undesired effect. For example, Rupprecht et al. [45] relaxed the $\arg\min$ operator with a small constant ε, such that predictions with non-minimal loss get updated with ε-scaled gradients.

https://doi.org/10.52202/079017-3975



(a) Denoising task (b) "Flat" K-means (c)Hierarchical K-means (d) Ours

Figure 3: **2D Gaussian mixture denoising**. (a) Underlying signal prior $p_{\mathbf{x}}(\boldsymbol{x})$ (blue heatmap), and training samples $(\boldsymbol{x}_i, \boldsymbol{y}_i) \sim p_{\mathbf{x},\mathbf{y}}(\boldsymbol{x}, \boldsymbol{y})$. (b) K-means with $K = 4$ applied to 10K samples $\boldsymbol{x}_i \sim p_{\mathbf{x}|\mathbf{y}}(\boldsymbol{x}|\boldsymbol{y}_t)$, for a given test point $\boldsymbol{y}_t$ (red circle). The resulting cluster centers (blue markers) partition the underlying posterior $p_{\mathbf{x}|\mathbf{y}}(\boldsymbol{x}|\boldsymbol{y}_t)$ (red heatmap), resulting in cluster probabilities $p(\boldsymbol{y}_t)$. (c) Hierarchical K-means applied twice with $K = 2$ on 10K samples $\boldsymbol{x}_i \sim p_{\mathbf{x}|\mathbf{y}}(\boldsymbol{x}|\boldsymbol{y}_t)$. At depth $d = 1$, the posterior is partitioned by the dashed blue line (blue triangles mark cluster centers). The resulting half spaces are subsequently halved by the dashed orange and green lines respectively. (d) Posterior trees (ours) with degree $K = 2$ and depth $d = 2$. Note that in all cases the estimated posterior mean $\hat{\boldsymbol{\mu}}(\boldsymbol{x}|\boldsymbol{y}_t)$ (black star) coincides with the analytical mean $\boldsymbol{\mu}(\boldsymbol{x}|\boldsymbol{y}_t)$ (red star), while in (c)-(d) the lowest density mode is better represented. $\mathcal{T}(\boldsymbol{y}_t)/p(\boldsymbol{y}_t)$ are drawn at the bottom of (b)-(d).

Moreover, Makansi et al. [54] proposed an evolving version of the oracle loss, where the top-k predictions are updated in each step, with k being annealed down to $k = 1$ during training, such that it starts with equal weights to all predictions and gradually shifts to updating only the best one towards convergence. Here, we employ a strategy that combines the simplicity of [45] with the adaptivity of [54]. Specifically, we scale the gradients of non-performing predictions with a constant ε that is annealed during training according to

$$\varepsilon_t = \varepsilon_0 \exp\{-[t - t_0]_+/2\}, \tag{8}$$

where t is the epoch number, and we fixed $\varepsilon_0 = 1$ and $t_0 = 5$ for all experiments. At the beginning of training, this has the benefit of bringing all predictions to a reasonable starting point, alleviating sensitivity to initialization and top prediction domination. As training progresses, this regularization is decayed, and only the relevant prediction is chosen for each sample, hence converging to the desired MCL behavior (see Appendix B).

3.5 Weighted Sampling

Albeit the regularization mentioned in Section 3.4, for tasks with highly imbalanced posteriors, training with Adam [55] leads to trees with a large disparity in leaf quality. This is because less likely leaves get chosen with lower frequency during training, resulting in transient gradients that highly affect the adaptive normalization in Adam. This hypothesis was also validated in matched settings by using SGD with an appropriate learning rate which resulted in significantly slower, yet improved convergence (see Appendix A.3). Therefore, to keep the speed advantage provided by Adam, in the task of image inpainting, we opted for a weighted sampler during training. The purpose of this sampler is to roughly balance out the overall number of occurrences at the bottom leaves during training. This is achieved by keeping track of an association matrix $\mathbf{A} \in \mathbb{R}^{K^d \times N}$, where $\mathrm{A}_{i,j}$ counts the number of times sample $\boldsymbol{x}_j$ was associated with leaf c_i over some time window. This allows us to estimate the conditional probability $p_{\mathbf{c}|\mathbf{x}}(c_i|\boldsymbol{x}_j)$, and subsequently derive an optimized sample probability to achieve a roughly uniform marginal leaf probability across the entire training set. The loss of each sample is then adjusted to account for this intervention, avoiding tampering with the original posterior probabilities. See Appendix C for more details.

4 Experiments

Here we demonstrate our method in several settings. In all experiments except for the toy example, we used variants of the U-Net architecture [51], with a custom number of output images, adjusted to accommodate K^d predictions according to the desired tree layout. Moreover, as explained in

Section 3.3, each model was also complemented with a small MLP for predicting leaf likelihood. Full details regarding the architectures, optimizer, and per-task settings are in Appendix A.

Toy Example. As a warm-up, we demonstrate posterior trees on a 2D denoising task. Here, $\mathbf{x}$ is sampled from a mixture of four Gaussians (arranged in a rhombus-like layout), and $\mathbf{y}$ is a noisy version of $\mathbf{x}$. The prior distribution $p_{\mathbf{x}}(\boldsymbol{x})$ and exemplar samples from $p_{\mathbf{x},\mathbf{y}}(\boldsymbol{x},\boldsymbol{y})$ are presented in Fig. 3(a). For this simple case, the posterior distribution $p_{\mathbf{x}|\mathbf{y}}(\boldsymbol{x}|\boldsymbol{y})$ can be calculated analytically (see Appendix G for the derivation) and is also a mixture of Gaussians. Therefore, this task can serve as a sanity check enabling us to benchmark our results. To demonstrate our method on this toy example, we train a model $\mathcal{T}(\boldsymbol{y};\boldsymbol{\theta})$ outputting a tree of depth $d = 2$ and degree $K = 2$. The results are compared against an approximate "ground truth" in Fig. 3(c). Note that even though we have a closed-form expression for the posterior density, we still have to approximate the "ground truth" posterior tree by applying hierarchical K-means clustering to samples for every $\boldsymbol{y}$. As seen in Fig. 3(d), our method resulted in highly accurate posterior trees both in terms of cluster centers and cluster likelihoods. Moreover, this result was achieved while being presented only with a *single* posterior sample per $\boldsymbol{y}$ in training. In contrast, the approximate ground truth was computed with $10K$ samples. Moreover, it is important to note that besides trivial cases, our result could not have been achieved with a "flat" K-means clustering with $K = 4$ (Fig. 3(b)). This is because K-means tends to focus on high-density modes, whereas our hierarchical trees better convey rare cases of the posterior. See Appendix G for more details and examples.

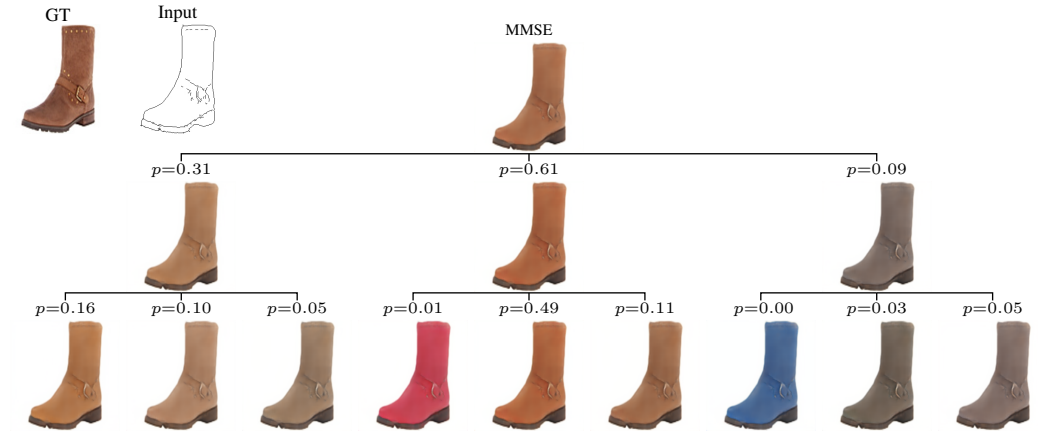
Handwritten Digits, Edges→Shoes, and Human Faces. Figure 2 demonstrates posterior trees on inpainting the top 70% of handwritten digits from the MNIST dataset. As can be seen, at depth $d = 1$, the learned tree exposes the two likely modes averaged in the mean estimate $\hat{\boldsymbol{x}}$, being either a "7" or a "9". In addition, at deeper tree levels, the different modes are further refined to reveal intricate intra-digit variations. More examples are available in Appendix M. We also applied posterior trees to the edges-to-shoes dataset taken from pix2pix [56, 57]. Here, the task is to convert an image of black and white edges to an output RGB image of a shoe. As shown in Fig. 4(a), our tree is capable of representing diverse shoe colors, in an adaptive manner to the input contours.

Next, we tested posterior trees on face images from the CelebA-HQ dataset, using the split from CelebA [58]. Figure 4(b) demonstrates our method applied to image colorization, where the input is a grayscale image and the desired output is its RGB version. The resulting trees hierarchically refine the output predictions by background, skin tone, and hat color. We also tested our method on the task of image inpainting. Figures 1 and 4(c) show the resulting trees for mouth/eye inpainting respectively. For example, the predicted tree in Fig. 4(c) explores the different options of eye-opening/closing, eyebrow-raising/lowering, eyeglasses, etc. This demonstrates that even shallow trees can still depict diverse reconstruction characteristics, showcasing the benefit of outputting multiple predictions.

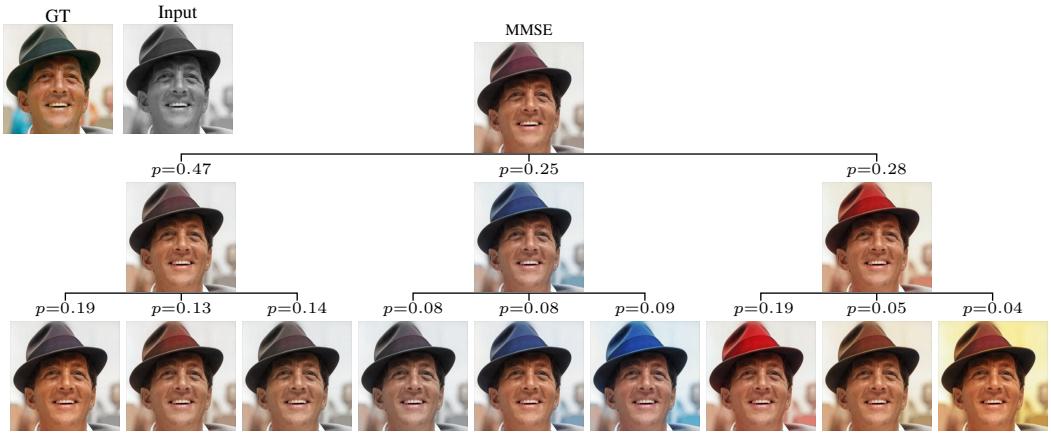
Bioimage Translation. Transforming images from one domain to match the statistics of images from another is commonly referred to as the task of image-to-image translation [57]. In the realm of bioimaging, such transformations were utilized to predict fluorescence from bright-field images [1], "virtually stain" unstained tissue [3], and transfer the images of one fluorescent dye to appear as if they were imaged with another [59]. The common scenario in image-to-image translation is the task being highly ill-posed with a wide range of plausible transformations satisfying the desired statistics. For general image editing/style transfer, this is a desired property adding to the artistic excitement; However, in bioimaging, this is highly problematic as the result often informs a downstream task with high stakes, and therefore output uncertainty should be communicated.

Here, we applied our method to a dataset of migrating cells, imaged in a spinning-disk microscope, simultaneously with two different fluorescent dyes (one staining the nuclei and one staining actin filaments) [59]. The task was to predict the image of one fluorescent dye (nuclear stain) from another (actin stain). Figure 5 demonstrates the predicted tree for a 128×128 test patch. The tree conveys important information to the user exposing uncertain cells, and exploring optional cell shapes. These can for example affect downstream tasks such as cell counting and morphological cell analysis.

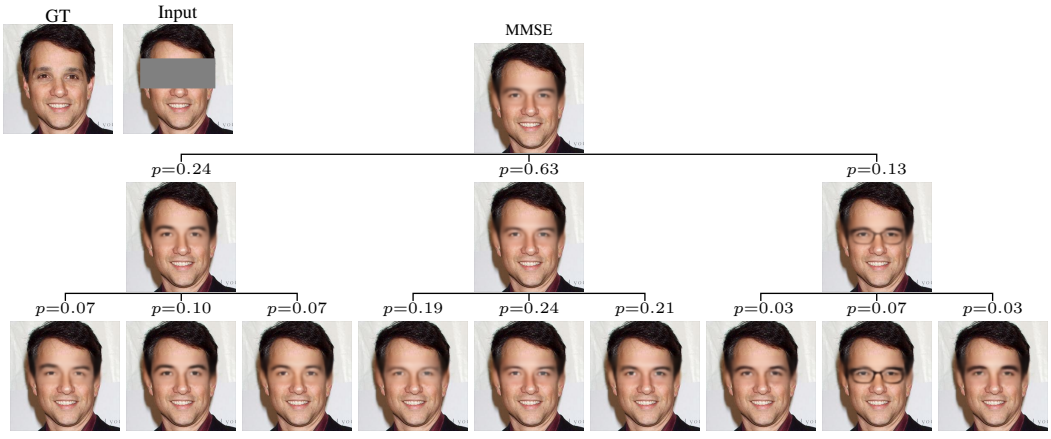
 https://doi.org/10.52202/079017-3975



(a) **Edges→Shoes**



(b) **Face colorization**



(c) **Eyes inpainting**

Figure 4: **Diverse applications of posterior trees**. The predicted trees represent inherent task uncertainty: *e.g.,* (a) Refining the mean estimate by color, grouping similar colors, while still depicting unlikely ones (*e.g.,* the blue boot); (b) Presenting various plausible colorizations varying by hat color, skin tone, and background; and (c) Exploring the diverse options of eyebrows/eyeglasses.

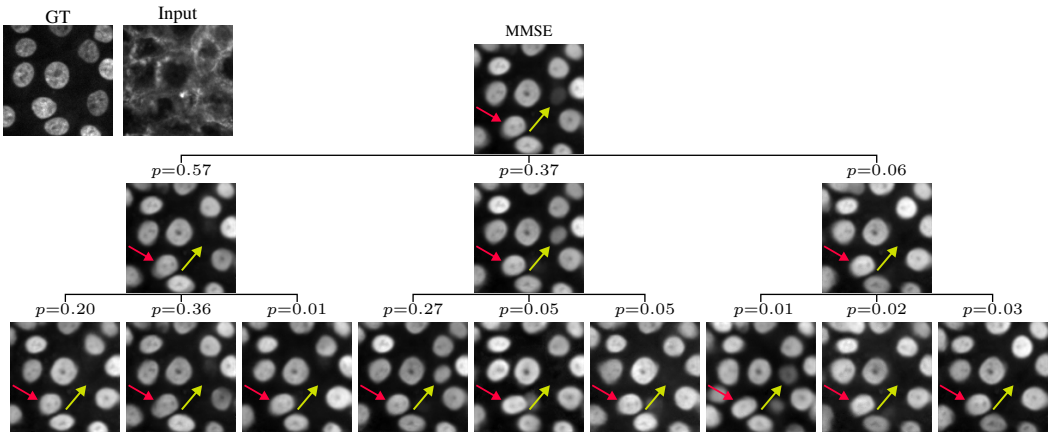


Figure 5: **Bioimage translation.** Here we explored posterior trees for the task of translating the image of a tissue from one fluorescent dye to another. The resulting trees expose important information regarding uncertain cells (yellow/red arrows), *e.g.,* ones that do not consistently appear in all branches, and additionally explore different plausible cellular morphology consistent with the input.

Table 1: Comparison to the proposed baseline on 100 test images from the FFHQ dataset. Hierarchical K-means was applied to 100 posterior samples per test image. Runtime is reported as both the speed of a forward pass (sec) and the memory usage (GB) required to infer a single test image with a batch of 1 on an A6000 GPU. The NLL at the root node ($d = 0$) is trivial and therefore omitted. Blue and Red indicate best and second best respectively.

Task	Method	Optimal PSNR (↑)			NLL (↓)		Speed (sec ↓)	Memory (GB ↓)
		$d=0$	$d=1$	$d=2$	$d=1$	$d=2$		
Color.	DDNM	**24.6±3.8**	**25.5±3.6**	**26.1±3.6**	**0.9±0.4**	**2.0±0.6**	340	18500
	DDRM	**22.7±2.7**	24.0±3.0	24.8±3.0	**1.1±0.4**	**2.0±0.5**	**68**	**3700**
	Ours	**24.6±4.1**	**25.7±3.9**	**26.4±4.0**	**0.9±0.4**	**2.0±0.7**	**0.014**	**1.3**
Mouth inpaint	DDNM	19.3±2.6	**19.9±2.2**	20.2±2.0	1.1±0.4	2.1±0.6	340	18500
	RePaint	**19.8±2.7**	**20.5±2.3**	**20.7±2.3**	**1.0±0.3**	**1.9±0.4**	15538	845450
	MAT	19.2±2.3	19.4±2.3	19.6±2.3	1.2±0.3	2.5±0.6	**15**	**300**
	Ours	**20.1±2.4**	**20.5±2.3**	**20.4±2.2**	**0.9±0.4**	**2.0±0.8**	**0.014**	**1.3**
Eyes inpaint	DDNM	19.3±2.7	19.6±2.5	19.5±3.7	**1.1±0.7**	**2.1±0.6**	340	18500
	RePaint	**20.1±2.8**	**20.4±2.8**	**20.6±2.8**	**1.0±0.4**	**1.9±0.5**	15538	845450
	MAT	19.0±3.0	19.2±3.0	19.3±3.0	1.2±0.3	2.4±0.6	**15**	**300**
	Ours	**19.6±2.7**	**19.9±2.5**	**19.8±2.4**	1.3±0.6	2.5±0.7	**0.014**	**1.3**

4.1 Quantitative Comparisons

Baseline. Building on the notion of clustering from Section 3.1, we propose a simple baseline to benchmark our results. As shown in Fig. 3(d), our predicted trees are constructed out of prototypes that yield a hierarchical clustering of the posterior. In discrete settings, this is equivalent to applying Hierarchical K-means on samples from $p_{\mathbf{x}|\mathbf{y}}(\boldsymbol{x}|\boldsymbol{y})$. Therefore, given a method to sample from the posterior (*e.g.,* using [7, 8, 10, 11]), a natural baseline in our case is a two-step procedure: (i) Generate N_s samples from the posterior, and (ii) Apply hierarchical K-means d times to generate a tree of degree K and depth d. Note that due to computational reasons, the comparison is performed over a random subset of 100 test images from FFHQ, where for each image we generate $N_s = 100$ samples, and apply hierarchical K-means with $K = 3$ and $d = 2$. To ensure a fair comparison, at each clustering step, K-means was run 5 times, keeping only the clusters with the best objective.

Metrics. To compare our predicted trees to the proposed baseline, we opted for two different metrics. The first metric is the PSNR (equivalent to MSE) between the ground truth test image and the tree nodes along the optimal path starting from the root and ending at the leaves. Intuitively, an accurate posterior clustering implies our tree nodes should maximize the PSNR, representing the ground truth test image with increasingly higher accuracy (lower MSE) as a function of (optimal)

https://doi.org/10.52202/079017-3975

node depth. The second metric we adopt is the sample negative log-likelihood (NLL) (using the natural logarithm) of the ground truth test image under the predicted posterior partitioning. Here, an accurate tree is expected to maximize the sample log-likelihood. This metric serves the purpose of verifying the predicted probabilities, as in practice we do not have access to the ground truth posterior distribution, and estimating cluster probability based on posterior samples becomes worse as a function of depth. Table 1 compares posterior trees to the proposed baseline, implemented with various state-of-the-art samplers. The results suggest that our method yields comparable performance in both the PSNR along the optimal path and in sample log-likelihood while requiring a single forward pass ($\approx$ 14 ms with a 1.3 GB memory footprint on a A6000 GPU). This is $10^3 - 10^7\times$ faster than the competition without considering our wide advantage in memory footprint. In case we do consider a batched setting with a sizable test set, our advantage becomes even more pronounced, where our method enables the inference of ≈ 430 test images in a single second. See Appendices I to L for more details and visual comparisons.

5 Discussion and Conclusion

We demonstrated the wide applicability of posterior trees across diverse tasks and datasets. However, our method is not free of limitations. First, the optimal hyper-parameters K and d are task-dependent, with no rule of thumb for determining them a-priori (see Appendix H). Second, in this work, we focused on balanced trees with a fixed degree K which might be sub-optimal for certain posteriors. Devising a strategy for an input-adaptive tree layout is an exciting direction for future research. Third, our method is limited in the number of output leaves, as we amortize the entire tree inference to a single forward pass. For predicting significantly deeper trees, an iterative inference procedure conditioning the model on the node index (in an analogous fashion to the timestep in diffusion models) is required. Finally, our method is tailored towards visualizing uncertainty and not sampling realistic-looking images. Although with increased depth our prototypes become increasingly sharper, they are nonetheless still cluster centers that average multiple plausible solutions and hence are not expected to lie on the image data manifold for shallow trees. A possible solution is to apply posterior trees in the latent space of an autoencoder such as VQ-VAE [60, 61], however, this is beyond the scope of this paper.

To conclude, in this work, we proposed a technique to output a hierarchical quantization of the posterior in a single forward pass. We discussed key design choices underlying our approach, including bottom-up tree construction, a principled training scheme, and proper regularization techniques to prevent tree collapse. We further demonstrated the benefit of hierarchical clustering over flat trees and discussed the intuition behind it on a toy example. In our experiments, we applied our method to highly ill-posed inverse problems and showed that diverse (prototypical) reconstructions are possible with a simple training scheme exposing uncertainty. Additionally, we also proposed an appropriate baseline based on posterior sampling, and quantitatively compared our approach to several strong samplers. Our method demonstrated at least comparable results while being orders of magnitude faster. Finally, we applied our method to the challenging task of bioimage translation, demonstrating its practical relevance in a real-world application.

Acknowledgments and Disclosure of Funding

This research was partially supported by the Israel Science Foundation (ISF) under Grant 2318/22 and by a gift from Elbit Systems. This work was done while Rotem Mulayoff was at the Technion. The authors also thank Hila Manor and Noa Cohen for their help with baseline comparisons.

References

[1] Chawin Ounkomol, Sharmishtaa Seshamani, Mary M Maleckar, Forrest Collman, and Gregory R Johnson. Label-free prediction of three-dimensional fluorescence images from transmitted-light microscopy. *Nature methods*, 15(11):917–920, 2018. 1, 7

[2] Eric M Christiansen, Samuel J Yang, D Michael Ando, Ashkan Javaherian, Gaia Skibinski, Scott Lipnick, Elliot Mount, Alison O'neil, Kevan Shah, Alicia K Lee, et al. In silico labeling: predicting fluorescent labels in unlabeled images. *Cell*, 173(3):792–803, 2018.

[3] Yair Rivenson, Hongda Wang, Zhensong Wei, Kevin de Haan, Yibo Zhang, Yichen Wu, Harun Günaydın, Jonathan E Zuckerman, Thomas Chong, Anthony E Sisk, et al. Virtual histological staining of unlabelled tissue-autofluorescence images via deep learning. *Nature biomedical engineering*, 3(6):466–477, 2019. 7

[4] Thorsten Falk, Dominic Mai, Robert Bensch, Özgün Çiçek, Ahmed Abdulkadir, Yassine Marrakchi, Anton Böhm, Jan Deubner, Zoe Jäckel, Katharina Seiwald, et al. U-net: deep learning for cell counting, detection, and morphometry. *Nature methods*, 16(1):67–70, 2019. 1

[5] Yang Song, Liyue Shen, Lei Xing, and Stefano Ermon. Solving inverse problems in medical imaging with score-based generative models. In *International Conference on Learning Representations*, 2021. 1, 2

[6] Guy Ohayon, Theo Adrai, Gregory Vaksman, Michael Elad, and Peyman Milanfar. High perceptual quality image denoising with a posterior sampling cgan. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1805–1813, 2021. 2

[7] Bahjat Kawar, Michael Elad, Stefano Ermon, and Jiaming Song. Denoising diffusion restoration models. In *Advances in Neural Information Processing Systems*, 2022. 9, 32

[8] 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. 2, 9, 32

[9] Hyungjin Chung, Jeongsol Kim, Michael Thompson Mccann, Marc Louis Klasky, and Jong Chul Ye. Diffusion posterior sampling for general noisy inverse problems. In *The Eleventh International Conference on Learning Representations*, 2022.

[10] 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. 9, 32

[11] Andreas Lugmayr, Martin Danelljan, Andres Romero, Fisher Yu, Radu Timofte, and Luc Van Gool. Repaint: Inpainting using denoising diffusion probabilistic models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11461–11471, 2022. 9, 32

[12] Chitwan Saharia, Jonathan Ho, William Chan, Tim Salimans, David J Fleet, and Mohammad Norouzi. Image super-resolution via iterative refinement. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2022.

[13] Chitwan Saharia, William Chan, Huiwen Chang, Chris Lee, Jonathan Ho, Tim Salimans, David Fleet, and Mohammad Norouzi. Palette: Image-to-image diffusion models. In *ACM SIGGRAPH 2022 Conference Proceedings*, pages 1–10, 2022.

 https://doi.org/10.52202/079017-3975

[14] Matthew Bendel, Rizwan Ahmad, and Philip Schniter. A regularized conditional gan for posterior sampling in image recovery problems. *Advances in Neural Information Processing Systems*, 36, 2024. 1, 2

[15] Noa Cohen, Hila Manor, Yuval Bahat, and Tomer Michaeli. From posterior sampling to meaningful diversity in image restoration. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=ff2g30cZxj. 1

[16] Vikash Sehwag, Caner Hazirbas, Albert Gordo, Firat Ozgenel, and Cristian Canton. Generating high fidelity data from low-density regions using diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11492–11501, 2022. 1

[17] Elias Nehme, Omer Yair, and Tomer Michaeli. Uncertainty quantification via neural posterior principal components. *Advances in Neural Information Processing Systems*, 36:37128–37141, 2023. 2

[18] Omer Belhasin, Yaniv Romano, Daniel Freedman, Ehud Rivlin, and Michael Elad. Principal uncertainty quantification with spatial correlation for image restoration problems. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2023. 2

[19] Hila Manor and Tomer Michaeli. On the posterior distribution in denoising: Application to uncertainty quantification. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=adSGeugiuj. 2

[20] Omer Yair, Elias Nehme, and Tomer Michaeli. Uncertainty visualization via low-dimensional posterior projections. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11041–11051, 2024. 2, 3

[21] Alex Kendall and Yarin Gal. What uncertainties do we need in bayesian deep learning for computer vision? *Advances in neural information processing systems*, 30, 2017. 2

[22] Guotai Wang, Wenqi Li, Michael Aertsen, Jan Deprest, Sébastien Ourselin, and Tom Vercauteren. Aleatoric uncertainty estimation with test-time augmentation for medical image segmentation with convolutional neural networks. *Neurocomputing*, 338:34–45, 2019.

[23] Anastasios N Angelopoulos, Amit P Kohli, Stephen Bates, Michael I Jordan, Jitendra Malik, Thayer Alshaabi, Srigokul Upadhyayula, and Yaniv Romano. Image-to-image regression with distribution-free uncertainty quantification and applications in imaging. In *International Conference on Machine Learning*, 2022. 2

[24] Andreas Lugmayr, Martin Danelljan, Luc Van Gool, and Radu Timofte. Srflow: Learning the super-resolution space with normalizing flow. In *ECCV*, 2020. 2

[25] Bahjat Kawar, Gregory Vaksman, and Michael Elad. Snips: Solving noisy inverse problems stochastically. *Advances in Neural Information Processing Systems*, 34:21757–21769, 2021.

[26] Hyungjin Chung, Suhyeon Lee, and Jong Chul Ye. Fast diffusion sampler for inverse problems by geometric decomposition. *arXiv preprint arXiv:2303.05754*, 2023.

[27] Berthy T Feng, Jamie Smith, Michael Rubinstein, Huiwen Chang, Katherine L Bouman, and William T Freeman. Score-based diffusion models as principled priors for inverse imaging. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 10520–10531, 2023. 2

[28] Aapo Hyvärinen and Peter Dayan. Estimation of non-normalized statistical models by score matching. *Journal of Machine Learning Research*, 6(4), 2005. 2

[29] Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *International conference on machine learning*, pages 2256–2265. PMLR, 2015.

[30] Yang Song and Stefano Ermon. Generative modeling by estimating gradients of the data distribution. *Advances in neural information processing systems*, 32, 2019.

[31] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020. 2

[32] Tim Salimans and Jonathan Ho. Progressive distillation for fast sampling of diffusion models. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=TIdIXIpzhoI. 2

[33] Eric Luhman and Troy Luhman. Knowledge distillation in iterative generative models for improved sampling speed. *arXiv preprint arXiv:2101.02388*, 2021.

[34] Chenlin Meng, Robin Rombach, Ruiqi Gao, Diederik Kingma, Stefano Ermon, Jonathan Ho, and Tim Salimans. On distillation of guided diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14297–14306, 2023.

[35] Yang Song, Prafulla Dhariwal, Mark Chen, and Ilya Sutskever. Consistency models. In *International Conference on Machine Learning*, pages 32211–32252. PMLR, 2023. 2

[36] Garoe Dorta, Sara Vicente, Lourdes Agapito, Neill DF Campbell, and Ivor Simpson. Structured uncertainty prediction networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5477–5485, 2018. 2

[37] Miguel Monteiro, Loïc Le Folgoc, Daniel Coelho de Castro, Nick Pawlowski, Bernardo Marques, Konstantinos Kamnitsas, Mark van der Wilk, and Ben Glocker. Stochastic segmentation networks: Modelling spatially correlated aleatoric uncertainty. *Advances in neural information processing systems*, 33:12756–12767, 2020.

[38] Chenlin Meng, Yang Song, Wenzhe Li, and Stefano Ermon. Estimating high order gradients of the data distribution by denoising. *Advances in Neural Information Processing Systems*, 34: 25359–25369, 2021.

[39] Frank Nussbaum, Jakob Gawlikowski, and Julia Niebling. Structuring uncertainty for fine-grained sampling in stochastic segmentation networks. *Advances in Neural Information Processing Systems*, 35:27678–27691, 2022. 2

[40] Abner Guzman-Rivera, Dhruv Batra, and Pushmeet Kohli. Multiple choice learning: Learning to produce multiple structured outputs. *Advances in neural information processing systems*, 25, 2012. 3, 5

[41] Abner Guzman-Rivera, Pushmeet Kohli, Dhruv Batra, and Rob Rutenbar. Efficiently enforcing diversity in multi-output structured prediction. In *Artificial Intelligence and Statistics*, pages 284–292. PMLR, 2014.

[42] Stefan Lee, Senthil Purushwalkam, Michael Cogswell, David Crandall, and Dhruv Batra. Why m heads are better than one: Training a diverse ensemble of deep networks. *arXiv preprint arXiv:1511.06314*, 2015. 3, 5

[43] Stefan Lee, Senthil Purushwalkam Shiva Prakash, Michael Cogswell, Viresh Ranjan, David Crandall, and Dhruv Batra. Stochastic multiple choice learning for training diverse deep ensembles. *Advances in Neural Information Processing Systems*, 29, 2016. 3, 5

[44] Kimin Lee, Changho Hwang, KyoungSoo Park, and Jinwoo Shin. Confident multiple choice learning. In *International Conference on Machine Learning*, pages 2014–2023. PMLR, 2017.

[45] Christian Rupprecht, Iro Laina, Robert DiPietro, Maximilian Baust, Federico Tombari, Nassir Navab, and Gregory D Hager. Learning in an uncertain world: Representing ambiguity through multiple hypotheses. In *Proceedings of the IEEE international conference on computer vision*, pages 3591–3600, 2017. 3, 4, 5, 6

[46] Michael Firman, Neill DF Campbell, Lourdes Agapito, and Gabriel J Brostow. Diversenet: When one right answer is not enough. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5598–5607, 2018. 5

[47] Eddy Ilg, Ozgun Cicek, Silvio Galesso, Aaron Klein, Osama Makansi, Frank Hutter, and Thomas Brox. Uncertainty estimates and multi-hypotheses networks for optical flow. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 652–667, 2018.

 https://doi.org/10.52202/079017-3975

[48] Kai Tian, Yi Xu, Shuigeng Zhou, and Jihong Guan. Versatile multiple choice learning and its application to vision computing. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6349–6357, 2019. 3

[49] Nitai Stein and Yaakov Oshman. Double-opportunity estimation via altruism. *IEEE Transactions on Aerospace and Electronic Systems*, 2022.

[50] Victor Letzelter, Mathieu Fontaine, Mickael Chen, Patrick Pérez, Slim Essid, and Gaël Richard. Resilient multiple choice learning: A learned scoring scheme with application to audio scene analysis. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. 3, 4, 5

[51] 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. 5, 6, 16

[52] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25, 2012. 5

[53] Mike Brodie, Chris Tensmeyer, Wes Ackerman, and Tony Martinez. Alpha model domination in multiple choice learning. In *2018 17th IEEE International Conference on Machine Learning and Applications (ICMLA)*, pages 879–884. IEEE, 2018. 5

[54] Osama Makansi, Eddy Ilg, Ozgun Cicek, and Thomas Brox. Overcoming limitations of mixture density networks: A sampling and fitting framework for multimodal future prediction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7144–7153, 2019. 5, 6

[55] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *International Conference on Learning Representations*, 3, 2015. 6, 16

[56] Aron Yu and Kristen Grauman. Fine-grained visual comparisons with local learning. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 192–199, 2014. 7

[57] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-image translation with conditional adversarial networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1125–1134, 2017. 7, 16

[58] 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)*, December 2015. 7

[59] 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. 7

[60] Aaron Van Den Oord, Oriol Vinyals, et al. Neural discrete representation learning. *Advances in neural information processing systems*, 30, 2017. 10

[61] Ali Razavi, Aaron Van den Oord, and Oriol Vinyals. Generating diverse high-fidelity images with vq-vae-2. *Advances in neural information processing systems*, 32, 2019. 10

[62] David Perera, Victor Letzelter, Théo Mariotte, Adrien Cortés, Mickael Chen, Slim Essid, and Gaël Richard. Annealed multiple choice learning: Overcoming limitations of winner-takes-all with annealing. *arXiv preprint arXiv:2407.15580*, 2024. 18

[63] Steven Diamond and Stephen Boyd. CVXPY: A Python-embedded modeling language for convex optimization. *Journal of Machine Learning Research*, 17(83):1–5, 2016. 21

[64] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 586–595, 2018. 23

[65] Ji Lin, Richard Zhang, Frieder Ganz, Song Han, and Jun-Yan Zhu. Anycost gans for interactive image synthesis and editing. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14986–14996, 2021. 23

[66] Qiang Du, Vance Faber, and Max Gunzburger. Centroidal voronoi tessellations: Applications and algorithms. *SIAM review*, 41(4):637–676, 1999. 25

[67] Yunjey Choi, Youngjung Uh, Jaejun Yoo, and Jung-Woo Ha. Stargan v2: Diverse image synthesis for multiple domains. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 8188–8197, 2020. 37

 https://doi.org/10.52202/079017-3975

Appendices

A Experimental Details

A.1 Architectures

As mentioned in Section 3.3, in our experiments we adopted the U-Net [51] architecture. Our architecture consisted of 4 downsampling/upsampling blocks. Downsampling by $2\times$ was performed using average pooling, and upsampling by $2\times$ was implemented with a nearest-neighbor interpolation. In both cases, the feature maps at the updated spatial resolution were processed by 2 convolution blocks consisting of 2D convolution, group normalization, and LeakyReLU activation with a negative slope of 0.2. The number of features at each spatial resolution was adapted to the predicted tree layout, *i.e.,* for predicting a tree of degree K and depth d, the initial number of channels before downsampling was set to $c_{\text{init}}K^d$, where K^d is the number of output leaves and $c_{\text{init}} \in \{4, 8\}$. At each successive encoder block, the number of channels was doubled, reaching $16c_{\text{init}}K^d$ at the bottleneck. For example, assuming a tree with degree $K = 3$ and depth $d = 2$, and setting $c_{\text{init}} = 4$, the number of channels per level in the encoder is given by $[36, 72, 144, 288, 576]$.

Similarly, in the decoder, the number of channels was halved at each upsampling step in a symmetric fashion. However, as mentioned in Section 3.3, in some tasks (*e.g.,* image inpainting), we noticed that fully sharing parameters between all leaves led to reduced diversity between predictions (see Fig. A1). On the other hand, learning a separate U-Net for each prediction leaf is computationally intensive, and highly inefficient as it is expected that the initial feature extraction stage for predicting the different leaves would be similar. Therefore, as a middle ground between these two extremes, we opted for an architecture that shares the encoder between the different leaves, while having disjoint decoders, each with a dedicated set of weights $\{\boldsymbol{\varphi}_n\}_{n=1}^{K^d}$, learned separately from others. As for the skip connections from the encoder, the concatenated feature maps are interleaved equally per level such that each prediction has an equal share.

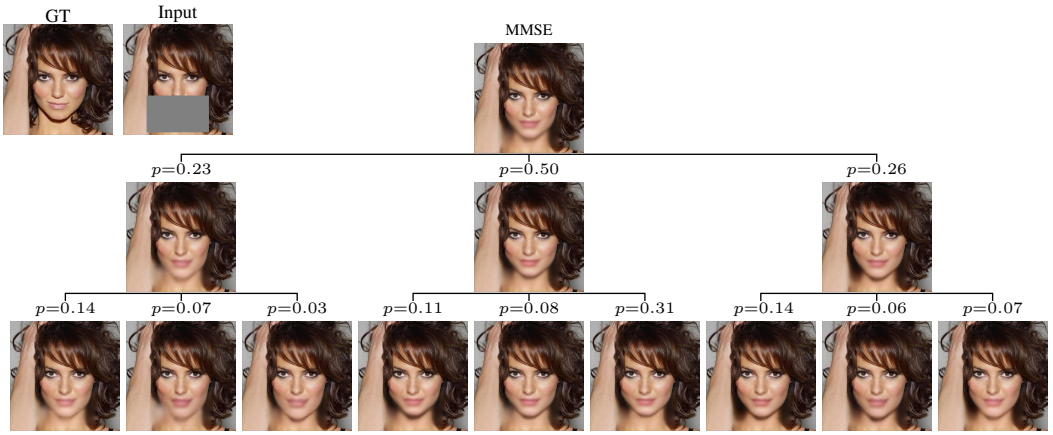
In addition to the output images, our architecture also predicts K^d scalars, which are the probabilities of the different leaves. This is achieved by global average pooling of all feature maps from the decoder, and feeding their concatenation to an additional lightweight MLP. This MLP has four linear layers with dimensions $[d_f, 256, 64, K^d]$, where d_f is the dimension of the concatenated pooled features from the decoder. Each linear layer is followed by a 1D batch normalization and SiLU non-linearity, and the output is passed through a $\mathrm{softmax}$ layer to produce a valid probability vector. Figure A2 summarizes our architecture.

A.2 Per-task Details

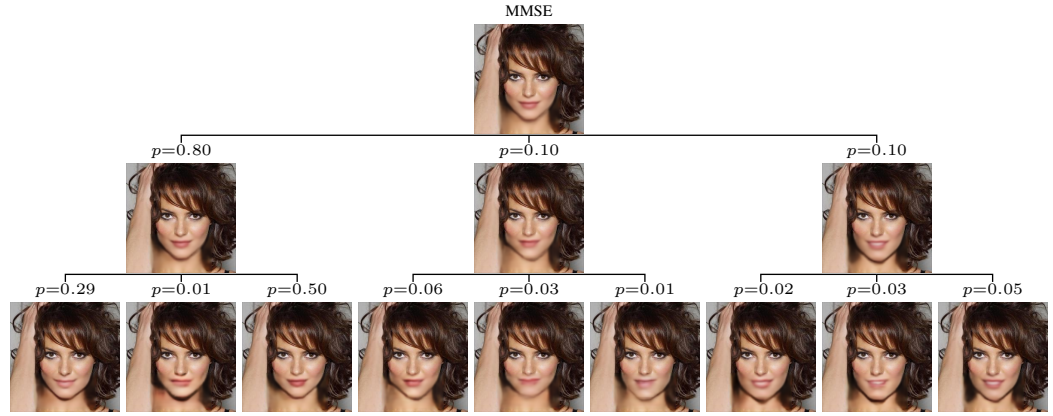
In all tasks, we only learned the required residual from the input to produce the predictions. For MNIST experiments, the images were padded to 32×32 to enable proper downsampling in the encoder. For Edges→Shoes experiments, the images were kept the same as in the pix2pix [57] paper, with a resolution of 256×256. For CelebA-HQ experiments, the images were resized to 256×256. Finally, for the bioimage translation dataset, we trained on 128×128 patches cropped from the full 1024×1024 images, as cell information tends to be local.

A.3 Optimization

We used the Adam optimizer [55] with $\beta_1 = 0.9, \beta_2 = 0.999$ for all experiments. For the U-Net predicting the output leaves, we used an initial step size of 0.001. For the MLP predicting leaf probability, however, we found it beneficial to use a smaller initial step size of 0.0002. This is because at the initial training phase, the predicted leaves are still not converged; hence, the MLP can easily "classify" which leaf is the most likely. This collapses the predicted probabilities to a sparse vector, leading to leaves with zero probability. Hence, to avoid this instability, we used a lower step size for the probabilities such that the leaves are first allowed to converge, leading to the desired learning dynamics. For both components, the step size was reduced by a factor of 10 if the validation loss stagnated for more than 10 epochs, and the minimum step size was set to $5 \cdot 10^{-6}$. We used a batch size of 32 for 70 epochs for all tasks. This resulted in training times of ≈40 mins, 10 hrs, 5 hrs, and 7 hrs for MNIST, Edges→Shoes, CelebA-HQ, and the bioimage datasets respectively.



(a) Fully shared



(b) Encoder shared

Figure A1: **Leaf weight sharing strategy.** (a) Fully shared architecture, with all leaves predicted jointly. (b) Leaves only share encoder (see Fig. A2).

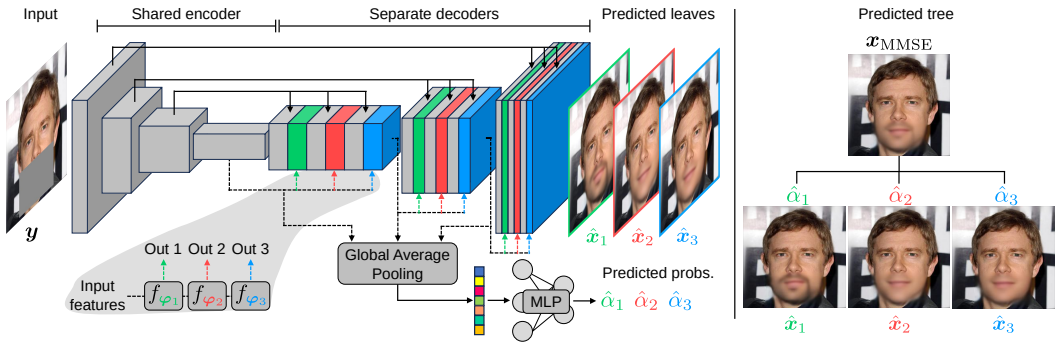
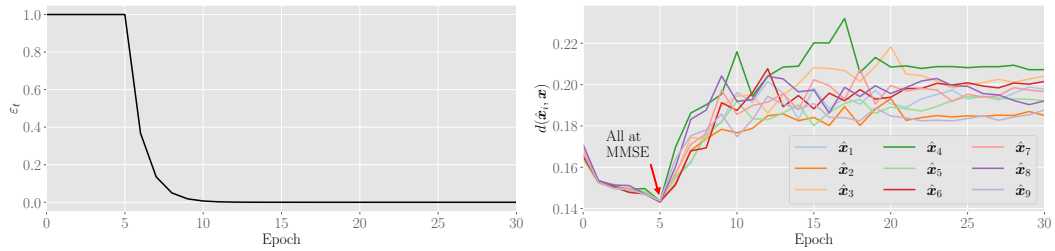


Figure A2: **Model architecture**. Our model receives a degraded image $\boldsymbol{y}$ and predicts the bottom K^d leaves and their probabilities $\{\hat{\boldsymbol{x}}_{k_1,\ldots,k_d}, \hat{\alpha}_{k_1,\ldots,k_d}\}_{k_1,\ldots,k_d=1}^{K}$ (illustrated here for $K=3$ and $d=1$). The encoder is shared between all leaves, however, in the decoding stage, each leaf has a separate decoder (*e.g.,* with parameters φ_1,φ_2,φ_3) to enable diverse outputs. To predict leaf probability, all features from the decoder are passed through a global average pooling layer, and the resulting concatenated vector is passed to a lightweight MLP with a softmax layer at the output. Afterward, as explained earlier, the output tree is constructed iteratively from the bottom up (right).

https://doi.org/10.52202/079017-3975

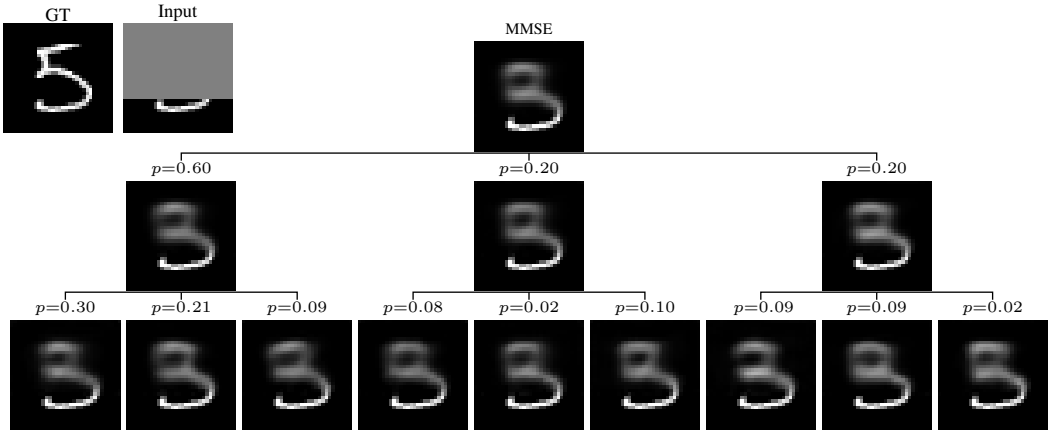
B Role of ε_t From Eq. (8)

As explained in Section 3.4, the convergence of MCL is strongly affected by initialization. This is because leaves that are better initialized are more likely to be chosen in the oracle loss, and hence will dominate the remaining leaves in training. This results in output leaves that practically do not train, and therefore give meaningless results at test time. To remedy this, we scale the gradients of non-performing predictions with a constant ε that is annealed during training according to Eq. (8). For the first $t_0 = 5$, our training degenerates to standard MSE minimization, bringing all leaves to a reasonable starting point near the posterior mean (Fig. A3(c)). This results in all prediction leaves having roughly the same initialization, with a similar number of associated training samples. Afterward, ε is gradually decayed, and the leaves start to specialize in their respective posterior mode, converging to the desired MCL behavior (Fig. A3(d)). In this research, we identified this strategy as a simple yet effective method of regularization. However, it is important to highlight that a parallel study by Perera et al. [62] presents a potentially improved solution that combines soft assignments with deterministic annealing. This new approach could be easily integrated with our method, and it would be intriguing to explore this in future work.

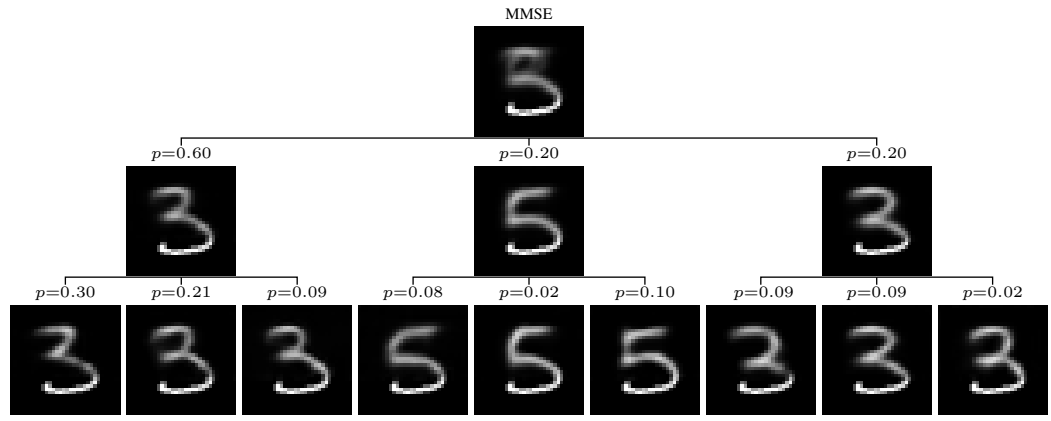


(a) ε_t

(b) Distance of leaves $\{\hat{\boldsymbol{x}}_i\}$ from the GT $\boldsymbol{x}$



(c) Predicted tree at epoch $t = 5$



(d) Predicted tree at convergence (epoch $t = 22$)

Figure A3: **Role of** ε_t. (a) ε_t. (b) Distances of leaf predictions $\{\hat{\boldsymbol{x}}_i\}$ from the GT $\boldsymbol{x}$ throughout epochs. For the first $t_0 = 5$ epochs all predictions are brought to the vicinity of the MMSE estimator. Afterward, ε_t is decayed and each of the leaves is free to specialize in a subset of the posterior, leading to higher distances on average for the entire test set. (c) Predicted tree at epoch $t = t_0 = 5$ where all leaves are near the MMSE estimator. (d) Predicted tree at convergence (epoch $t = 22$) where each leaf specializes in a different posterior mode (*e.g.,* a "3" or a "5").

https://doi.org/10.52202/079017-3975

C Weighted Sampler

For imbalanced posteriors where the dominant mode is significantly more likely than the weakest mode (*e.g.,* $10\times$), we found that an additional regularization complementary to the one provided by ε_t is needed. This is because even if we eliminate the dependence of training on leaf initialization, still, leaves associated with (much) less likely posterior modes will be chosen with a lower frequency during training, resulting in transient gradients that highly affect the adaptive normalization in Adam. To test this hypothesis, we trained two models on the task of image inpainting: (i) one with Adam using an initial learning rate of 0.001, and (ii) one with SGD using a momentum of 0.9 and an initial learning rate of 0.1. Figure A4 demonstrates the results of this experiment. As evident in the resulting trees, Adam leads to highly implausible predictions in leaves with low likelihoods (*e.g.,* $p = 0.0$ in Fig. A4(c)). SGD, On the other hand, does not lead to nonsensical predictions; however, it requires significantly longer training time and is more challenging to converge (Fig. A4(d))

Therefore, in our method, we proposed a novel non-stochastic weighted sampling scheme as a simple fix (Fig. A4(e)). The proposed weighted sampler enables us to still enjoy the convergence speed of Adam while tackling the aforementioned optimization deficiency of MCL training for posteriors that are far from uniform. The goal of our sampler is to ensure that on average the number of occurrences at each output leaf during training is roughly the same (Fig. A4(b)). This is done by undersampling training samples associated with leaves that represent probable posterior modes while oversampling training samples associated with leaves that represent rare posterior modes.

Formally, let c_i denote the ith prediction leaf ($i = 1, \dots, K^d$), and $\boldsymbol{s}_j = (\boldsymbol{x}_j, \boldsymbol{y}_j)$ denote the jth (paired) training sample. We assume we are given a training set $\mathcal{D} = \{\boldsymbol{s}_j\}_{j=1}^N$ of N *i.i.d.* samples, such that $p_{\mathbf{s}}(\boldsymbol{s}_j) = \frac{1}{N}$. The goal of the sampler is to manipulate $p_{\mathbf{s}}(\boldsymbol{s}_j)$ via oversampling/undersampling, such that the new sample probability in training $q_{\mathbf{s}}(\boldsymbol{s}_j)$, leads to a uniform marginal leaf distribution, *i.e.,* $q_{\mathbf{c}}(c_i) = \frac{1}{K^d}$. To estimate the marginal leaf probability during training, we keep track of an association matrix $\boldsymbol{A} \in \mathbb{R}^{K^d \times N}$, where $A_{i,j}$ counts the number of times sample $\boldsymbol{s}_j$ was associated with leaf c_i. Specifically, after each training batch $\boldsymbol{b}_\tau$ of size $B = |\boldsymbol{b}_\tau|$, the association matrix is updated with momentum $\mu = 1 - 2^{-\frac{B}{N}}$, such that

$$\boldsymbol{A}_0^{\text{EMA}} = \mathbf{0}, \qquad \boldsymbol{A}_\tau^{\text{EMA}} = \mu \boldsymbol{A}_{\tau-1}^{\text{EMA}} + (1-\mu)\boldsymbol{A}_\tau, \tag{A1}$$

where $\boldsymbol{A}_\tau$ is a binary matrix that encodes the sample-leaf association for the batch $\boldsymbol{b}_\tau$. This smoothing is necessary to accumulate statistics across epochs, making sample probability change gracefully while avoiding large changes to the probability of uncertain samples that switch leaf association between batches. Next, given the updated association matrix $\boldsymbol{A}_\tau^{\text{EMA}}$, we can normalize it to obtain an estimate of the current joint distribution $p_{\mathbf{c},\mathbf{s}}(c_i, \boldsymbol{s}_j)$ as

$$p_{\mathbf{c},\mathbf{s}}(c_i, \boldsymbol{s}_j) = \frac{A_{i,j}^{\text{EMA}}}{\sum_{p=1}^{K^d} \sum_{\ell=1}^{N} A_{p,\ell}^{\text{EMA}}}. \tag{A2}$$

Manipulating the sample probability $p_{\mathbf{s}}(\boldsymbol{s}_j)$ will affect this statistic. Here we can assume that the conditional distribution over the leaves $\{c_i\}_{i=1}^{K^d}$ given a sample $\boldsymbol{s}_j$ stays the same under the change of distribution for the samples. This insight can then be leveraged as follows. First, we obtain the current conditional leaf probability via marginalization

$$p_{\mathbf{c}|\mathbf{s}}(c_i|\boldsymbol{s}_j) = \frac{p_{\mathbf{c},\mathbf{s}}(c_i, \boldsymbol{s}_j)}{\sum_{i=1}^{K^d} p_{\mathbf{c},\mathbf{s}}(c_i, \boldsymbol{s}_j)}. \tag{A3}$$

Let $\boldsymbol{P} \in \mathbb{R}^{K^d \times N}$ denote the matrix where $P_{i,j} = p_{\mathbf{c}|\mathbf{s}}(c_i|\boldsymbol{s}_j)$. Recall that our goal is to dictate a new sample probability $q_{\mathbf{s}}(\boldsymbol{s}_j)$ such that the induced marginal leaf probability $q_{\mathbf{c}}(c_i)$ is uniform. Let $\boldsymbol{Q}^{\mathbf{c}|\boldsymbol{s}} \in \mathbb{R}^{K^d \times N}$ denote the conditional distribution matrix under probability q such that $Q_{i,j}^{\mathbf{c}|\boldsymbol{s}} = q_{\mathbf{c}|\mathbf{s}}(c_i|\boldsymbol{s}_j)$. In matrix form, we are looking for a probability vector over the samples $\boldsymbol{q}^{\boldsymbol{s}} \in \mathbb{R}^N$, that satisfies $\boldsymbol{Q}^{\mathbf{c}|\boldsymbol{s}}\boldsymbol{q}^{\boldsymbol{s}} = \boldsymbol{q}^{\mathbf{c}} = \frac{1}{K^d}\mathbf{1}_{K^d \times 1}$. Since the conditional probability of the leaves is assumed to be not affected by the change from $p_{\mathbf{s}}$ to $q_{\mathbf{s}}$, we have that $\boldsymbol{Q}^{\mathbf{c}|\boldsymbol{s}} = \boldsymbol{P}$. Therefore, in simpler notations, we are searching for a probability vector $\boldsymbol{q} \in \mathbb{R}^N$ for which

$$\boldsymbol{P}\boldsymbol{q} = \frac{1}{K^d}\mathbf{1}_{K^d \times 1}. \tag{A4}$$

Naturally, since $K^d \ll N$, there are infinitely many different options $\boldsymbol{q}$ that satisfy this constraint. Note that some of them can correspond to an imbalanced distribution, where some samples associated with a certain leaf have high probability, while other samples that are associated with the same leaf have low probability. Therefore, among all the different options that satisfy Eq. (A4), we want the one that has the flattest probability. The flattest distribution minimizes the ℓ_2 loss, namely $\|\boldsymbol{q}\|_2^2$. However, minimizing this objective under the constraint in Eq. (A4) can be problematic for ill-conditioned cases. Nevertheless, we can achieve similar results by solving the following Tikhonov-regularized relaxed problem:

$$\begin{aligned} \min_{\boldsymbol{q}\in\mathbb{R}^N} \quad & \frac{1}{2}\|\boldsymbol{P}\boldsymbol{q} - \frac{1}{K^d}\mathbf{1}\|_2^2 + \frac{\lambda}{2}\|\boldsymbol{q}\|_2^2 \\ \text{s.t.} \quad & \boldsymbol{q}_j \geq 0, \quad j = 1, \ldots, N \\ & \boldsymbol{q}^\top \mathbf{1}_{N\times 1} = 1 \end{aligned} \tag{A5}$$

One can obtain an exact solution $\boldsymbol{q}^\star$ to Eq. (A5) using standard methods for convex optimization, *e.g.* algorithms in CVXPY [63]. However, in typical settings, this results in longer training times. Therefore, we opted to use an approximation. Specifically, if we drop the positivity constraint $\boldsymbol{q}_j \geq 0, \forall j$, and solve Eq. (A5) accounting only for the constraint $\boldsymbol{q}^\top \mathbf{1}_{N\times 1} = 1$, we obtain the following closed-form solution

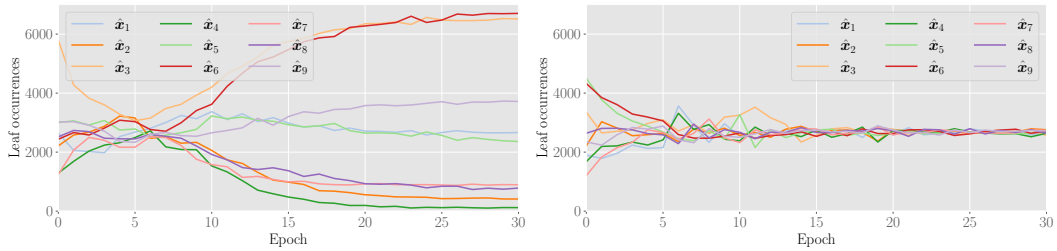
$$\begin{aligned} \tilde{\boldsymbol{q}} &= \left(\boldsymbol{I}_{N\times N} - \boldsymbol{P}^\top \left(\boldsymbol{P}\boldsymbol{P}^\top + \lambda \boldsymbol{I}_{K^d\times K^d}\right)^{-1} \boldsymbol{P}\right) \mathbf{1}_{N\times 1} \\ \boldsymbol{q}^\diamond &= \frac{1}{\tilde{\boldsymbol{q}}^\top \mathbf{1}_{N\times 1}} \tilde{\boldsymbol{q}}. \end{aligned} \tag{A6}$$

In practice, in most cases, the positivity constraints are inactive, and therefore $\boldsymbol{q}^\diamond = \boldsymbol{q}^\star$. In the rare cases where Eq. (A6) resulted in negative values, we clipped entries below zero in $\boldsymbol{q}^\diamond$ and renormalized the result.

Given the desired sampling distribution $q_{\mathbf{s}}(\boldsymbol{s}_j)$ we pick for the next batch $\boldsymbol{b}_{\tau+1}$ the B distinct samples that their selection will minimize the maximal distance between the future statistics at time $\tau + 1$ and $q_{\mathbf{s}}$. Specifically, at each training step, the next batch of samples is chosen such that the maximal difference between the next iteration estimated sample probability $p_{\mathbf{s}}(\boldsymbol{s}_j) = \sum_{i=1}^{K^d} p_{\mathbf{c},\mathbf{s}}(c_i, \boldsymbol{s}_j)$ and the desired sample probability $q_{\mathbf{s}}(\boldsymbol{s}_j)$, *i.e.,* $\|p_{\mathbf{s}} - q_{\mathbf{s}}\|_\infty$, is minimized. In practice, fetching the samples for future batches is done in advance to speed up the training time. Therefore, to prevent the training from waiting for the successor samples, we use a buffer of m batches ahead and adjust the implementation accordingly. During training, we activated our weighted sampler one epoch after ε_t started decaying to 0.

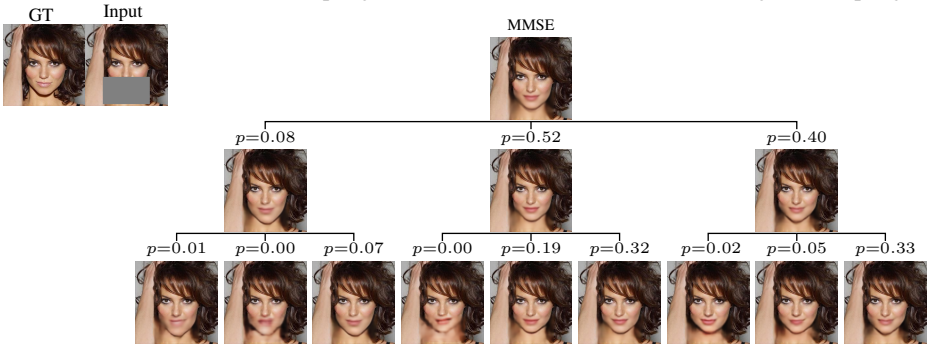
Finally, note that switching from $p_{\mathbf{s}}$ to $q_{\mathbf{s}}$ during training effectively changes the dataset by repetition/omission, and if not accounted for will distort the learned posterior probabilities. Therefore, to undo this undesired effect while still maintaining an increased number of optimization steps for weak posterior modes, we scaled the loss of sample $\boldsymbol{s}_j$ by the factor $\gamma_j = \frac{1}{N\boldsymbol{q}_{\mathbf{s}}(\boldsymbol{s}_j)}$. Hence, we effectively only changed the number of optimization steps taken per output leaf, enabling all leaves to continuously receive gradients and train equally as well with Adam.

 https://doi.org/10.52202/079017-3975

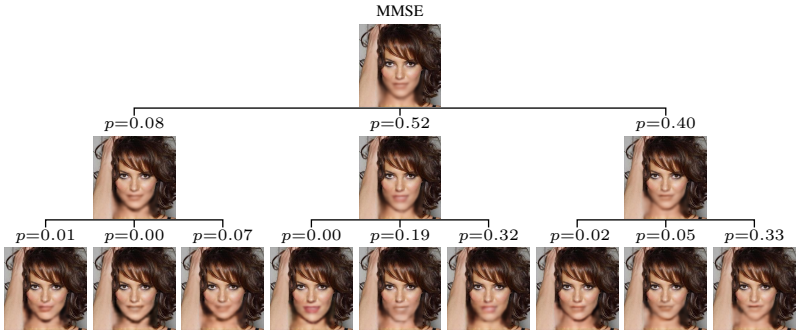


(a) Standard sampling

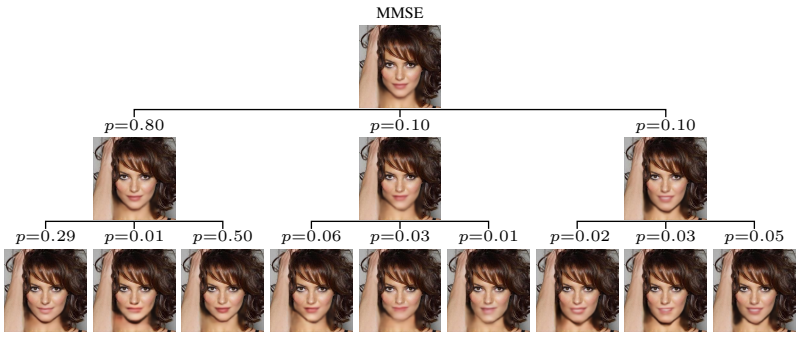
(b) Weighted sampling



(c) Adam with standard sampling



(d) SGD with standard sampling



(e) Adam with weighted sampling

Figure A4: **Weighted sampling effect.** Optimization with Adam requires weighted sampling to train properly (see text in Appendix C).

D Tree Loss Function and Training Scheme

Our tree training algorithm is summarized in Algorithm 1. Throughout this work, we used the MSE loss $\ell(\boldsymbol{x}, \hat{\boldsymbol{x}}) = \|\boldsymbol{x} - \hat{\boldsymbol{x}}\|_2^2$ to both determine the closest tree node and as our optimization loss. However, in general, a distinction should be made between the measure $\ell(\cdot, \cdot)$ used for the clustering (i.e. determining the associations of samples to clusters) and the loss used within each cluster to determine the cluster representative. In theory, our approach can work without modification with any association measure (e.g. LPIPS [64], some domain-specific classifier [65], etc.). However, changing the within-cluster loss requires some modifications. Specifically, the use of the MSE loss is what provides us with the hierarchical decomposition of the posterior mean (see Eqs. (2) to (7)). In particular, when using the MSE/L_2 loss, each cluster representative becomes the posterior cluster mean, and a weighted combination of those representatives gives the overall posterior mean (which is the tree root). This allows us to have the network output only the leaves of the tree, which implicitly defines the entire tree (as the nodes of each level are obtained as linear combinations of the nodes of its children).

One natural extension to investigate using our approach is to take the association measure to be $\ell(\hat{\boldsymbol{x}}, \boldsymbol{x}) = \|f(\hat{\boldsymbol{x}}) - f(\boldsymbol{x})\|_2^2$, where $f(\cdot)$ is some relevant domain-specific feature extractor. Specifically, we experimented with the deep features of the AnyCostGAN attribute predictor [65], and employed it in the task of eyes inpainting using the CelebAHQ dataset (Fig. A5). This preliminary result indicates that our method could potentially be used with other association metrics. However, further in-depth analysis is necessary to ensure semantically meaningful results, which will be addressed in future work.

Algorithm 1 Tree Training

Require: Tree degree K, Tree depth d, Network architecture $\mathcal{T}(\cdot; \boldsymbol{\theta})$, Training set $\mathcal{D} = \{(\boldsymbol{x}_i, \boldsymbol{y}_i)\}$, Distance measure $\ell(\cdot, \cdot)$, Loss weights $\boldsymbol{\beta} \in \mathbb{R}^{d+1}$.

repeat

`/* Sample data and infer leaves */`

$(\boldsymbol{x}_i, \boldsymbol{y}_i) \sim \mathcal{D}$ ▷ Sample paired training example

$\{\hat{\boldsymbol{x}}_n(\boldsymbol{y}_i; \boldsymbol{\theta}), \hat{\alpha}_n(\boldsymbol{y}_i; \boldsymbol{\theta})\}_{n=1}^{K^d} \leftarrow \mathcal{T}(\boldsymbol{y}_i; \boldsymbol{\theta})$ ▷ Infer tree leaves and their probabilities

`/* Bottom-up tree construction using Eqs. (6) and (7) */`

Run $\texttt{BottomUp}(\{\hat{\boldsymbol{x}}_n(\boldsymbol{y}_i; \boldsymbol{\theta}), \hat{\alpha}_n(\boldsymbol{y}_i; \boldsymbol{\theta})\}_{n=1}^{K^d})$ to get:

$$\hat{\boldsymbol{x}}_{\text{MMSE}}(\boldsymbol{y}_i), \{\hat{\boldsymbol{x}}_{k_1}(\boldsymbol{y}_i), \hat{\alpha}_{k_1}(\boldsymbol{y}_i)\}_{k_1=1}^{K}, \ldots, \{\hat{\boldsymbol{x}}_{k_1,\ldots,k_d}(\boldsymbol{y}_i), \hat{\alpha}_{k_1,\ldots,k_d}(\boldsymbol{y}_i)\}_{k_1,\ldots,k_d=1}^{K}$$

`/* Accumulate x̂ along min path */`

$\hat{\mathcal{X}} \leftarrow \{\}$

$\hat{\boldsymbol{x}}_{k^\star}(\boldsymbol{y}_i) \leftarrow \hat{\boldsymbol{x}}_{\text{MMSE}}(\boldsymbol{y}_i)$ ▷ Initialize closest tree node to root

repeat d **times**

$\hat{\mathcal{X}} \leftarrow \hat{\mathcal{X}} \cup \hat{\boldsymbol{x}}_{k^\star}(\boldsymbol{y}_i)$ ▷ Add closest node to min path

$\{\hat{\boldsymbol{x}}_k(\boldsymbol{y}_i)\}_{k=1}^{K} \leftarrow \texttt{GetChildren}(\hat{\boldsymbol{x}}_{k^\star}(\boldsymbol{y}_i))$ ▷ Get K children of closest node

$k^\star \leftarrow \underset{k=1,\ldots,K}{\arg\min}\, \ell(\boldsymbol{x}_i, \hat{\boldsymbol{x}}_k(\boldsymbol{y}_i))$ ▷ Update closest node to closest child

end

$\hat{\mathcal{X}} \leftarrow \hat{\mathcal{X}} \cup \hat{\boldsymbol{x}}_{k^\star}(\boldsymbol{y}_i)$ ▷ Include also the closest leaf

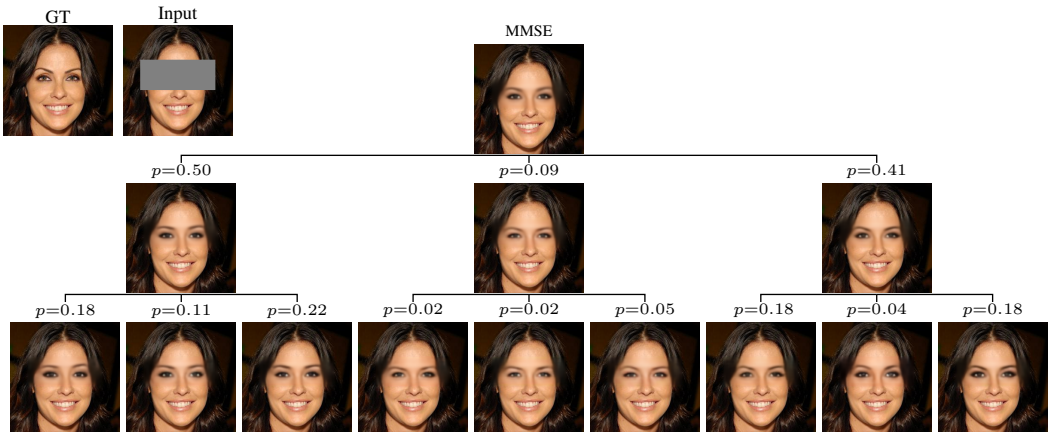
`/* Take gradient step on weighted MSE loss to update θ */`

$\mathcal{L}(\boldsymbol{\theta}) = \sum_{j=1}^{d+1} \boldsymbol{\beta}_j \|\hat{\mathcal{X}}_j - \boldsymbol{x}_i\|_2^2$

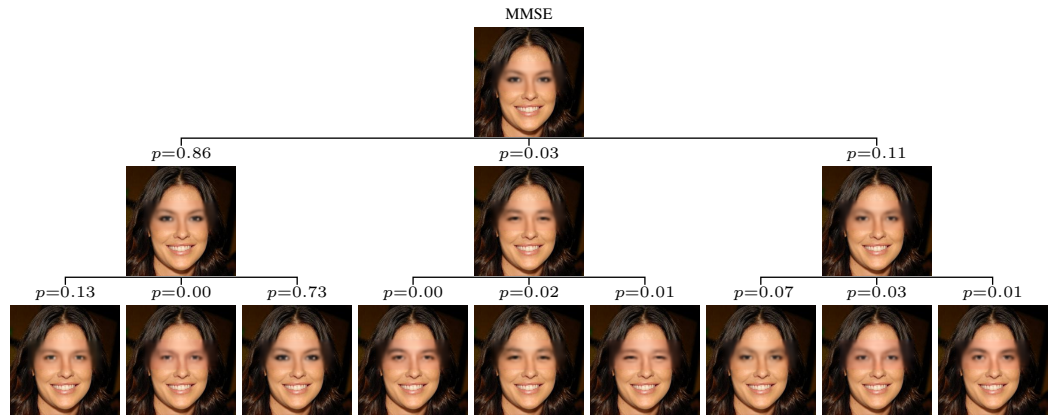
$\boldsymbol{\theta} \leftarrow \texttt{Optimizer}(\boldsymbol{\theta}, \nabla_{\boldsymbol{\theta}} \mathcal{L})$

until converged

https://doi.org/10.52202/079017-3975



(a) Pixel space training $\ell(\boldsymbol{x}, \hat{\boldsymbol{x}}) = \|\boldsymbol{x} - \hat{\boldsymbol{x}}\|_2^2$



(b) Feature space training $\ell(\boldsymbol{x}, \hat{\boldsymbol{x}}) = \|f(\boldsymbol{x}; \boldsymbol{\varphi}) - f(\hat{\boldsymbol{x}}; \boldsymbol{\varphi})\|_2^2$

Figure A5: **Different association metrics** $\ell(\cdot, \cdot)$. (a)-(b) Predicted tree in the task of eye inpainting with an MSE association loss in pixel-space/feature-space respectively.

E Centroidal Voronoi Tessellations

Consider a domain $\mathcal{X} \subseteq \mathbb{R}^d$ and let $\rho : \mathcal{X} \to \mathbb{R}_+$ be a weight function over this domain. Additionally, let $\{\boldsymbol{x}_i\}_{i=1}^K$ denote any set of K points belonging to $\mathcal{X}$ and let $\{V_i\}_{i=1}^K$ denote any tessellation of $\mathcal{X}$ into K regions. Using these notations, we can define the following loss function,

$$\mathcal{L}\left(\{(\boldsymbol{x}_i, V_i)\}_{i=1}^K\right) = \sum_{i=1}^K \int_{\boldsymbol{x}\in V_i} \rho(\boldsymbol{x})\|\boldsymbol{x} - \boldsymbol{x}_i\|^2 \mathrm{d}\boldsymbol{x}. \tag{A7}$$

This formalism arises in various applications such as data compression, optimal quantization, clustering, and other applications [66]. In all of these applications, our goal is the same; we want to minimize Eq. (A7) to obtain optimal performance.

A fundamental result in this setting is that for Eq. (A7) to be minimized, it is necessary that $\{V_i\}_{i=1}^K$ are the Voronoi regions corresponding to $\{\boldsymbol{x}_i\}_{i=1}^K$ and, simultaneously, $\{\boldsymbol{x}_i\}_{i=1}^K$ are the *centroids* of the corresponding $\{V_i\}_{i=1}^K$. Namely, $\{(\boldsymbol{x}_i, V_i)\}_{i=1}^K$ form a special tessellation of space, commonly referred to as a Centroidal Voronoi Tessellation (CVT). Specifically, it can be shown that the optimal centroids of the resulting CVT are given by

$$\boldsymbol{x}_i = \frac{\int_{\boldsymbol{x}\in V_i} \boldsymbol{x}\rho(\boldsymbol{x})\mathrm{d}\boldsymbol{x}}{\int_{\boldsymbol{x}\in V_i} \rho(\boldsymbol{x})\mathrm{d}\boldsymbol{x}}, \tag{A8}$$

where $\{V_i\}_{i=1}^K$ form a Voronoi tessellation of $\mathcal{X}$ w.r.t. $\{\boldsymbol{x}_i\}_{i=1}^K$, and are defined as

$$V_i = \{\boldsymbol{x} : \|\boldsymbol{x} - \boldsymbol{x}_i\|_2 < \|\boldsymbol{x} - \boldsymbol{x}_j\|_2, \quad \forall j \neq i\}. \tag{A9}$$

In general, the solution to Eq. (A7) is not unique, and no existing algorithm is guaranteed to converge to an optimal solution. Nevertheless, efficient algorithms like Lloyd's algorithm (Voronoi Iteration) are known to converge to a local minimum, which frequently produces satisfactory results in practice.

https://doi.org/10.52202/079017-3975

F Unconditional Hierarchical K-means

In Section 3.1 we mentioned that in a discrete setting, our approach is equivalent to a *hierarchical* K-means clustering. Here, we describe the hierarchical K-means algorithm and review its properties in the standard (unconditional) setting.

Given a set of N data points $\{\boldsymbol{x}_i\}$ where $\boldsymbol{x}_i \in \mathbb{R}^{d_x}$, the goal of K-means is to partition the data points into K clusters/sets $\{\mathcal{C}_1, \dots, \mathcal{C}_K\}$, such that each data point belongs to the cluster with the nearest mean/centroid serving as a prototype of the cluster. This results in a partitioning/tessellation of the data space into centroidal Voronoi cells, where for each cluster the within-cluster variances (squared Euclidean distances) are minimized. Formally, the objective function for finding the clusters is given by

$$\underset{\mathcal{C}_1,\dots,\mathcal{C}_K}{\arg\min} \sum_{k=1}^{K} \sum_{\boldsymbol{x}_i \in \mathcal{C}_k} \|\boldsymbol{x}_i - \boldsymbol{\mu}_k\|_2^2, \tag{A10}$$

where $\boldsymbol{\mu}_k$ is the mean/centroid of cluster $\mathcal{C}_k$, given by $\boldsymbol{\mu}_k = \frac{1}{|\mathcal{C}_k|} \sum_{\boldsymbol{x}_i \in \mathcal{C}_k} \boldsymbol{x}_i$. While in general Problem (A10) is NP-hard, given some initial means/centroids, the K-means algorithm (also known as Lloyd's method) finds a local minimum by alternating between (i) assigning each data point to the cluster with the nearest mean, and (ii) updating the cluster means based on the new assignment.

Extending K-means hierarchically is rather straightforward. Starting from a single cluster comprised of all data points $\{\boldsymbol{x}_i\}$, at each level of the hierarchy, the data points belonging to the kth cluster $\mathcal{C}_k$ are split further into K sub-clusters $\{\mathcal{C}_{k,1}, \dots, \mathcal{C}_{k,K}\}$ by applying K-means. We refer to this extended algorithm as *hierarchical* K-means. The result of applying hierarchical K-means d times is a dendrogram/top-down balanced tree $\mathcal{T}$ of degree K, depth d, and breadth (final number of leaves) K^d. Note that this successive process is different from applying K-means with K^d clusters once. This is because the hierarchical memberships are enforced, restricting data points in subclusters $\{\mathcal{C}_{i,1}, \dots, \mathcal{C}_{i,K}\}$ from being assigned to subclusters $\{\mathcal{C}_{j,1}, \dots, \mathcal{C}_{j,K}\}$ if $i \neq j$.

Hierarchical K-means enjoys several advantages compared to the standard ("Flat") K-means algorithm. One of these advantages is increased robustness to initialization as demonstrated in Appendix G.3.

G Toy Example

G.1 Model Architecture

The model used to produce our results in Fig. 3 consisted of two 5-layer MLPs with 256 hidden units, one for predicting the leaves and one for predicting the likelihoods. Both MLPs consisted of linear layers interleaved with the SiLU activation. In addition, the output of the second MLP was passed through a $\mathrm{softmax}$ to represent a valid probability distribution.

G.2 Analytical Posterior

In Fig. 3, we plotted the posterior trees on the (unknown) ground truth posterior. Here we provide the closed-form expression for the analytically derived posterior for completeness. The denoising task we assumed was $\mathbf{y} = \mathbf{x} + \mathbf{n}$, where $\mathbf{x}$ comes from a mixture of $L = 4$ Gaussians, $p_{\mathbf{x}}(\boldsymbol{x}) = \sum_{\ell=1}^{L} \pi_\ell \mathcal{N}(\boldsymbol{x}; \boldsymbol{\mu}_\ell, \boldsymbol{\Sigma}_\ell)$, and $\mathbf{n} \sim \mathcal{N}(\cdot; \mathbf{0}, \sigma_\varepsilon^2 \boldsymbol{I})$ is a white Gaussian noise. Specifically, in our toy example we used equally probable spherical Gaussians (*i.e.,* $\pi_\ell = \frac{1}{4}, \boldsymbol{\Sigma}_\ell = \boldsymbol{I}$), with the following means

$$\boldsymbol{\mu}_1 = \begin{pmatrix} -6.0 \\ +2.5 \end{pmatrix}, \ \boldsymbol{\mu}_2 = \begin{pmatrix} +1.0 \\ +2.5 \end{pmatrix}, \ \boldsymbol{\mu}_3 = \begin{pmatrix} -2.5 \\ +6.0 \end{pmatrix}, \ \boldsymbol{\mu}_4 = \begin{pmatrix} -2.5 \\ -1.5 \end{pmatrix}. \tag{A11}$$

Let c be an auxiliary random variable taking values in $\{1, \dots, L\}$ with probabilities $\{\pi_1, \dots, \pi_L\}$. Then we can write the posterior by invoking the law of total probability conditioned on the event $\mathrm{c} = \ell$,

$$\begin{aligned} p(\boldsymbol{x}|\boldsymbol{y}) &= \sum_{\ell=1}^{L} p_{\mathbf{x}|\mathbf{y},\mathrm{c}}(\boldsymbol{x}|\boldsymbol{y}, \ell) p_{\mathrm{c}|\mathbf{y}}(\ell|\boldsymbol{y}) \\ &= \sum_{\ell=1}^{L} p_{\mathbf{x}|\mathbf{y},\mathrm{c}}(\boldsymbol{x}|\boldsymbol{y}, \ell) \frac{p_{\mathbf{y}|\mathrm{c}}(\boldsymbol{y}|\ell) p_{\mathrm{c}}(\ell)}{p_{\mathbf{y}}(\boldsymbol{y})} \\ &= \sum_{\ell=1}^{L} \mathcal{N}(\boldsymbol{x}; \tilde{\boldsymbol{\mu}}_\ell, \tilde{\boldsymbol{\Sigma}}_\ell) \frac{q_\ell \pi_\ell}{\sum_{\ell'=1}^{L} q_{\ell'} \pi_{\ell'}}, \end{aligned} \tag{A12}$$

where the first step is by Bayes rule, and in the result we denoted

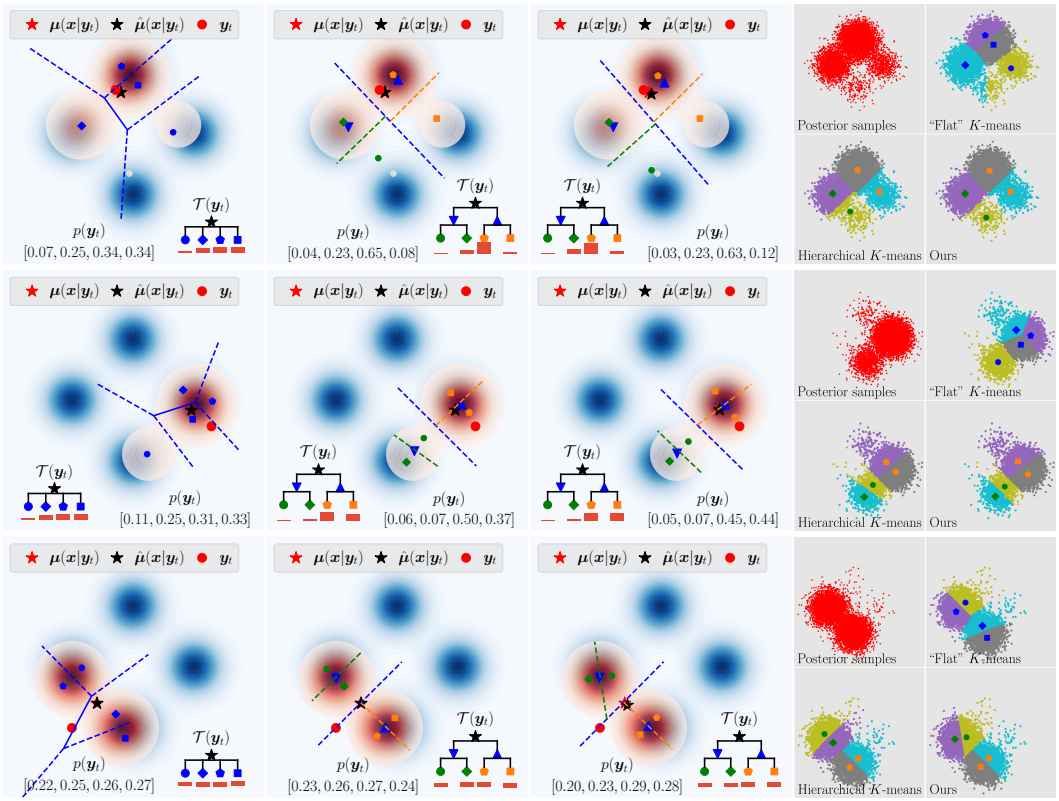
$$\begin{aligned} q_\ell &= \mathcal{N}(\boldsymbol{y}; \boldsymbol{\mu}_\ell, \boldsymbol{\Sigma}_\ell + \sigma_\varepsilon^2 \boldsymbol{I}), \\ \tilde{\boldsymbol{\mu}}_\ell &= \boldsymbol{\mu}_\ell + \boldsymbol{\Sigma}_\ell (\boldsymbol{\Sigma}_\ell + \sigma_\varepsilon^2 \boldsymbol{I})^{-1} (\boldsymbol{y} - \boldsymbol{\mu}_\ell) \\ \tilde{\boldsymbol{\Sigma}}_\ell &= \boldsymbol{\Sigma}_\ell - \boldsymbol{\Sigma}_\ell (\boldsymbol{\Sigma}_\ell + \sigma_\varepsilon^2 \boldsymbol{I})^{-1} \boldsymbol{\Sigma}_\ell, \quad \ell = 1, \dots, L. \end{aligned} \tag{A13}$$

In Fig. A6, in a similar fashion to Fig. 3 from the main text, we visualize posterior trees for additional test inputs $\boldsymbol{y}_t$. As clearly evident in all cases, our method recovers the (approximated) ground truth posterior tree with high accuracy.

G.3 Stability of "Flat" vs. Hierarchical K-means

As mentioned earlier, the hierarchical K-means algorithm has several advantages over the classical "Flat" K-means algorithm. Here, we examine the effect of the random initialization on the resulting clusters found by "Flat"/Hierarchical K-means. In both cases, we applied the respective method to 10K posterior samples to avoid errors resulting from an insufficient sample size. Figure A7 demonstrates the resulting clusters for 3 different seeds. As can be seen by the result, despite using the widely adopted K-means++ initialization for both algorithms, "Flat" K-means was less resilient to a bad initialization compared to its hierarchical counterpart. This is yet another advantage of working with trees of depth bigger than $d = 1$.

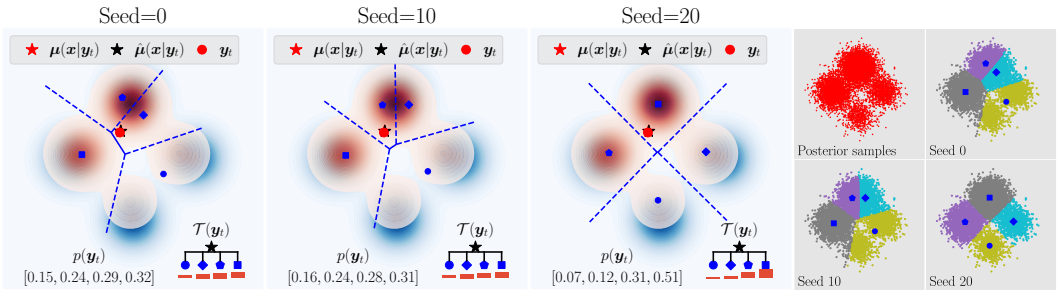
https://doi.org/10.52202/079017-3975



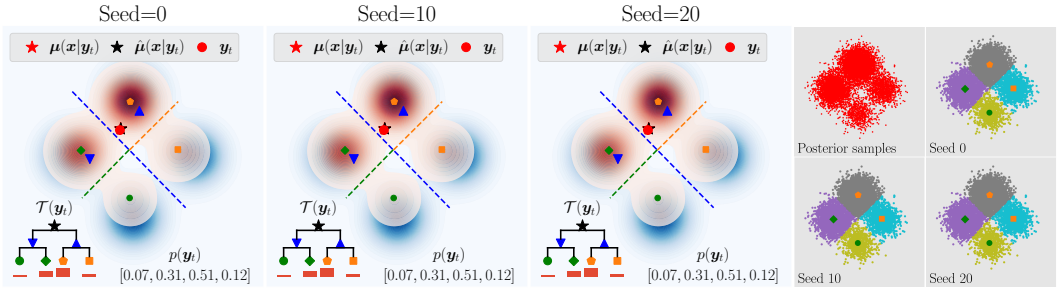
(a) "Flat" K-means (b) Hier. K-means (c) Ours (d) Clusters

Figure A6: **Additional test points y_t.** Each row above shows the results for a different test point y_t (red dot) when clustering with (a) "Flat" K-means, (b) Hierarchical K-means, (c) Posterior trees (ours). In (d) we show the resulting partition/clustering induced by each method by coloring 10K samples from the posterior $p_{\mathbf{x}|\mathbf{y}}(x|y_t)$ according to their nearest cluster. Our method consistently recovers the result of hierarchical K-means in all cases. Moreover, other than trivial cases such as the bottom row, the hierarchy better represents weaker modes compared to "Flat" K-means.

https://doi.org/10.52202/079017-3975



(a) "Flat" K-means



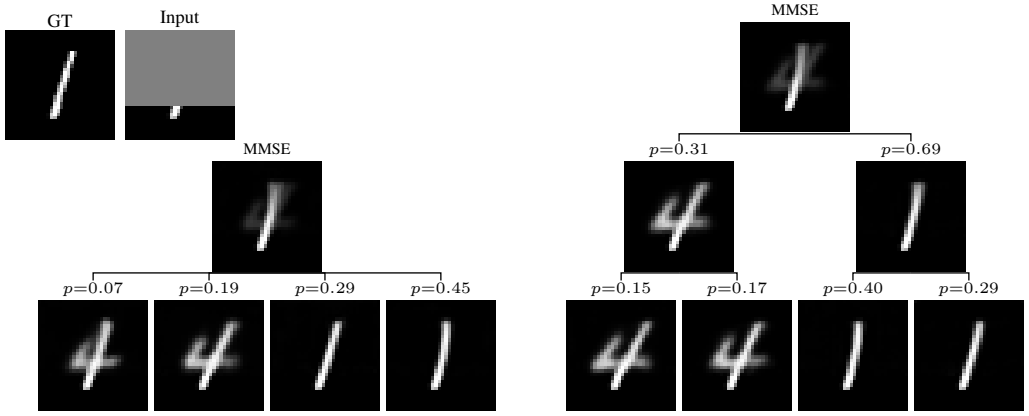
(b) Hierarchical K-means

Figure A7: **Resilience to initialization**. Here we inspect the effect of the seed used for initialization, on the resulting clusters recovered by (a) "Flat" K-means and (b) Hierarchical K-means. The hierarchy results in a more resilient algorithm that recovers the same result regardless of the seed used for initialization.

https://doi.org/10.52202/079017-3975

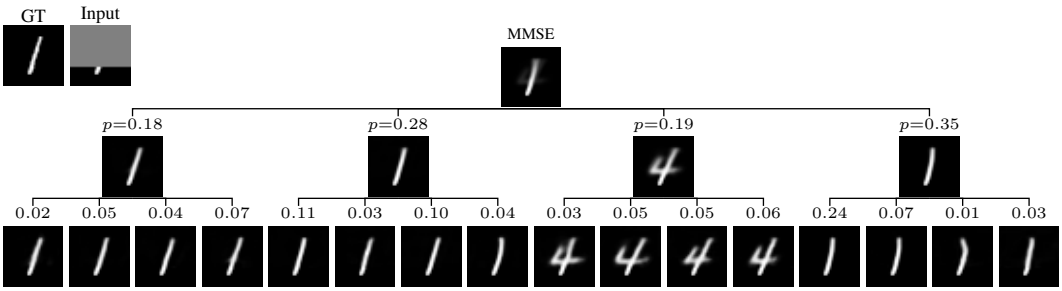
H Tree Width vs. Depth

Throughout Section 4 we presented results with trees of degree $K = 3$ and depth $d = 2$. This was done only for the convenience of the exposition and our method is not restricted to this specific layout. Here we examine the effect of tree width compared to tree depth. Figures A8 and A9 presents the resulting trees for four different model configurations on the task of digit inpainting. For a fair comparison between model pairs, we fix the number of output leaves which dictates the number of model parameters, and only change the hierarchical structure. Figure A8(a) compares a layout of $K = 4, d = 1$ to $K = 2, d = 2$. As evident in the result, the hierarchy better organizes the different options, presenting the "4" cluster more faithfully than the flat tree which presents a cluster midway between a "4" and a "1". Figure A9 repeats this experiment for deeper trees comparing $K = 4, d = 2$ to $K = 2, d = 4$. A couple of remarks are in order regarding the results. First, in both cases, the "4" cluster has a similar probability relative to the "1" cluster. This indicates that our method is consistent in the predicted posterior modes and their likelihoods, with the only change between different layouts being the chosen tessellation of the output space. Second, the degree K controls the emphasis/over-representation devoted to weaker posterior modes. A smaller degree leads to more emphasis on weaker modes as tree depth grows. Lastly, it is important to note that the optimal layout K and d is task and input-dependant, and setting these adaptively is an interesting direction for future research.

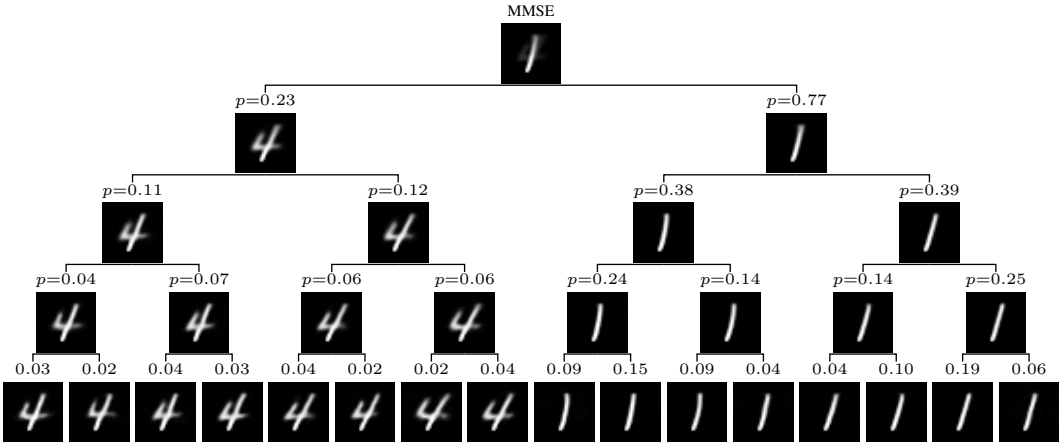


(a) Tree with $K = 4, d = 1$ (b) Tree with $K = 2, d = 2$

Figure A8: **Flat vs. hierarchical trees.** (a) Flat tree with $K = 4, d = 1$. (b) Binary tree with $K = 2, d = 2$, with overall 4 leaves. The binary tree categorizes the leaves and better represents the "4" mode.



(a) Tree with $K = 4, d = 2$



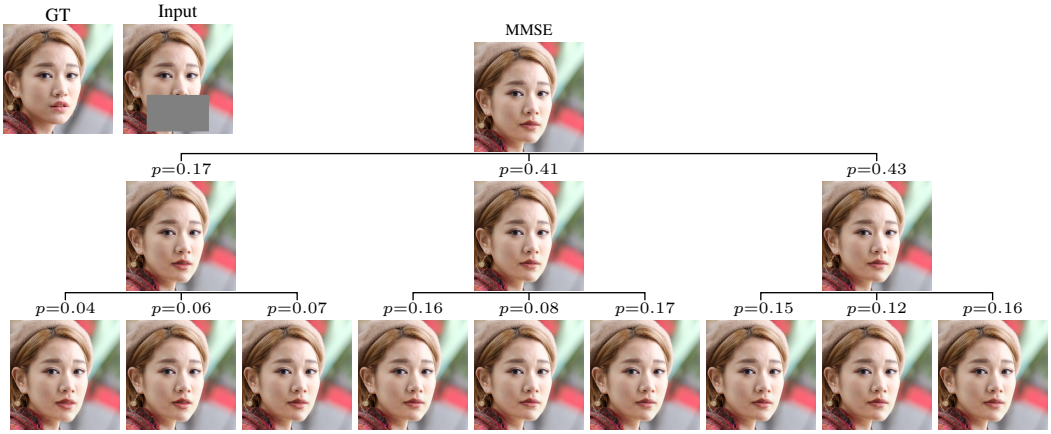
(b) Tree with $K = 2, d = 4$

Figure A9: **Tree depth vs. width.** (a) Tree with $K = 4, d = 2$. (b) Binary tree with $K = 2, d = 4$ also resulting in 16 leaves. The binary tree emphasizes more the "4" mode relative to the "1" mode, although in both cases the probability mass associated with the "4" is $\approx 20\%$.

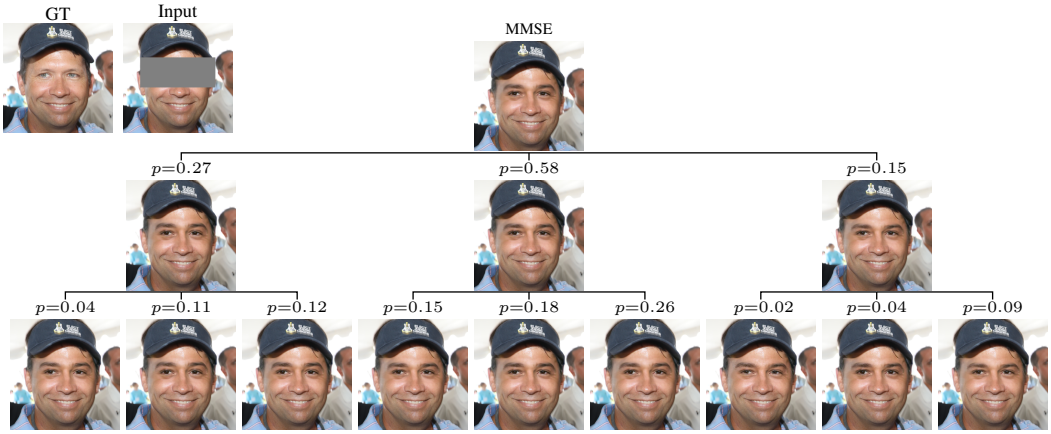
https://doi.org/10.52202/079017-3975

I GAN-based Posterior Samplers

In Section 4 we compared our method mainly to diffusion-based posterior samplers (*e.g.,* [7, 8, 11]). While these samplers often lead to state-of-the-art sample quality, they are known to be computationally intensive. In theory, GAN-based samplers such as MAT [10], can significantly speed up the proposed two-step baseline of sampling followed by hierarchical K-means; however, as evident in Fig. A10, even top-performing GAN-based samplers often collapse to a single mode of the posterior, producing samples with little variability that do not faithfully reflect the full distribution.



(a) Mouth inpainting



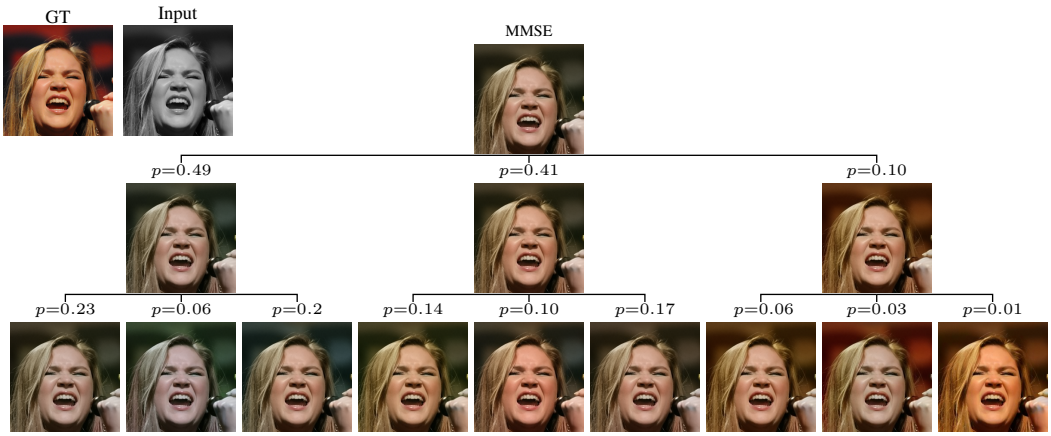
(b) Eyes inpainting

Figure A10: **GAN-based posterior sampler.** (a)-(b) Posterior trees constructed with MAT, by applying hierarchical K-means to 100 samples in the tasks of mouth/eyes inpainting respectively. As evident in both cases, the resulting tree exhibits little to no variance due to mode collapse.

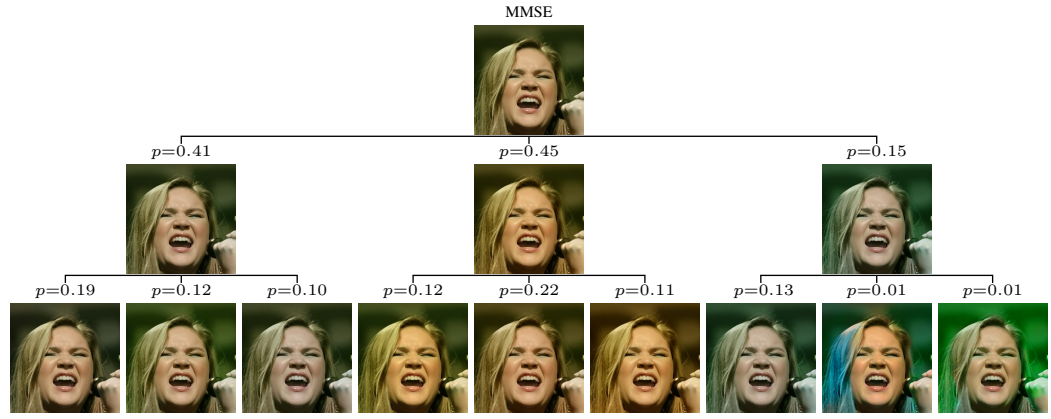
https://doi.org/10.52202/079017-3975

J Visual Comparison to Baselines

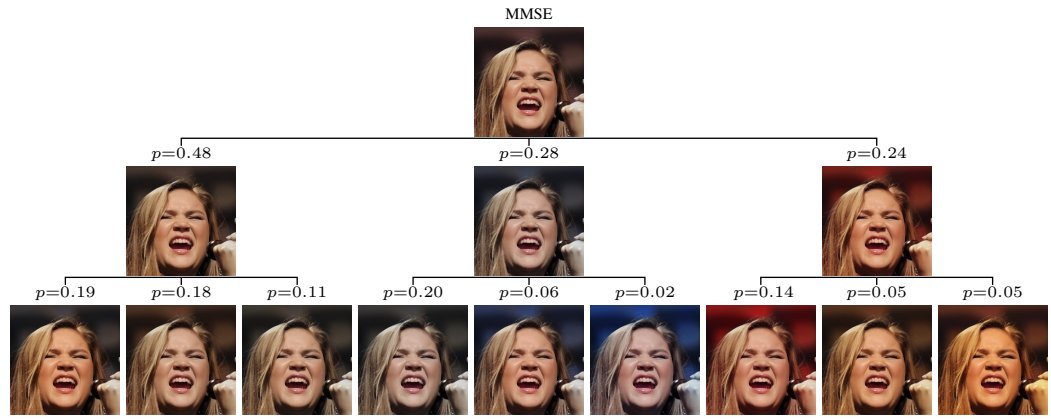
In Table 1 we reported quantitative comparisons to the proposed baseline implemented with various samplers. In Figs. A11 and A12, we include two sample visual comparisons of the resulting trees with our method and the baselines for the tasks of face colorization and mouth inpainting.



(a) DDNM



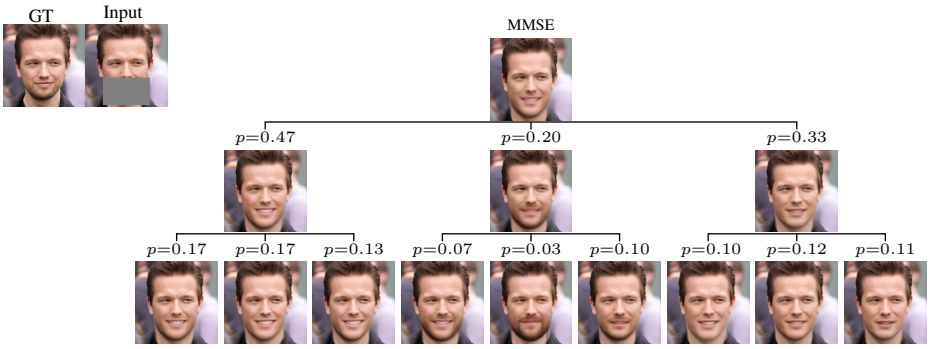
(b) DDRM



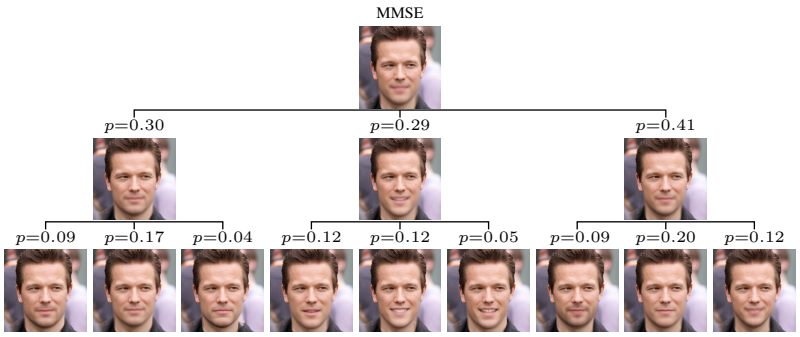
(c) Ours

Figure A11: **Tree comparison in colorization.**

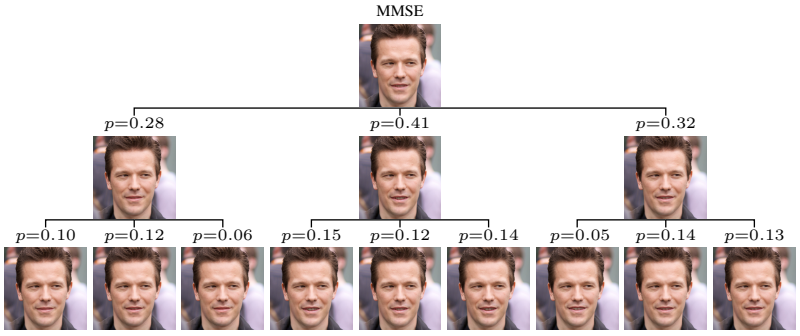
https://doi.org/10.52202/079017-3975



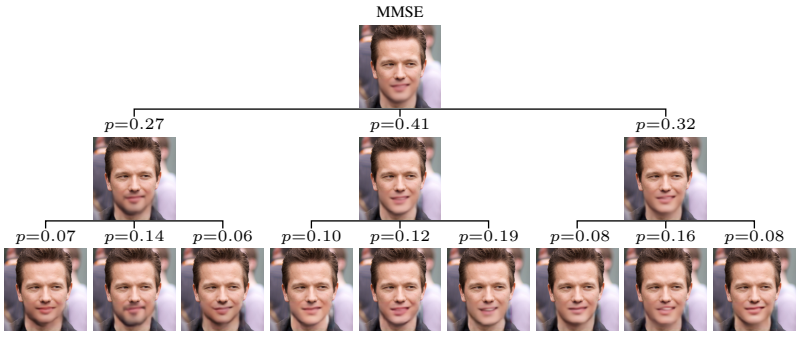
(a) DDNM



(b) RePaint



(c) MAT



(d) Ours

Figure A12: **Tree comparison in mouth inpainting.**

K Runtime Comparison as a Function of Batch Size

In Table 1 we reported the runtime and memory footprint assuming a batch of size 1. However, in GPU computing, memory footprint and hardware parallelization ultimately determine the overall runtime. We tried to refrain from factoring these in our calculation in Table 1 as these numbers are hardware-specific depending on the available GPU. Nonetheless, it is important to note that our method is not only faster than the competing approaches with a batch of size 1, it also has a lower memory footprint and is therefore more amenable to parallelization. Table 2 benchmarks the speed of the forward pass and the GPU memory usage as a function of the batch size on an A6000 GPU with 48 GB. For each method, we tested three different batch sizes: 1, 10, and the maximal number of samples that fit in memory. The forward pass speed and the memory usage in each setting were tested 100 times using CUDA events (reported as mean $\pm$ std). Our method is extremely fast and parallelizable, enabling the inference of 430 test images with a single forward pass of 1.15 seconds.

Even in the case of a **single** test image, our method is far superior. For simplicity let us assume the cost of running hierarchical K-means is negligible, and that we can squeeze a batch size of ≈ 100 samples for the diffusion-based samplers, and $\approx$"33.33" samples for MAT. Sampling $N_s = 100$ posterior samples with DDRM (fastest diffusion baseline, requires only 20 denoising steps) lasts 46.3 seconds, and with MAT lasts 7.8 seconds. In comparison, our method requires only 0.014 seconds, leading to a $557\times$ speedup compared to the fastest baseline.

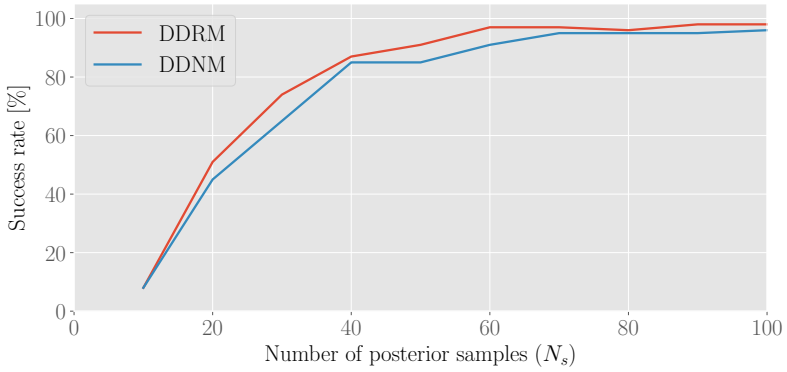
Table 2: Runtime comparison as a function of batch size on an Nvidia A6000 GPU. Note that unlike Table 1, here we report the speed of a forward pass separately from the number of required forward passes to infer a posterior tree for a single test image.

Method	Batch Size	Forward pass speed (ms)	Memory (GB)	Forward passes per tree	Total time per tree (sec)
Diffusion	1	34±0.4	1.85	2,000-456,000	68-15,504
	10	24.5±0.1	7.5		49-11,172
	97	23.9±0.6	46.8		48-10,898
MAT	1	150±0.8	3.0	100	15
	10	86.7±1.2	16.0		8.67
	30	86.6±2.5	47.2		8.66
Ours	1	14±0.2	1.3	1	0.014
	10	3.9±0.1	2.1		0.0039
	430	2.67±0.1	46.9		0.00267

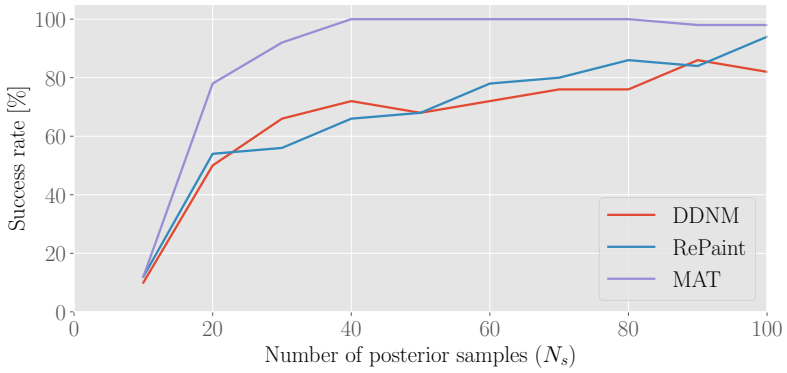
https://doi.org/10.52202/079017-3975

L Baselines Performance as a Function of Compute

As mentioned in Section 4.1, baseline trees were computed using a two-step procedure: (1) Sampling N_s images from the posterior, and (2) performing hierarchical K-means d times to build a tree of degree K and depth d. Therefore, saving computation in this process requires sampling fewer images N_s per test input. However, in our experiments with $K = 3, d = 2$ (i.e. a tree with 9 leaves), we noticed that using less than $N_s = 100$ images often led to degenerate trees with one or more leaves having 1 sample or less. For example, consider the case of using $N_s = 9$ samples. Even if the posterior is perfectly balanced (often not the case), each leaf will have only 1 sample to "average" at depth 2. Figure A13 plots the success probability of building a tree with $K^d = 9$ leaves as a function of the number of sampled images N_s in the different tested tasks. A tree is considered to be successfully constructed if each of its leaves has at least 2 samples to average. As evident in the results (*e.g.,* Fig. A13(b)), it is likely that to achieve the optimal performance with the baselines more than $N_s = 100$ samples are needed, which would be even more computationally demanding. Moreover, the number of samples needed to successfully construct trees with more than 9 leaves is expected to be significantly larger rendering this baseline impractical, especially when considering a sizable test set.



(a) Colorization



(b) Eyes inpainting

Figure A13: **Tree success rate vs. number of posterior samples N_s.** (a)-(b) Success rate of building a (posterior) tree by applying hierarchical K-means to N_s samples in the tasks of colorization and eyes inpainting respectively.

M More Results

Here we include more results for each of the tasks presented in Section 4 from the main text. Additionally, we also include colorization results on the AFHQ(v2) dataset [67], with a similar performance to that obtained on CelebA-HQ.

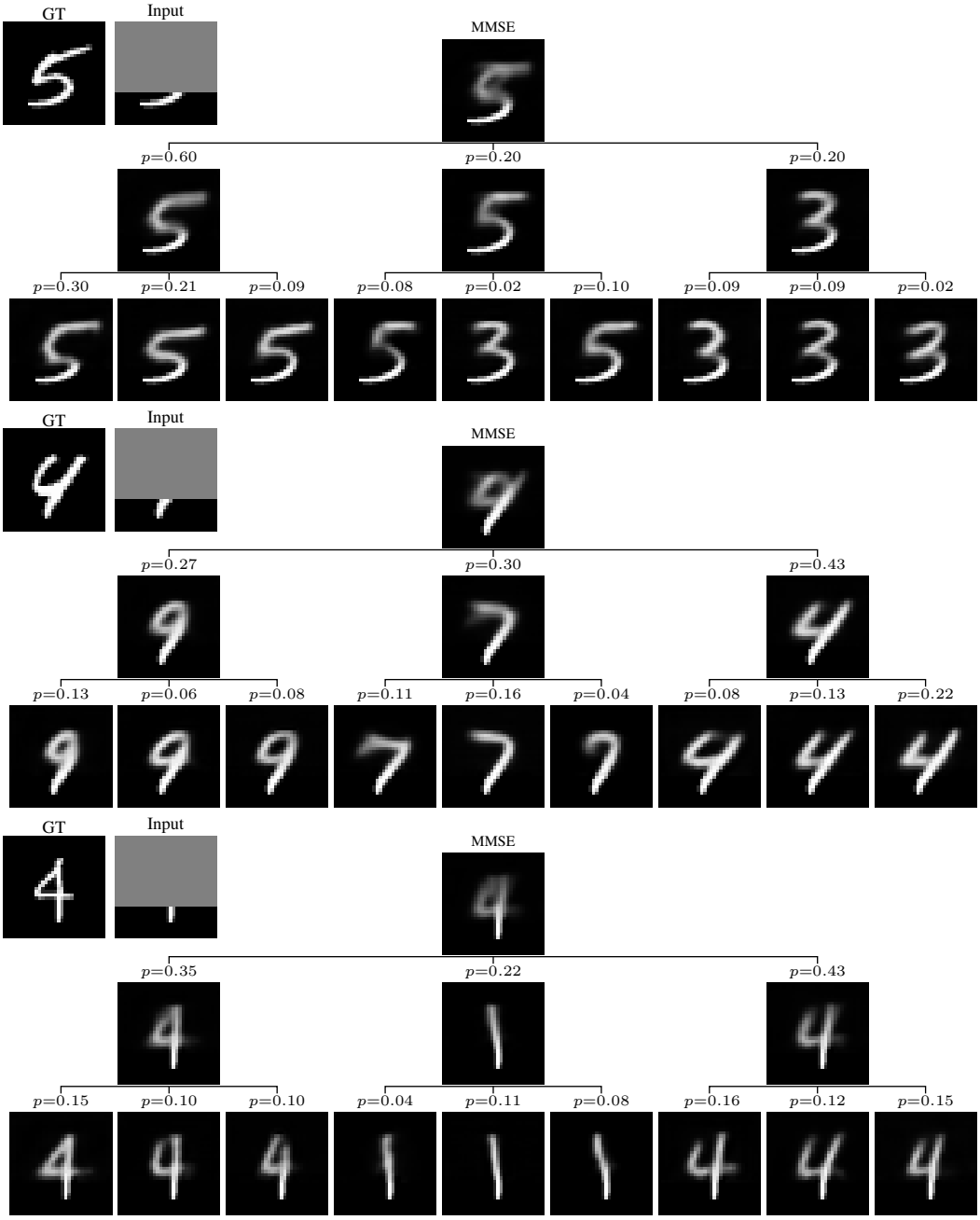


Figure A14: **More digit inpainting results.**

https://doi.org/10.52202/079017-3975

Figure A15: **More Edges→Shoes results.**

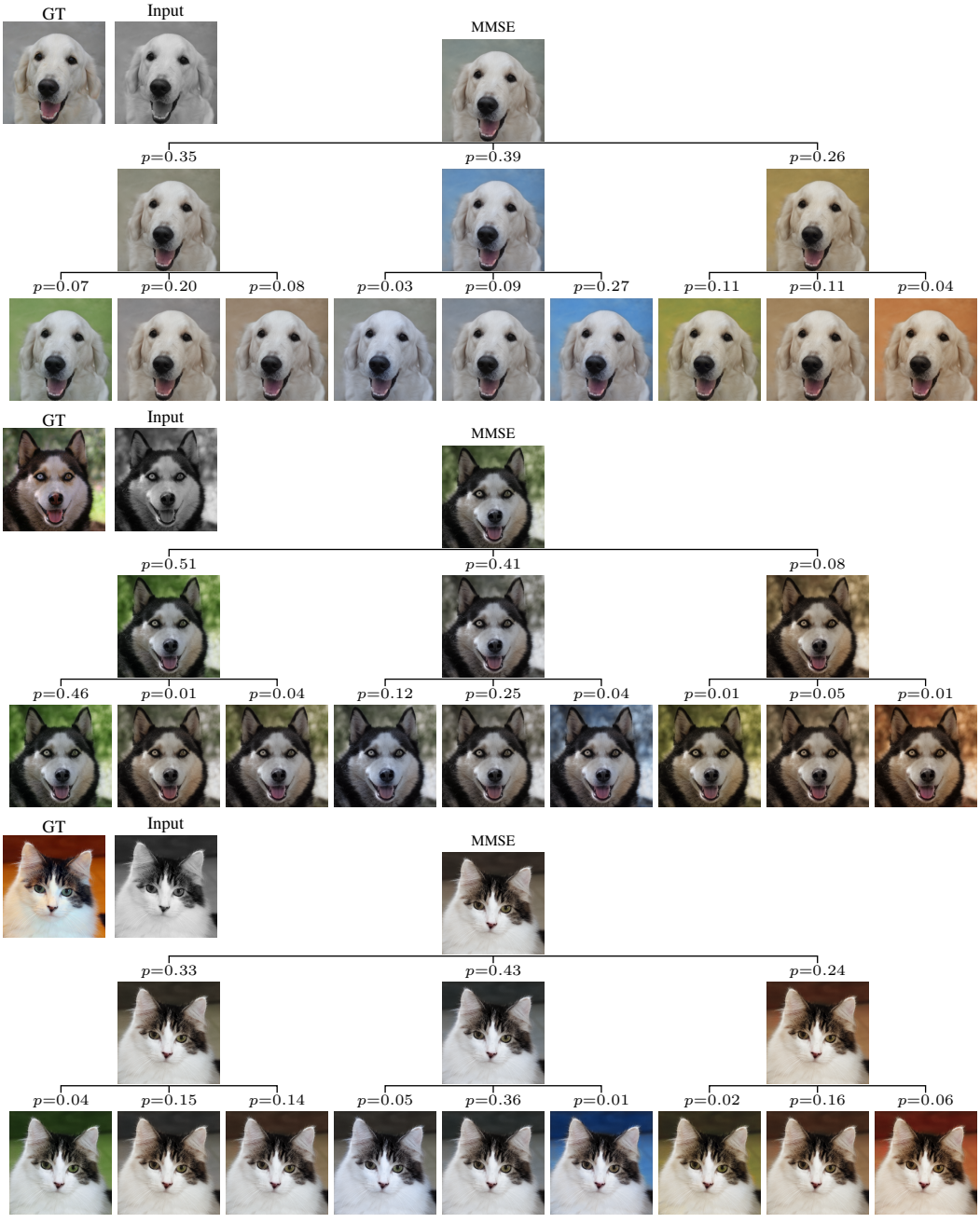


Figure A16: **AFHQ(v2) colorization results.**

https://doi.org/10.52202/079017-3975

Figure A17: **More CelebA-HQ colorization results.**

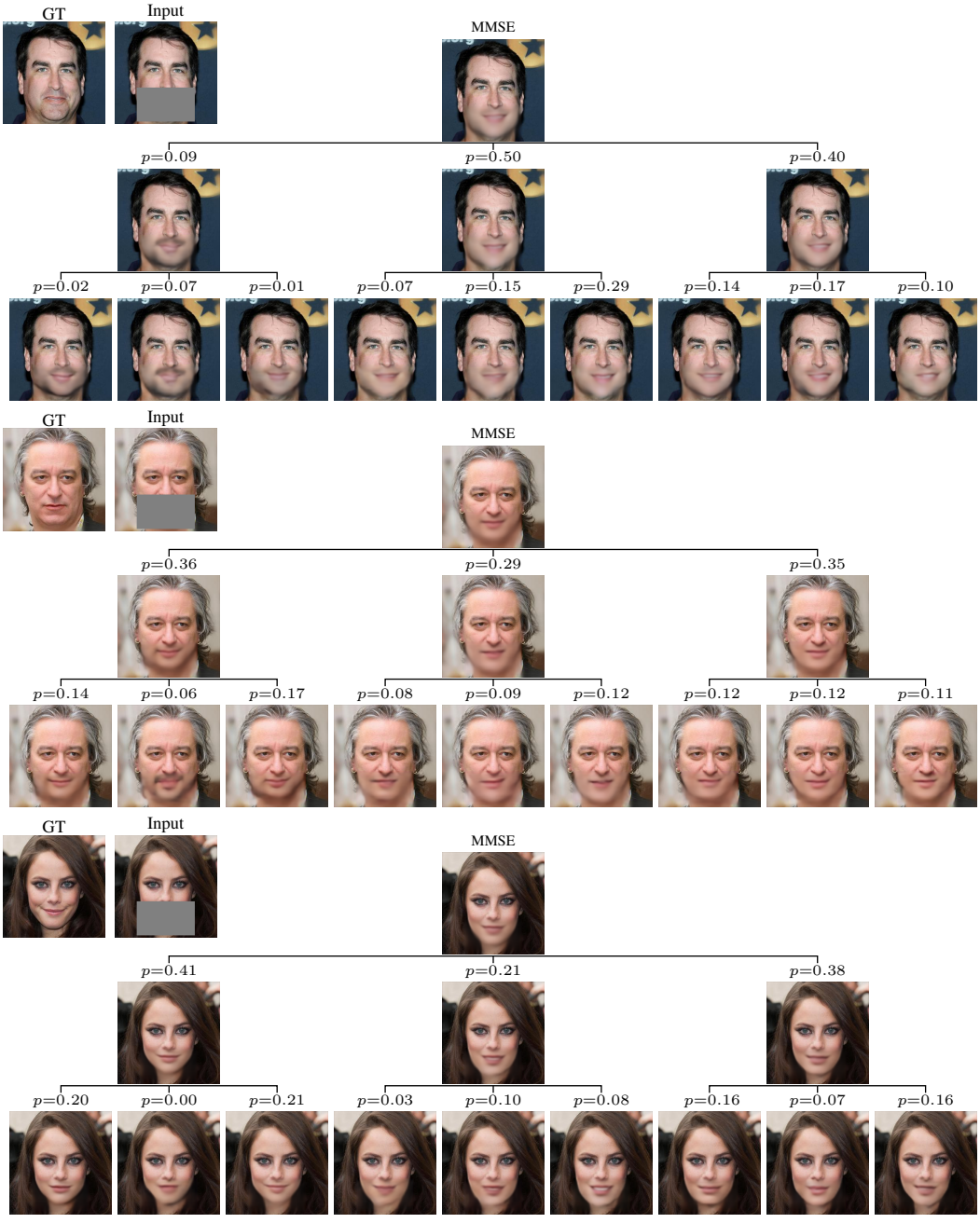


Figure A18: **More mouth inpainting results.**

https://doi.org/10.52202/079017-3975

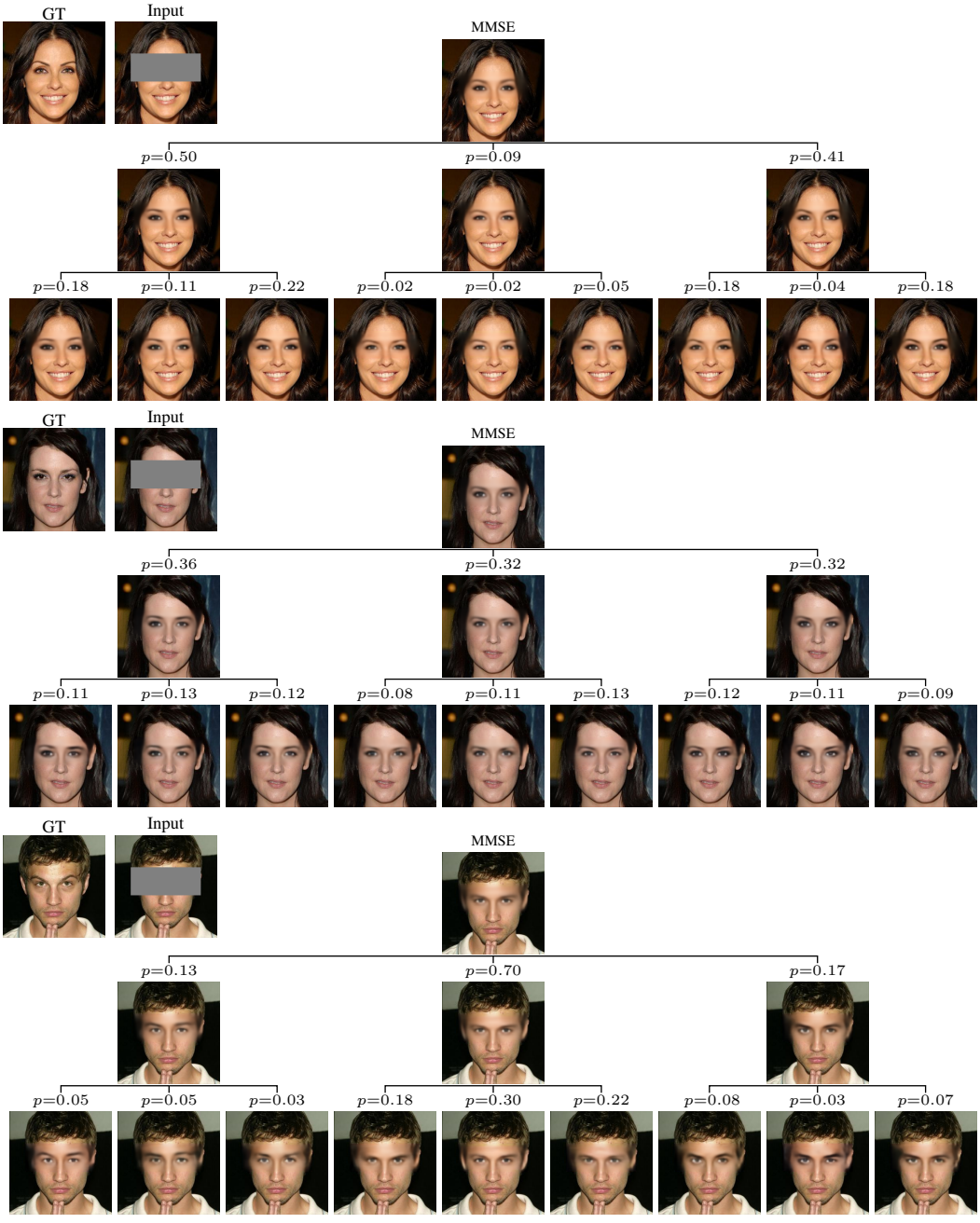


Figure A19: **More eyes inpainting results.**

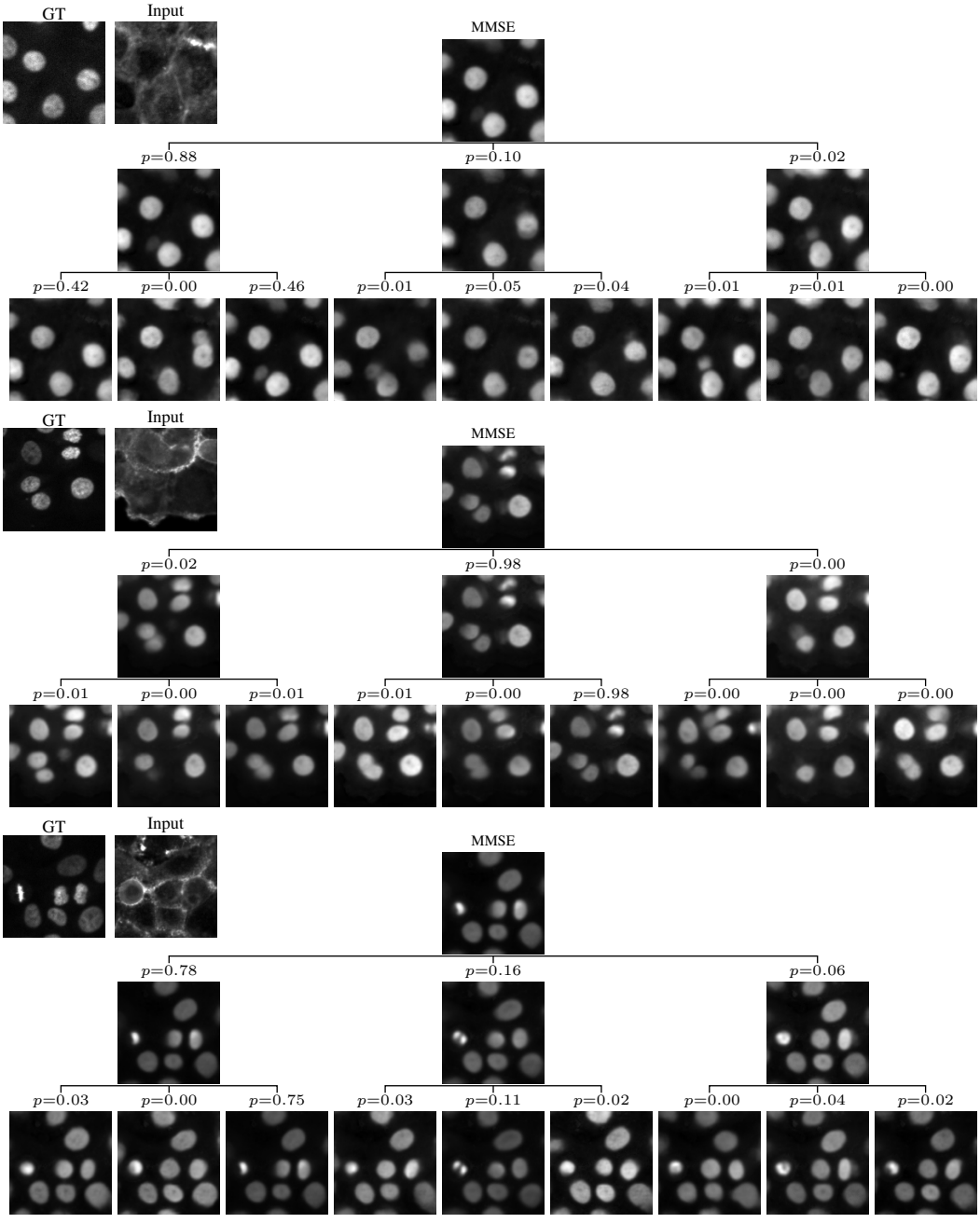


Figure A20: **More Bioimage translation results.**

https://doi.org/10.52202/079017-3975

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The claims made in the abstract and introduction are reflected in our theoretical and experimental results found in Sections 3 and 4.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: We discuss the limitations of our method in Section 5.

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory Assumptions and Proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes]

Justification: Our results in Section 3 are followed by a proper theoretical justification where applicable.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental Result Reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: Both the main text (Section 3) and the appendix (Appendices A, C and G) provide all essential details to replicate our experimental results.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

https://doi.org/10.52202/079017-3975

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: All datasets used in the paper are publicly available and can be retrieved from the cited papers. Our code is made publicly available on GitHub.

Guidelines:

- The answer NA means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (https://nips.cc/public/guides/CodeSubmissionPolicy) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (https://nips.cc/public/guides/CodeSubmissionPolicy) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental Setting/Details**

Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

Answer: [Yes]

Justification: Our settings are discussed in detail in the main text (Section 4) and in Appendix A.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
- The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment Statistical Significance**

Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

Answer: [Yes]

Justification: In Table 1 we report the standard error of the mean over the test images for both MSE and NLL.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.

- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments Compute Resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: The per-task training times are detailed in Appendix A.3. In terms of inference times and memory usage, we discussed these extensively in Tables 1 and 2 and Appendices K and L.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code Of Ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics https://neurips.cc/public/EthicsGuidelines?

   Answer: [Yes]

   Justification: the paper fully conforms with the NeurIPS Code of Ethics.

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader Impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: In Section 1 we discussed the potential positive societal impact of our method, and in Section 4 we demonstrated its practical use for properly visualizing reconstruction uncertainty in safety-critical tasks such as bioimage translation.

https://doi.org/10.52202/079017-3975

Guidelines:

- The answer NA means that there is no societal impact of the work performed.
- If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [NA]

    Justification: This paper poses no such risks.

    Guidelines:

    - The answer NA means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: All dataset papers are properly cited. Licenses are as follow: MNIST (CC BY-SA 3.0), Edges→Shoes (CC BY-NC 4), AFHQ(v2) (CC BY-NC 4), CelebA-HQ (CC BY-NC 4), Bioimage Translation (CC BY 4).

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.

- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, paperswithcode.com/datasets has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New Assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [NA]

    Justification: The paper does not release new assets.

    Guidelines:

    - The answer NA means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and Research with Human Subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [NA]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional Review Board (IRB) Approvals or Equivalent for Research with Human Subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.

https://doi.org/10.52202/079017-3975

- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.