Conditional Generative Models is Gaussian Kernel Regression (under certain conditions)

Haojun Qiu

1. The Equivalence

The full statement

- When a conditional generative model is train with gaussian noise augmenting the conditioning input, and towards a memorization stage, then its inference on new conditioning input is equivalent to Gaussian kernel regression.

1. The Equivalence

How to derive it?

Assume the generative model is trained to the memorization stage, and the training is done with noise augmenting the conditions by $\mathcal{N}(\mathbf{0}, \sigma_{\text{aug}}\mathbf{I})$. Then at inference given a test $\mathbf{x}'_{\text{src}}$, the prob. of sample/retrieve $\mathbf{x}^{(i)}_{\text{dst}}$ in dataset $\mathcal{D}$ is

$$\begin{aligned}
\Pr[\mathbf{x}_{\text{dst}} = {\mathbf{x}_{\text{dst}}}^{(i)} \,|\, \mathbf{x}_{\text{src}} = {\mathbf{x}_{\text{src}}}'] &= \frac{\Pr[\mathbf{x}_{\text{src}} = {\mathbf{x}_{\text{src}}}' \,|\, \mathbf{x}_{\text{dst}} = {\mathbf{x}_{\text{dst}}}^{(i)}] \Pr[\mathbf{x}_{\text{dst}} = {\mathbf{x}_{\text{dst}}}^{(i)}]}{\Pr[\mathbf{x}_{\text{src}} = {\mathbf{x}_{\text{src}}}']} \\
&= \frac{\Pr[\mathbf{x}_{\text{src}} = {\mathbf{x}_{\text{src}}}' \,|\, \mathbf{x}_{\text{dst}} = {\mathbf{x}_{\text{dst}}}^{(i)}] \Pr[\mathbf{x}_{\text{dst}} = {\mathbf{x}_{\text{dst}}}^{(i)}]}{\sum_{j=1}^{N} \Pr[\mathbf{x}_{\text{src}} = {\mathbf{x}_{\text{src}}}' \,|\, \mathbf{x}_{\text{dst}} = {\mathbf{x}_{\text{dst}}}^{(j)}] \Pr[\mathbf{x}_{\text{dst}} = {\mathbf{x}_{\text{dst}}}^{(j)}]} \\
&= \frac{\Pr[\mathbf{x}_{\text{src}} = {\mathbf{x}_{\text{src}}}' \,|\, \mathbf{x}_{\text{dst}} = {\mathbf{x}_{\text{dst}}}^{(i)}](1/N)}{\sum_{j=1}^{N} \Pr[\mathbf{x}_{\text{src}} = {\mathbf{x}_{\text{src}}}' \,|\, \mathbf{x}_{\text{dst}} = {\mathbf{x}_{\text{dst}}}^{(j)}](1/N)} \\
&= \frac{\Pr[\mathbf{x}_{\text{src}} = {\mathbf{x}_{\text{src}}}' \,|\, \mathbf{x}_{\text{dst}} = {\mathbf{x}_{\text{dst}}}^{(i)}]}{\sum_{j=1}^{N} \Pr[\mathbf{x}_{\text{src}} = {\mathbf{x}_{\text{src}}}' \,|\, \mathbf{x}_{\text{dst}} = {\mathbf{x}_{\text{dst}}}^{(j)}]} \\
&= \frac{\mathcal{N}\left({\mathbf{x}_{\text{src}}}'; {\mathbf{x}_{\text{src}}}^{(i)}, {\sigma_{\text{aug}}}^2\mathbf{I}\right)}{\sum_{j=1}^{N} \mathcal{N}\left({\mathbf{x}_{\text{src}}}'; {\mathbf{x}_{\text{src}}}^{(j)}, {\sigma_{\text{aug}}}^2\mathbf{I}\right)} \\
&= \frac{\exp\left(-\frac{1}{2{\sigma_{\text{aug}}}^2} \left\| {\mathbf{x}_{\text{src}}}' - {\mathbf{x}_{\text{src}}}^{(i)} \right\|_2^2\right)}{\sum_{j=1}^{N} \exp\left(-\frac{1}{2{\sigma_{\text{aug}}}^2} \left\| {\mathbf{x}_{\text{src}}}' - {\mathbf{x}_{\text{src}}}^{(j)} \right\|_2^2\right)}.
\end{aligned}$$

1. The Equivalence

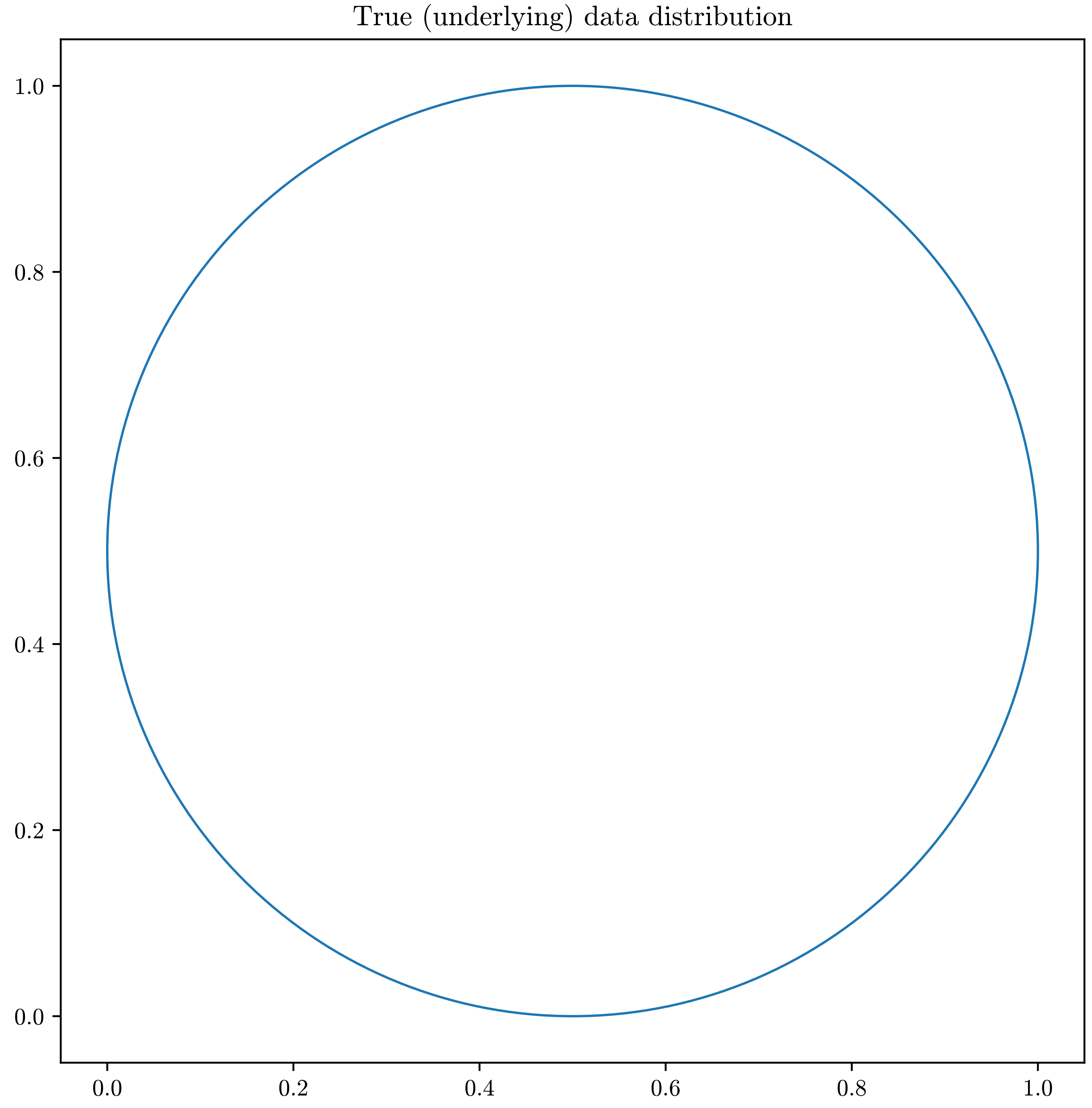
To start — what is memorization?

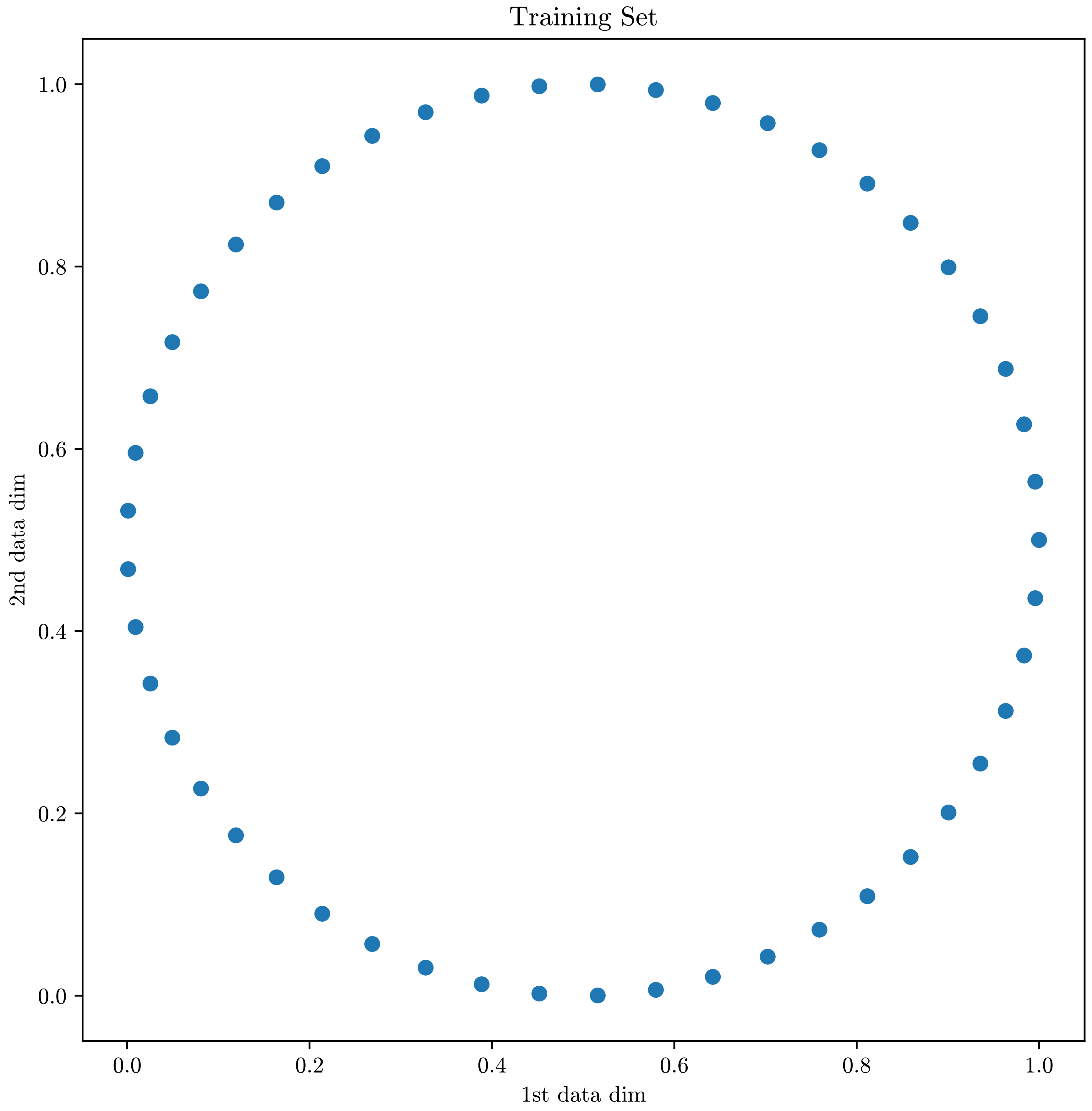
- For common training context, only observe empirical distribution (mixture of N dirac deltas samples) instead of true underlying distribution.
- So, expressive diffusion model is analytically able to generate from empirical data only — that reduces to data retrieval instead of what's desired (novel generation)
- This should hold true for both unconditional/conditional
- Some even says — diffusion model generate novel data by approximation error (e.g., generative if not overfit)

1. The Equivalence

Memorization in toy example

$$p_{\text{true}}(\mathbf{x}) \qquad \mathcal{D} = \{\mathbf{x}^{(i)}\}_{i=1}^{N} \sim p_{\text{true}}(\mathbf{x}) \qquad p_{\text{data}}(\mathbf{x}) = \frac{1}{N}\sum_{i=1}^{N}\delta(\mathbf{x}-\mathbf{x}^{(i)})$$

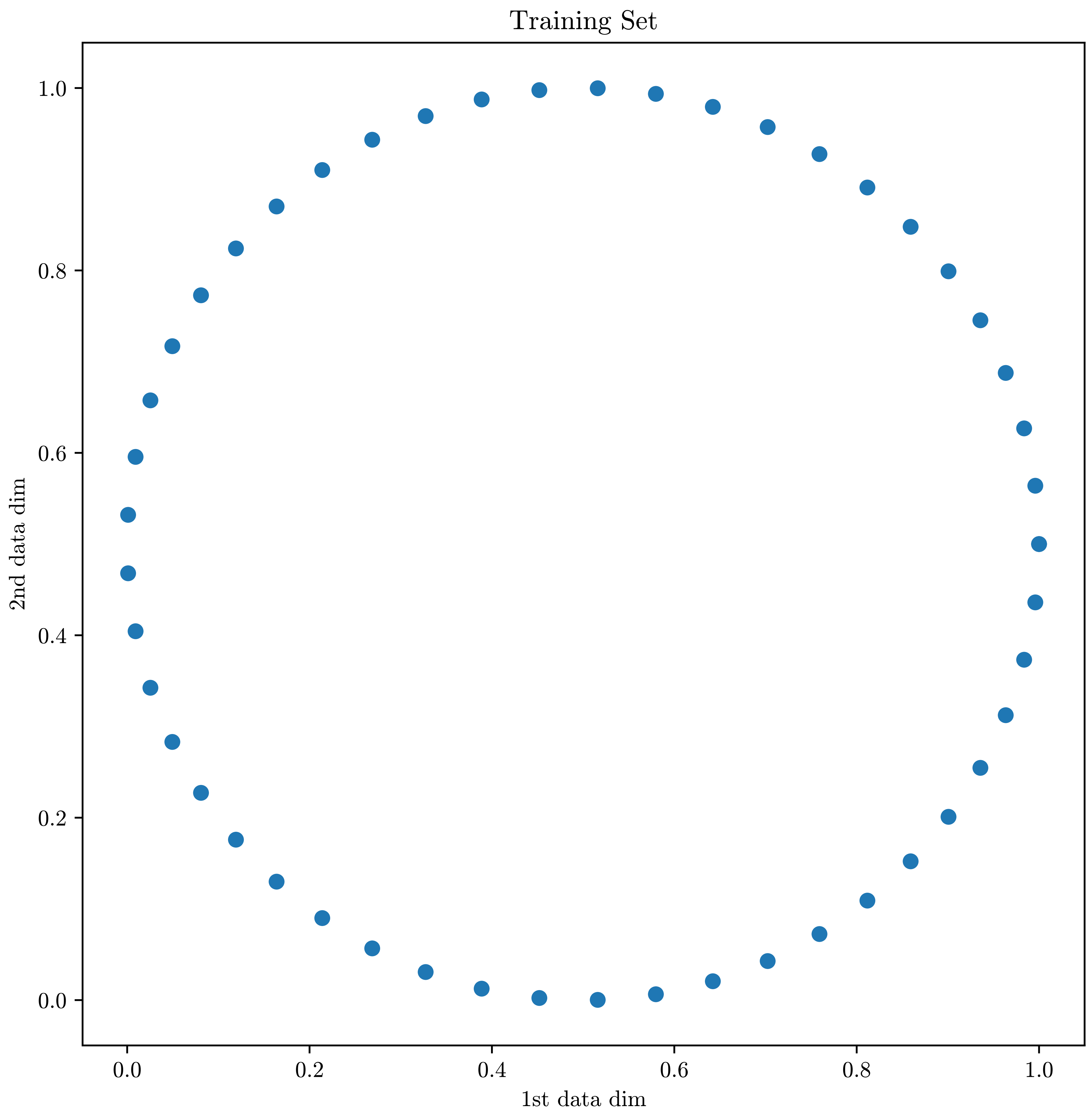




1. The Equivalence

Memorization in toy example

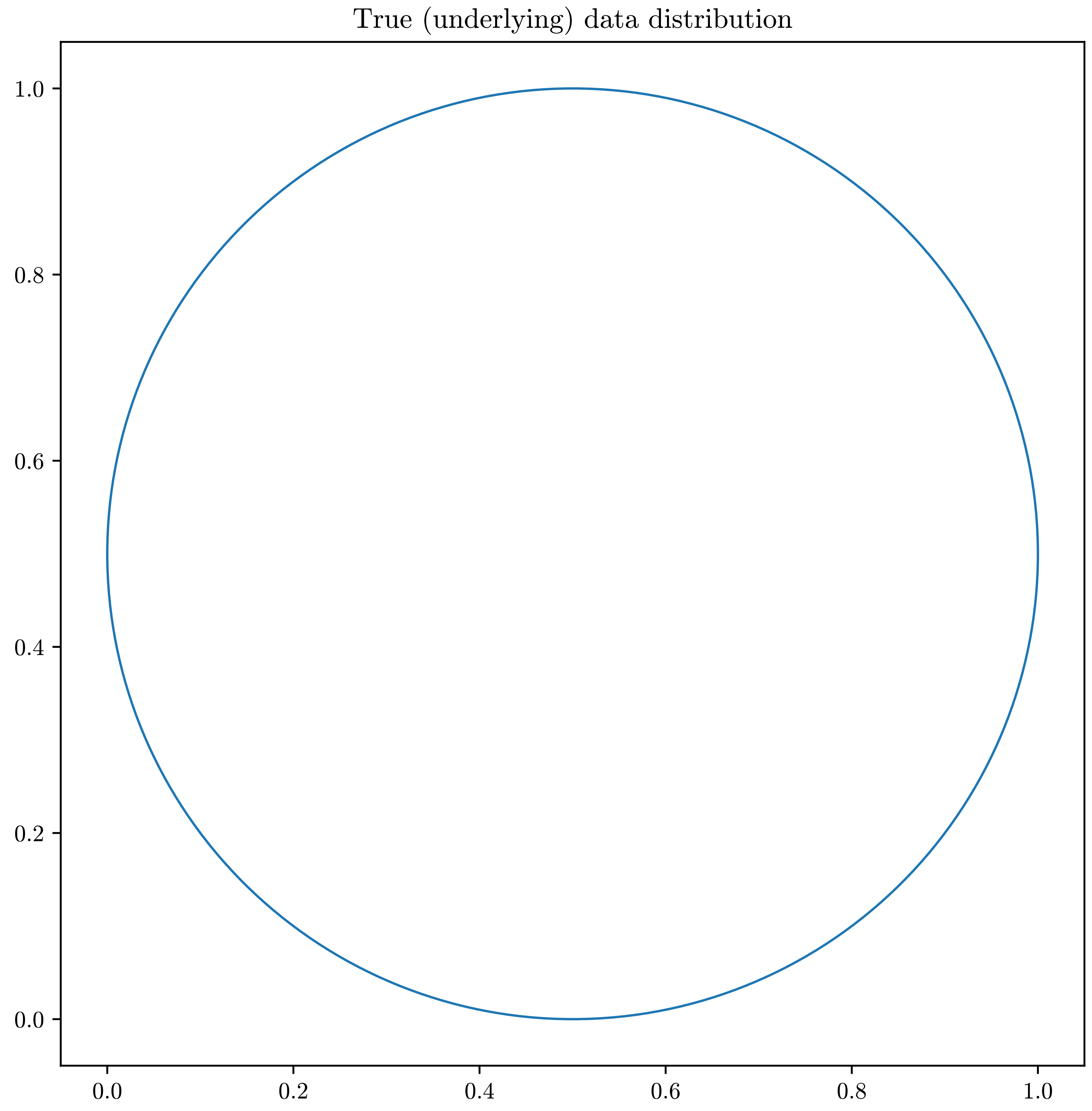
$$\mathcal{D} = \{\mathbf{x}^{(i)}\}_{i=1}^{N} \sim p_{\text{true}}(\mathbf{x}) \qquad p_{\text{data}}(\mathbf{x}) = \frac{1}{N}\sum_{i=1}^{N}\delta(\mathbf{x}-\mathbf{x}^{(i)})$$



1. The Equivalence

Memorization in toy example

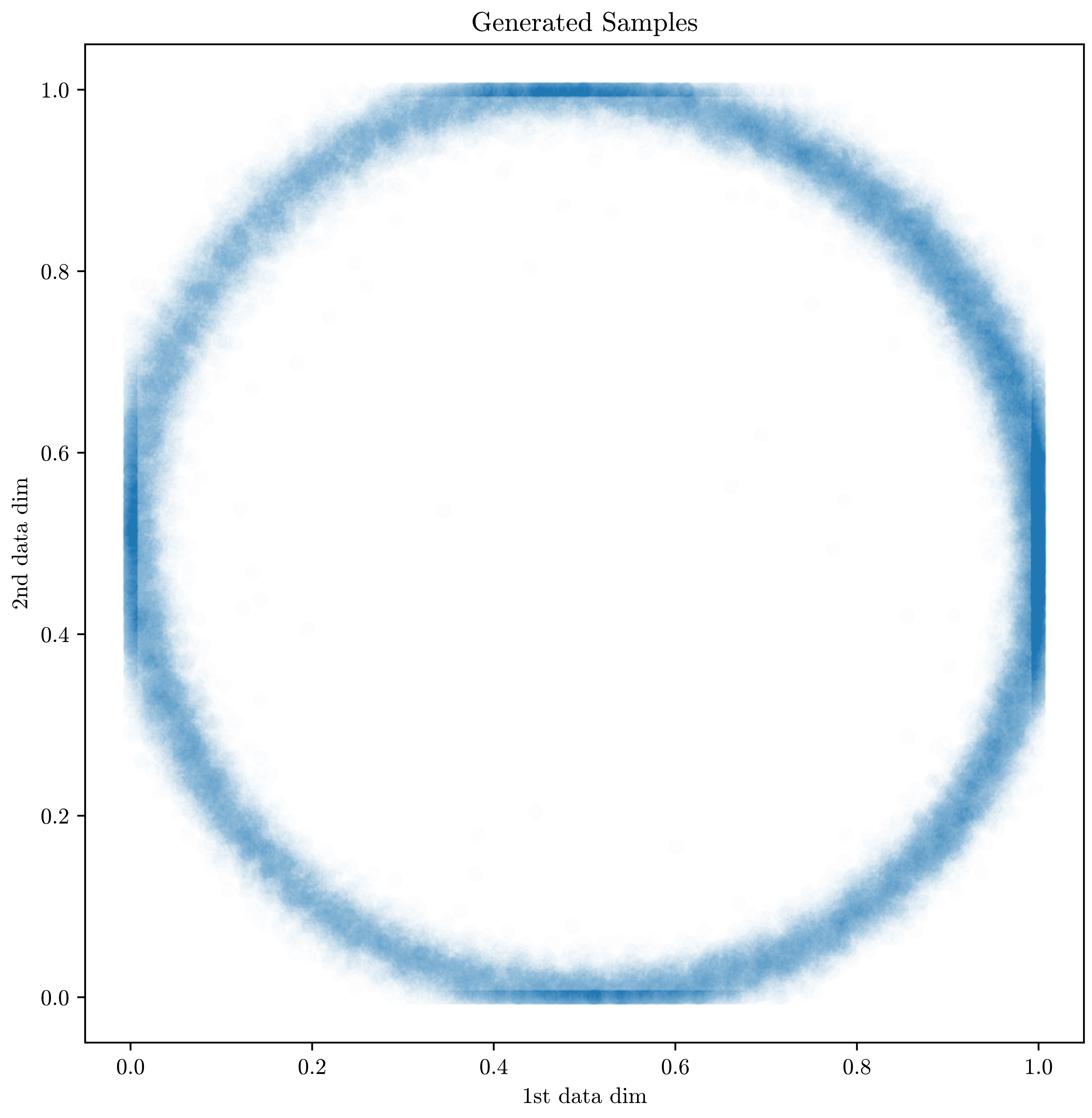
$$p_{\text{true}}(\mathbf{x})$$

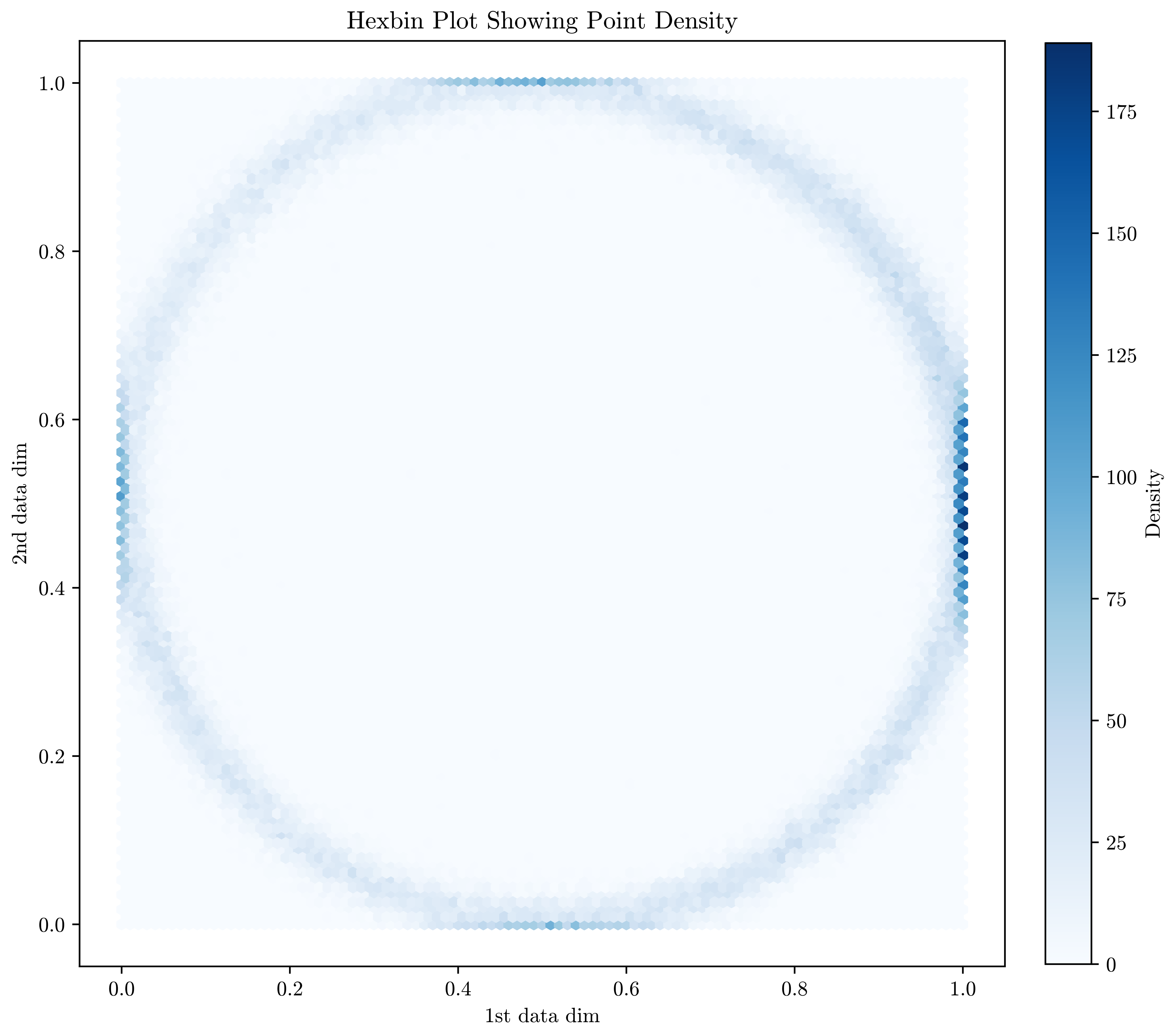


1. The Equivalence

Memorization in toy example

- Under-fit phase

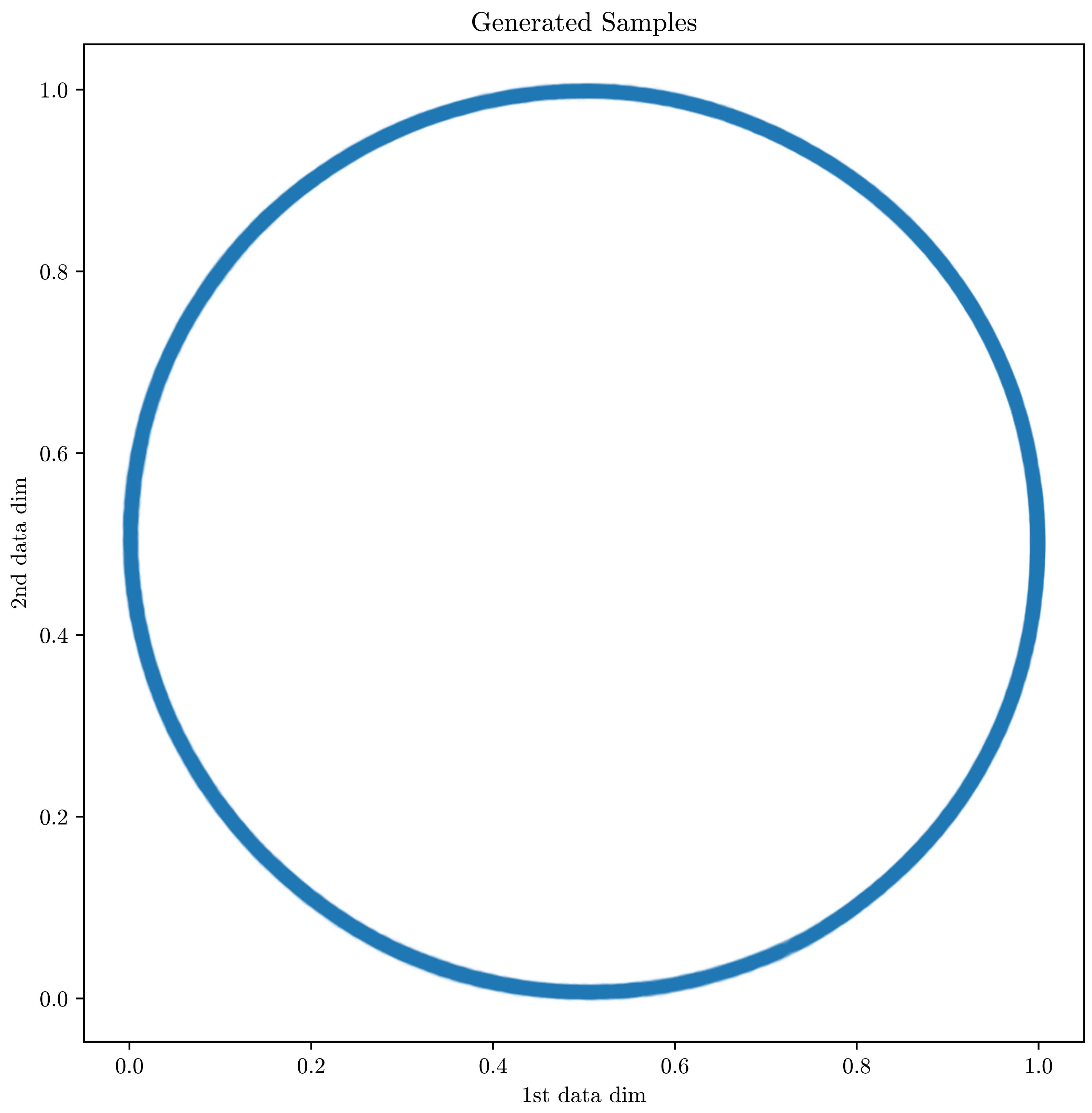


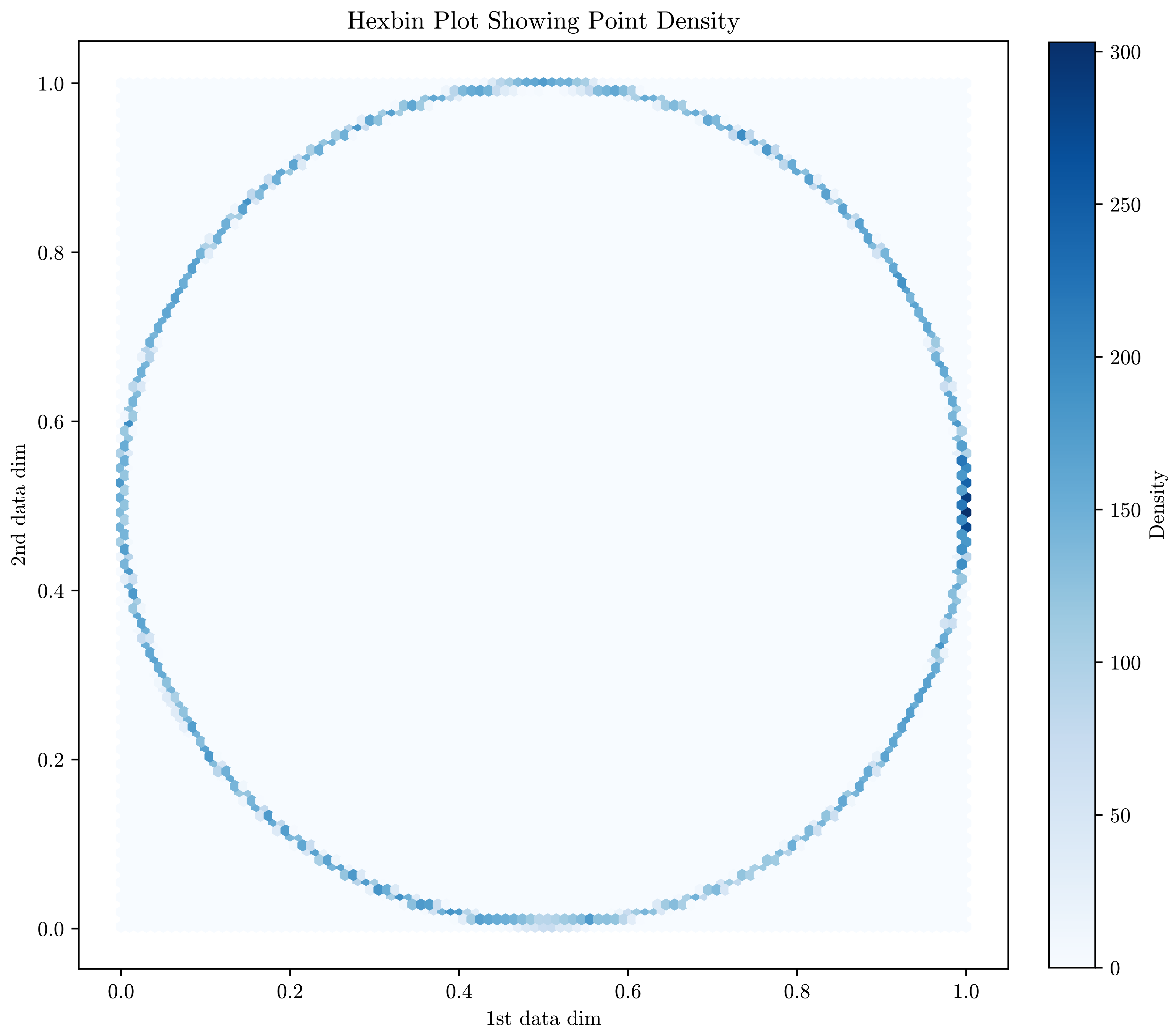


1. The Equivalence

Memorization in toy example

- desired $p_{\mathrm{true}}(\mathbf{x})$ phase

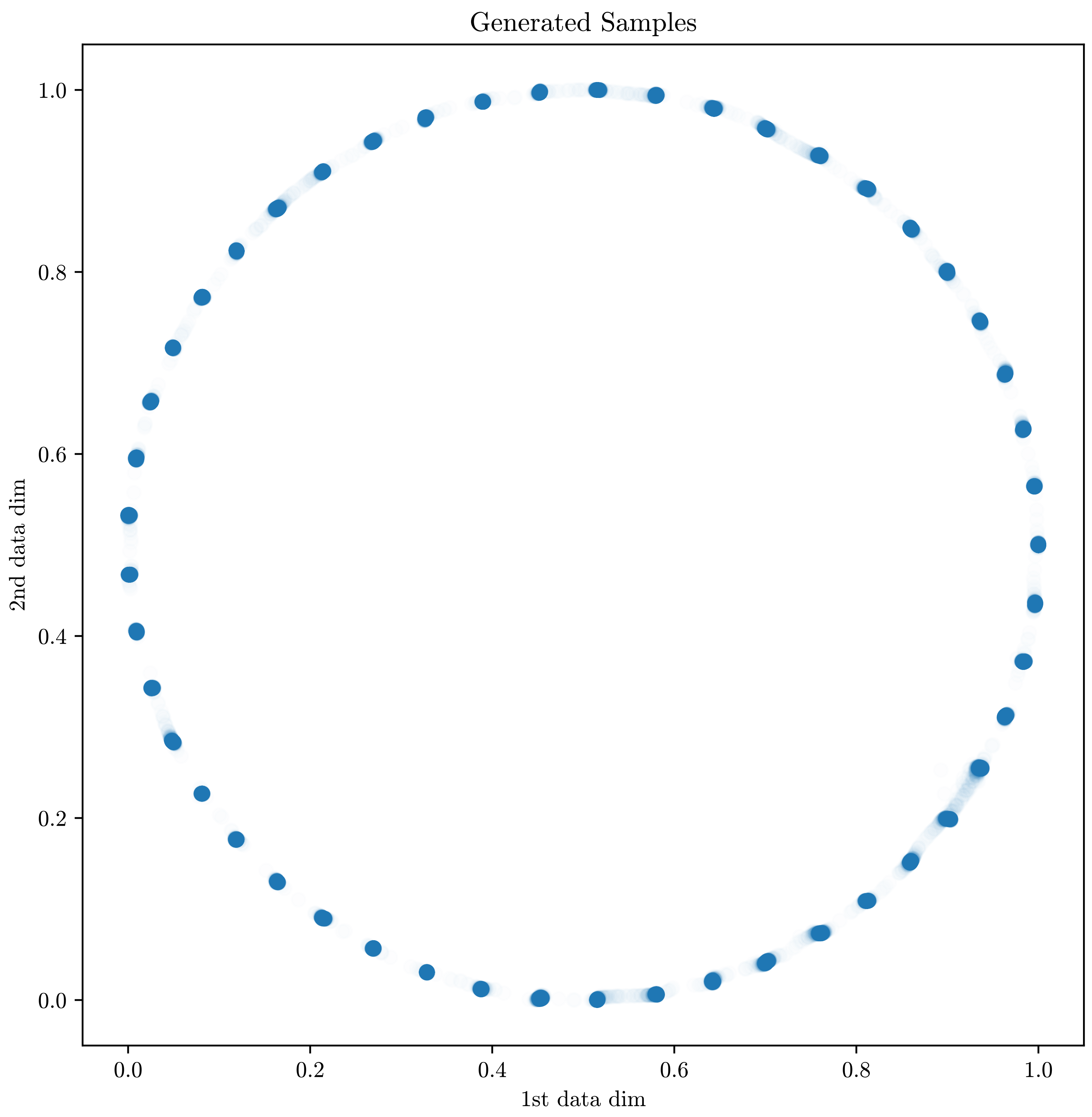


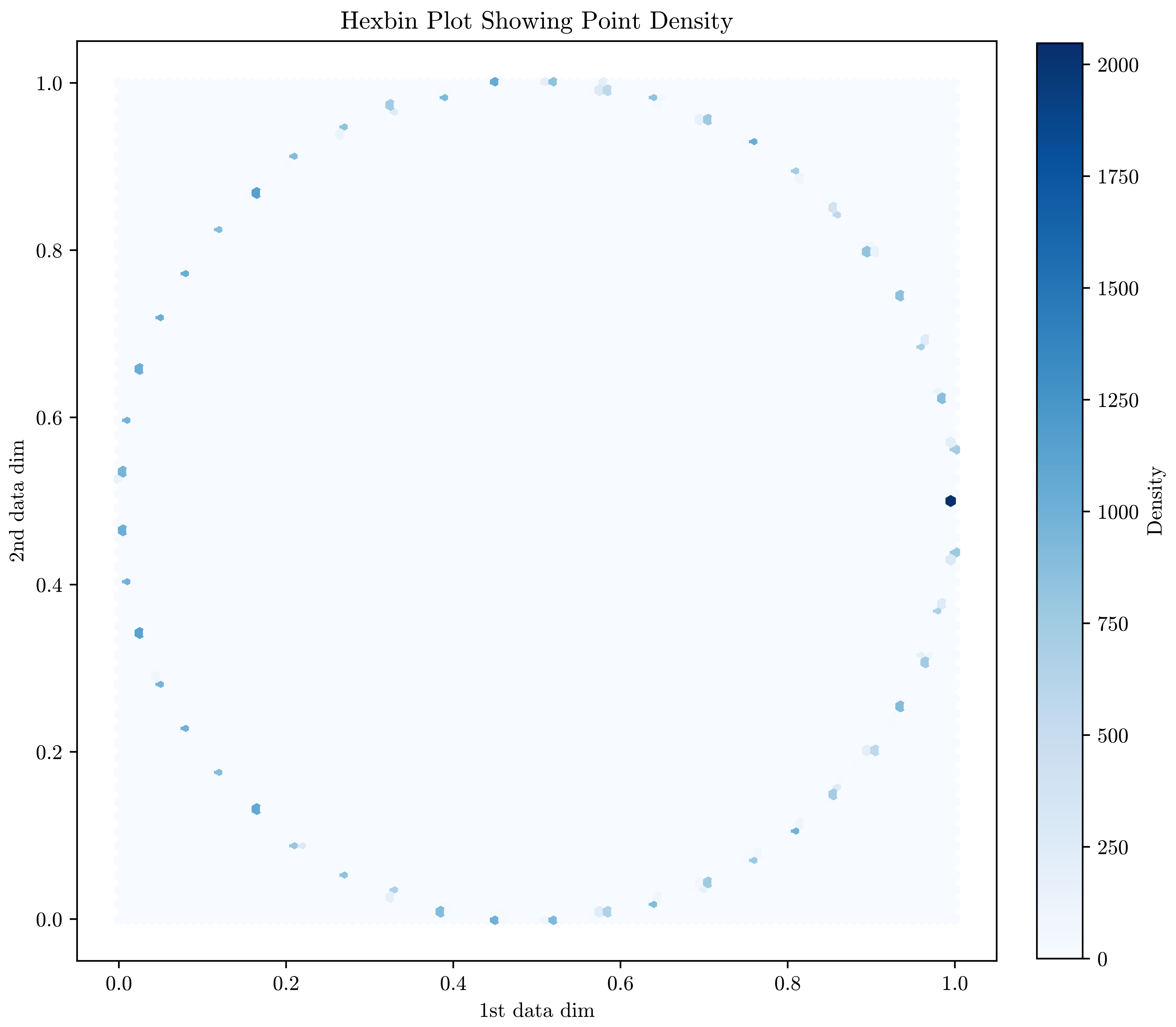


1. The Equivalence

Memorization in toy example

- complete memorization phase

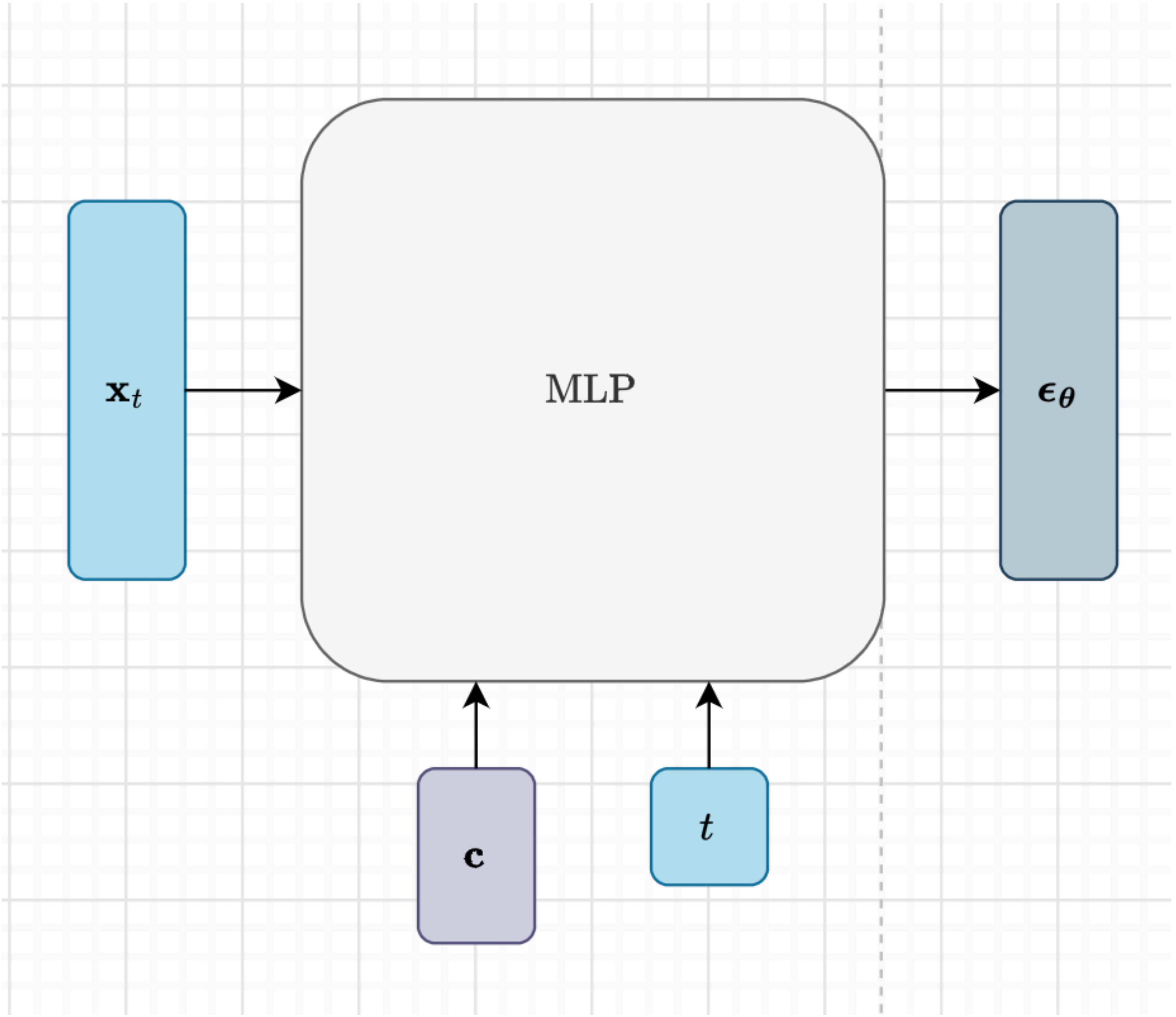




1. The Equivalence

Statement for image patches as conditions

When training a conditional generative model $p(\mathbf{x} \mid \mathbf{c})$, (assume $\mathbf{c}$ is patch instance) with noise augmentation (isotropic gaussian) to conditioning input $\mathbf{c}$, the trained diffusion model should converge to patch-distances closed-form parametrization (when converge to memorization/retrieval).



1. The Equivalence

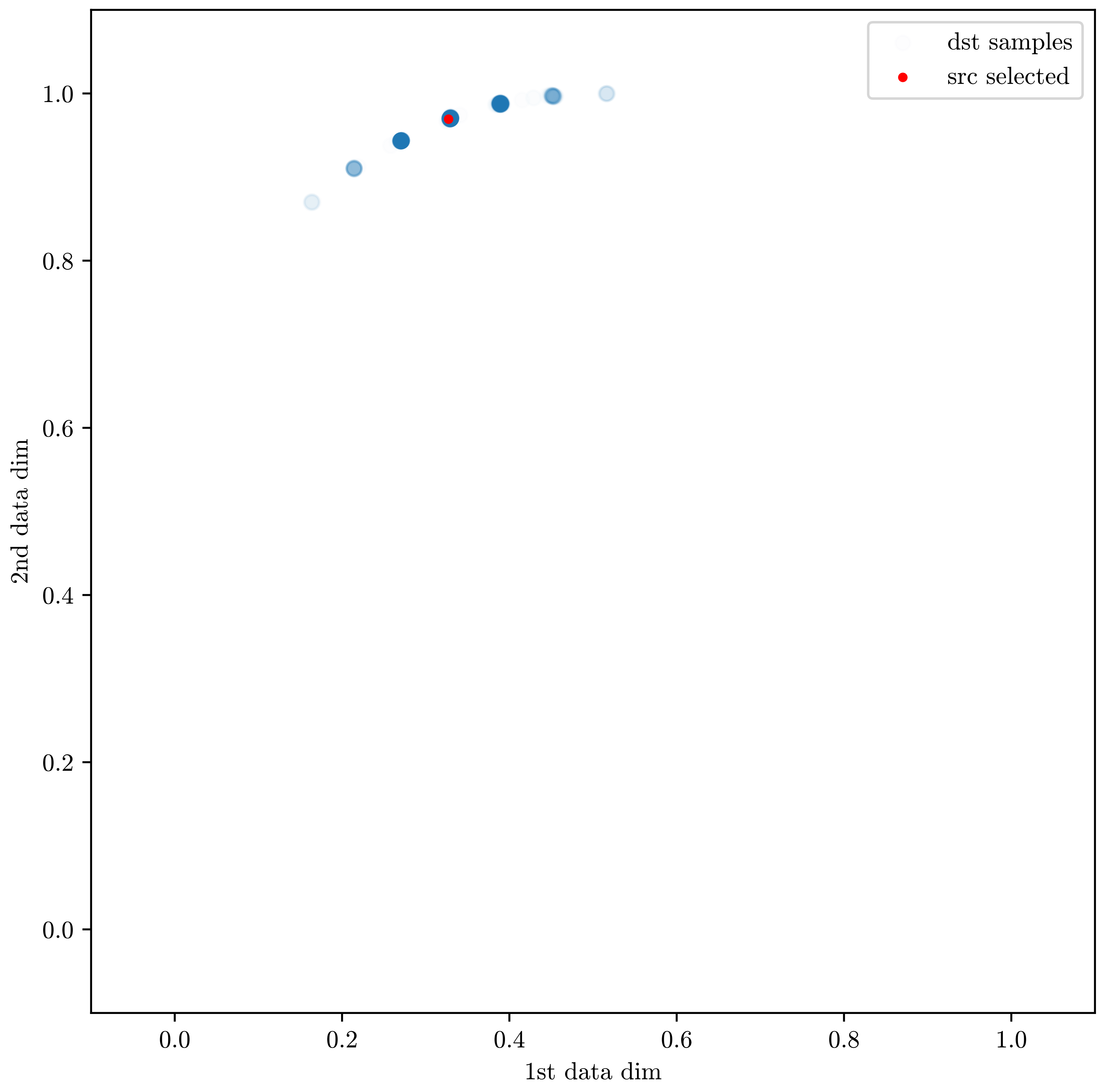
How to derive it?

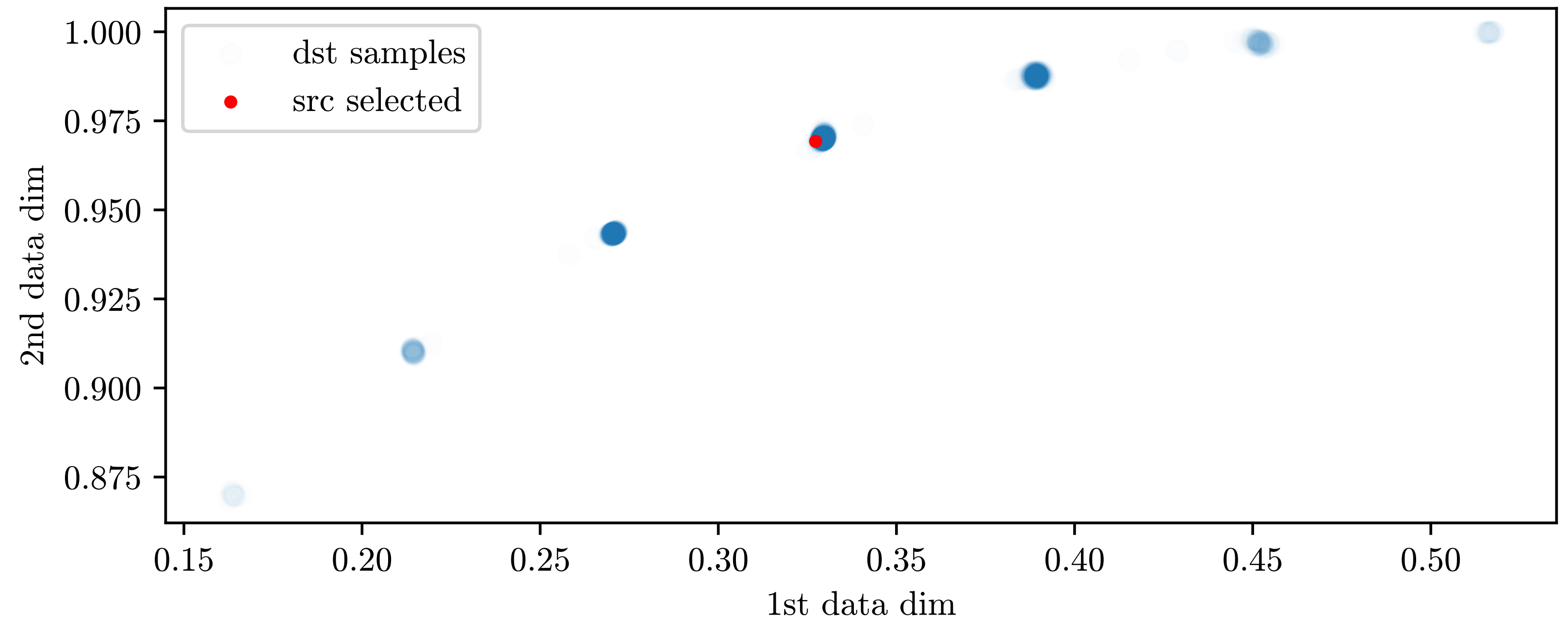
Assume the generative model is trained to the memorization stage, and the training is done with noise augmenting the conditions by $\mathcal{N}(\mathbf{0}, \sigma_{\text{aug}}\mathbf{I})$. Then at inference given a test $\mathbf{x}'_{\text{src}}$, the prob. of sample/retrieve $\mathbf{x}^{(i)}_{\text{dst}}$ in dataset $\mathcal{D}$ is

$$
\begin{aligned}
\Pr[\mathbf{x}_{\text{dst}} = {\mathbf{x}_{\text{dst}}}^{(i)} \,|\, \mathbf{x}_{\text{src}} = {\mathbf{x}_{\text{src}}}'] &= \frac{\Pr[\mathbf{x}_{\text{src}} = {\mathbf{x}_{\text{src}}}' \,|\, \mathbf{x}_{\text{dst}} = {\mathbf{x}_{\text{dst}}}^{(i)}] \Pr[\mathbf{x}_{\text{dst}} = {\mathbf{x}_{\text{dst}}}^{(i)}]}{\Pr[\mathbf{x}_{\text{src}} = {\mathbf{x}_{\text{src}}}']} \\
&= \frac{\Pr[\mathbf{x}_{\text{src}} = {\mathbf{x}_{\text{src}}}' \,|\, \mathbf{x}_{\text{dst}} = {\mathbf{x}_{\text{dst}}}^{(i)}] \Pr[\mathbf{x}_{\text{dst}} = {\mathbf{x}_{\text{dst}}}^{(i)}]}{\sum_{j=1}^{N} \Pr[\mathbf{x}_{\text{src}} = {\mathbf{x}_{\text{src}}}' \,|\, \mathbf{x}_{\text{dst}} = {\mathbf{x}_{\text{dst}}}^{(j)}] \Pr[\mathbf{x}_{\text{dst}} = {\mathbf{x}_{\text{dst}}}^{(j)}]} \\
&= \frac{\Pr[\mathbf{x}_{\text{src}} = {\mathbf{x}_{\text{src}}}' \,|\, \mathbf{x}_{\text{dst}} = {\mathbf{x}_{\text{dst}}}^{(i)}](1/N)}{\sum_{j=1}^{N} \Pr[\mathbf{x}_{\text{src}} = {\mathbf{x}_{\text{src}}}' \,|\, \mathbf{x}_{\text{dst}} = {\mathbf{x}_{\text{dst}}}^{(j)}](1/N)} \\
&= \frac{\Pr[\mathbf{x}_{\text{src}} = {\mathbf{x}_{\text{src}}}' \,|\, \mathbf{x}_{\text{dst}} = {\mathbf{x}_{\text{dst}}}^{(i)}]}{\sum_{j=1}^{N} \Pr[\mathbf{x}_{\text{src}} = {\mathbf{x}_{\text{src}}}' \,|\, \mathbf{x}_{\text{dst}} = {\mathbf{x}_{\text{dst}}}^{(j)}]} \\
&= \frac{\mathcal{N}\left({\mathbf{x}_{\text{src}}}'; {\mathbf{x}_{\text{src}}}^{(i)}, {\sigma_{\text{aug}}}^2\mathbf{I}\right)}{\sum_{j=1}^{N} \mathcal{N}\left({\mathbf{x}_{\text{src}}}'; {\mathbf{x}_{\text{src}}}^{(j)}, {\sigma_{\text{aug}}}^2\mathbf{I}\right)} \\
&= \frac{\exp\left(-\frac{1}{2{\sigma_{\text{aug}}}^2} \left\| {\mathbf{x}_{\text{src}}}' - {\mathbf{x}_{\text{src}}}^{(i)} \right\|_2^2\right)}{\sum_{j=1}^{N} \exp\left(-\frac{1}{2{\sigma_{\text{aug}}}^2} \left\| {\mathbf{x}_{\text{src}}}' - {\mathbf{x}_{\text{src}}}^{(j)} \right\|_2^2\right)}.
\end{aligned}
$$

2. Verify the Equivalence on a Toy Example

The equivalence in toy example — $p(\mathbf{x}_{\mathrm{dst}} \,|\, \mathbf{x}_{\mathrm{src}}, \mathbf{\Delta} = \mathrm{identity})$

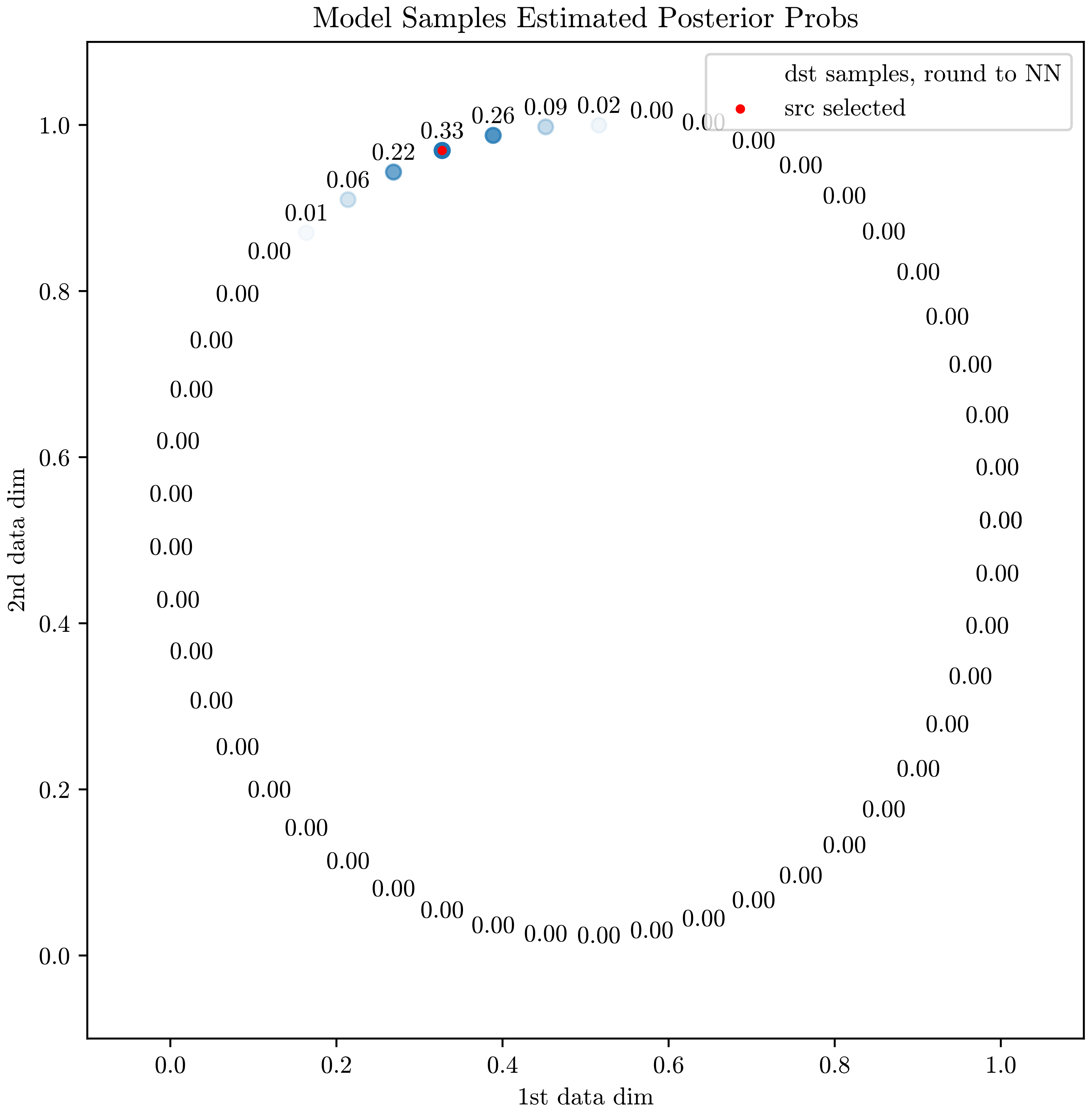


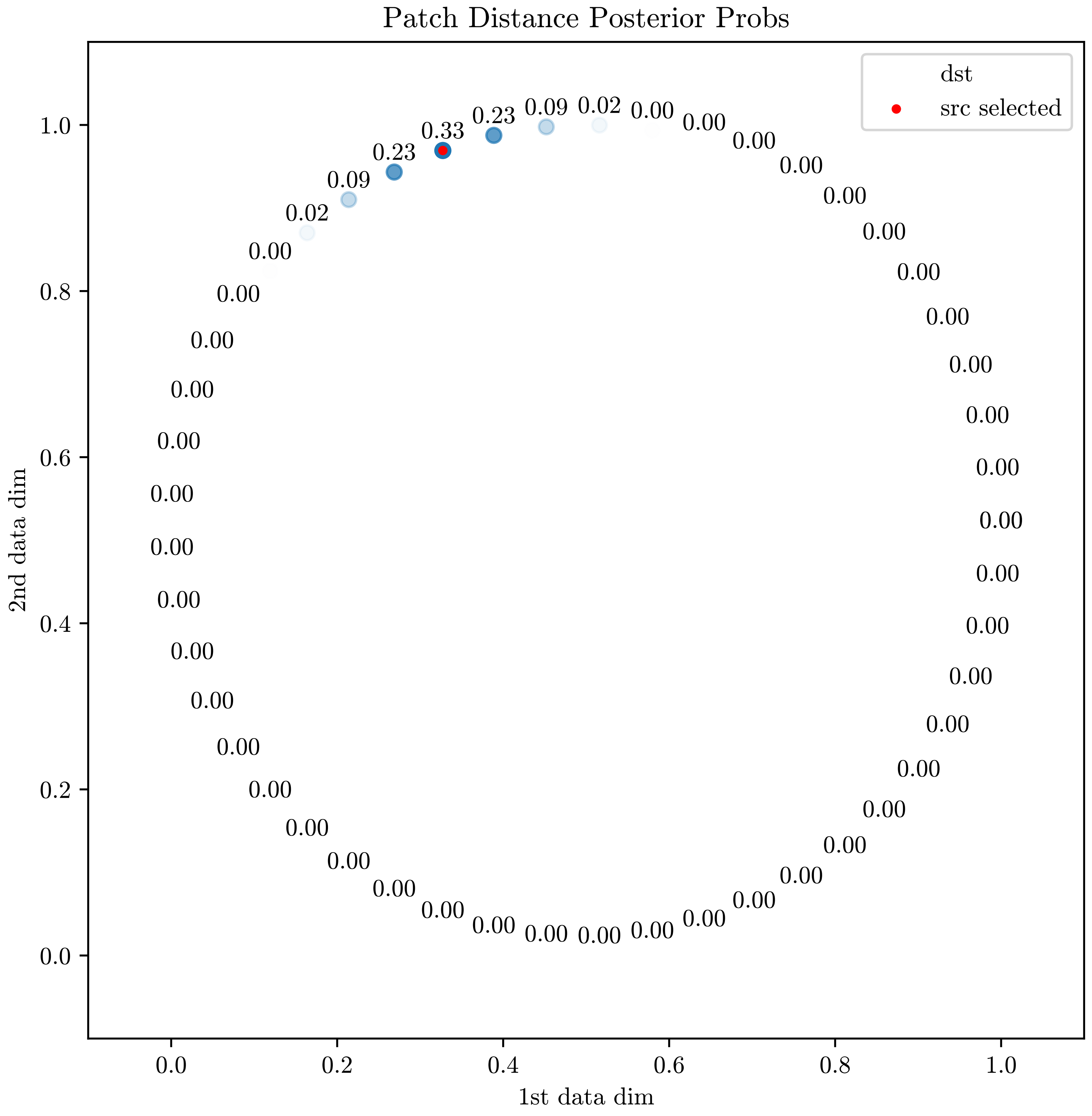


2. Verify the Equivalence on a Toy Example

The equivalence in toy example — $p(\mathbf{x}_{\mathrm{dst}} \mid \mathbf{x}_{\mathrm{src}}, \boldsymbol{\Delta} = \mathrm{identity})$

- Results converge to closed-form posterior probs, i.e., kernels.



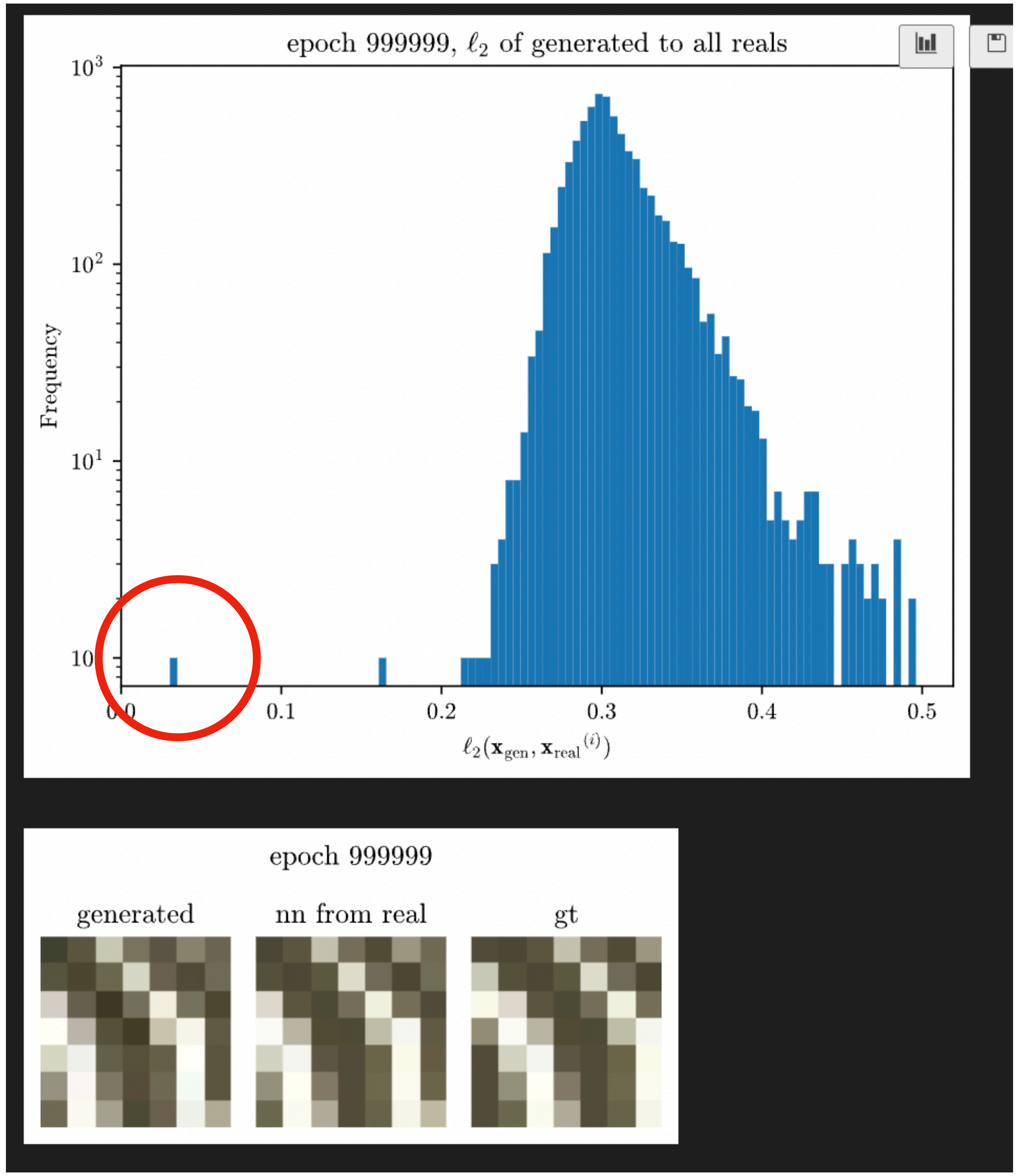


3. Verify Memorization on Real Patches

Train 0.5 to 1.0 for Upsample

- The distribution of l2 dists between a generated patch and all 'real' patches

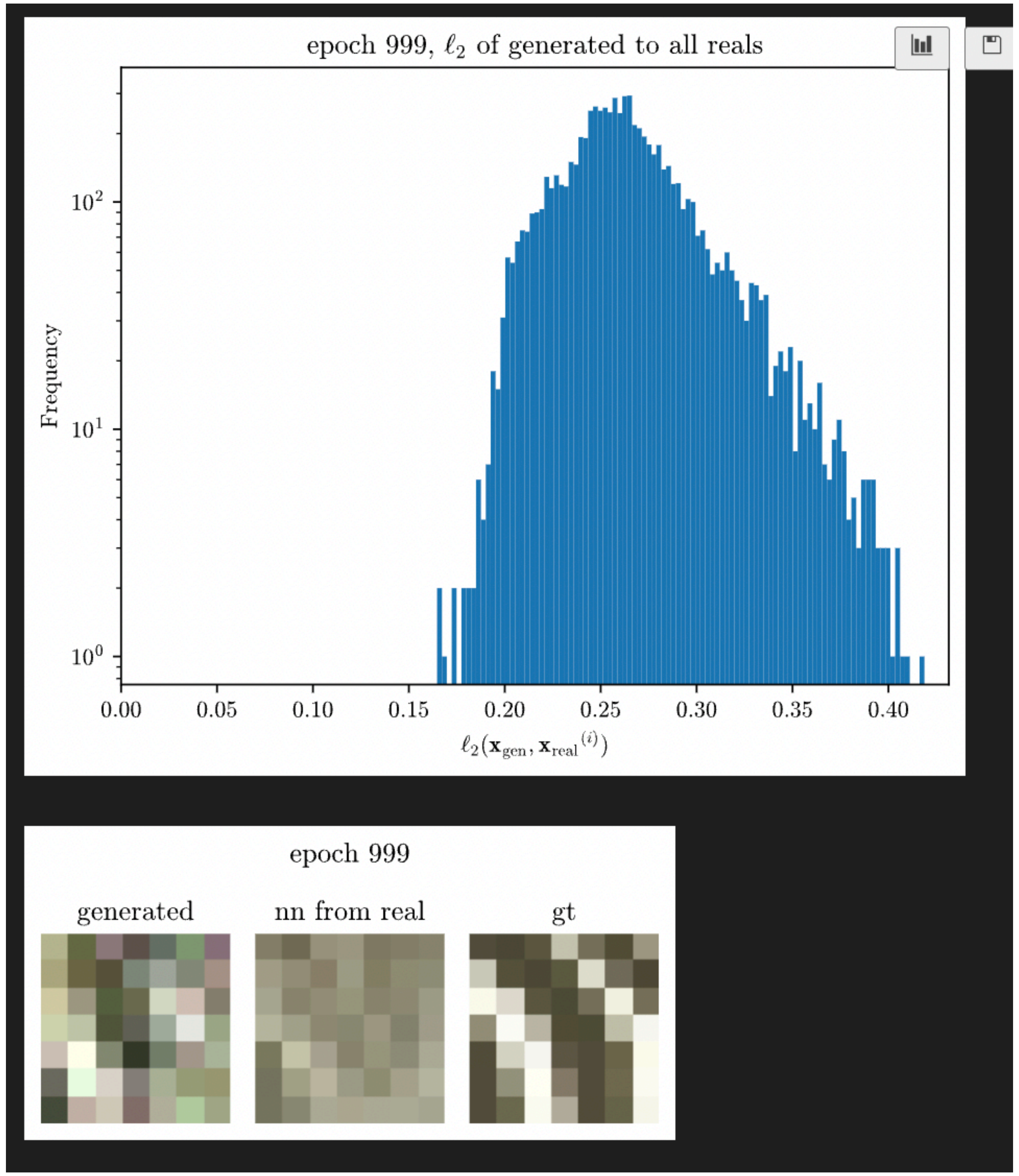
- Memorization occurs at end of training (have a very small l2 distance to a particular patch)



3. Verify Memorization on Real Patches

Train 0.5 to 1.0 for Upsample

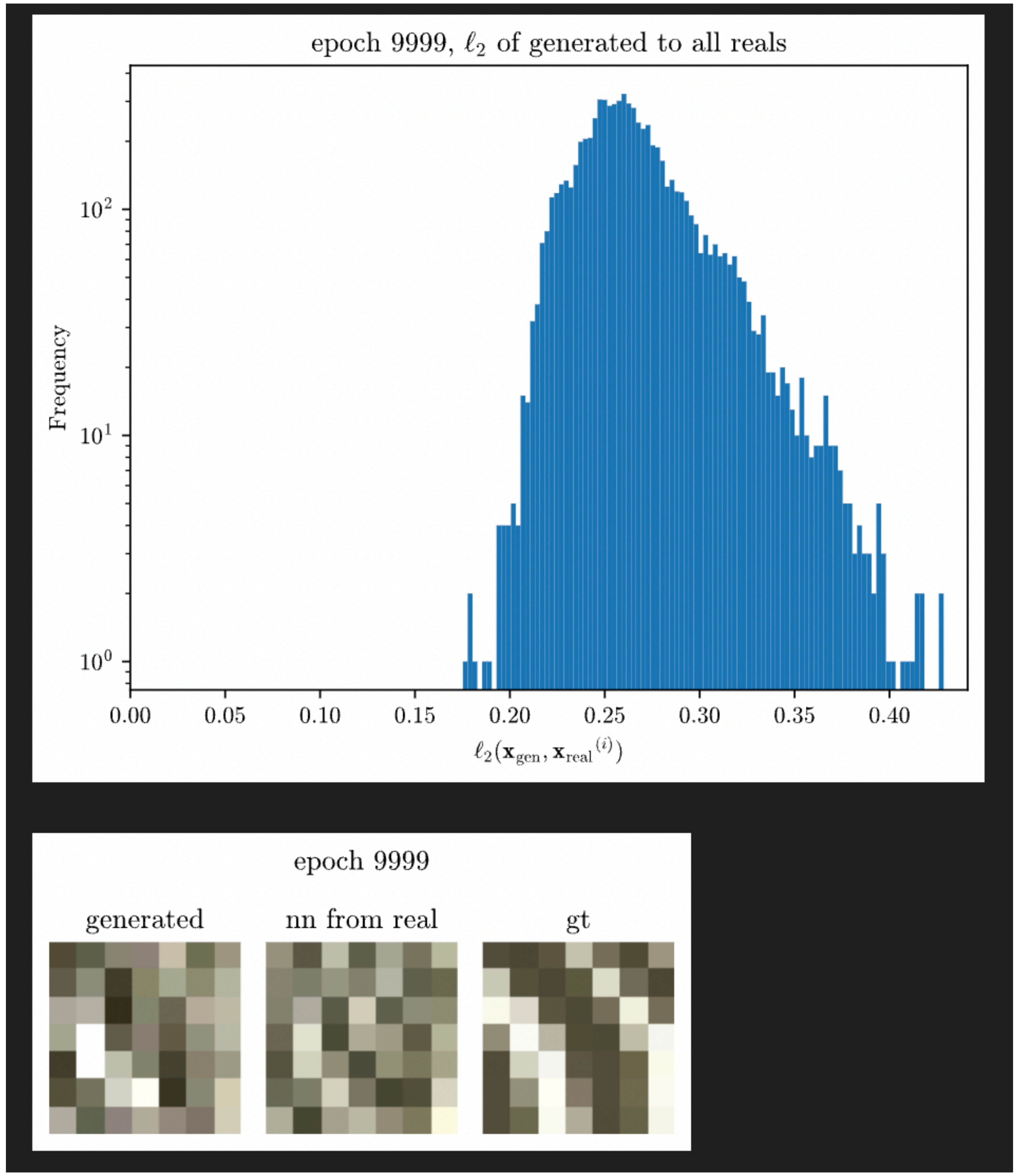
It's a evolving progress



3. Verify Memorization on Real Patches

Train 0.5 to 1.0 for Upsample

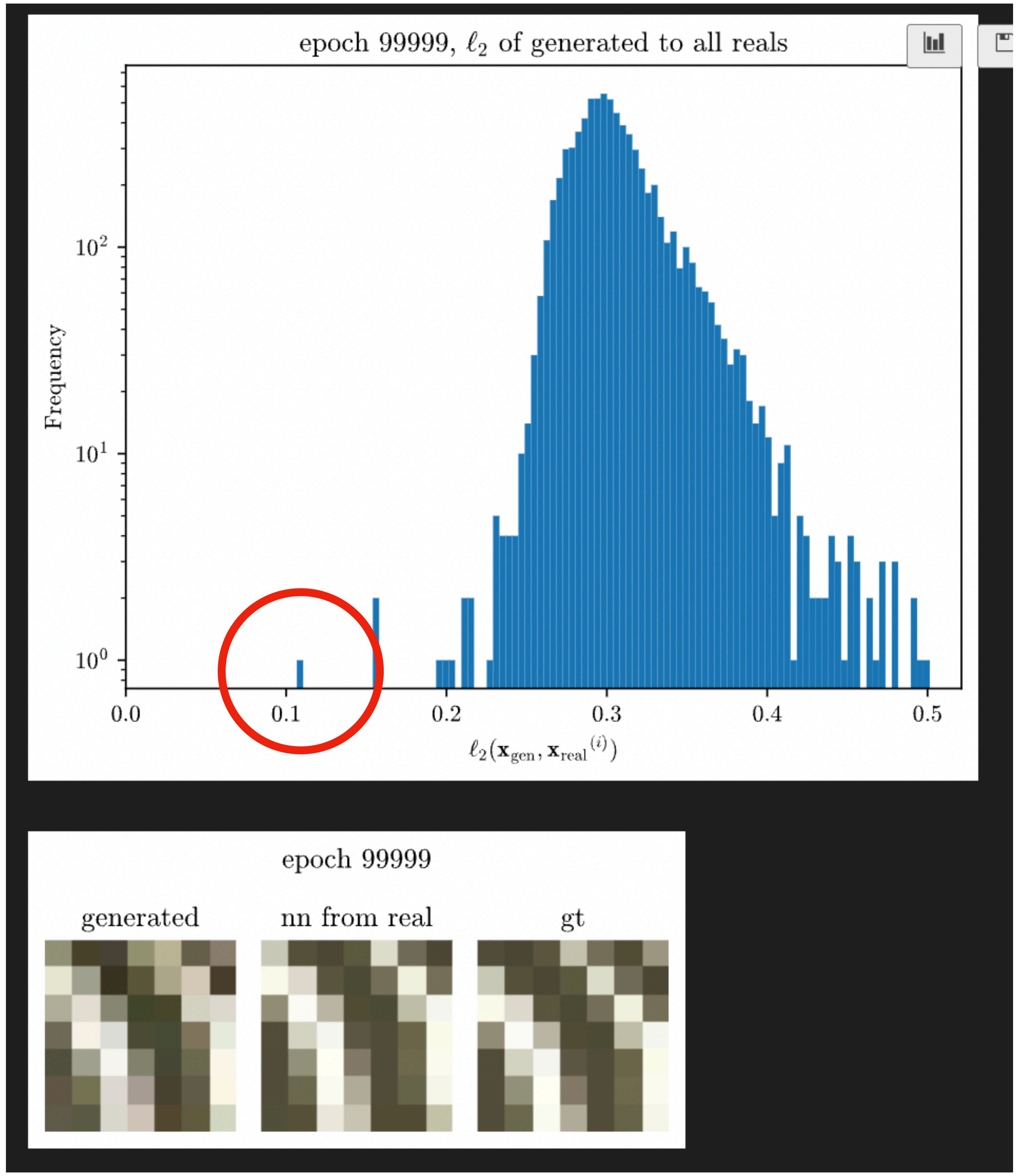
It's a evolving progress



3. Verify Memorization on Real Patches

Train 0.5 to 1.0 for Upsample

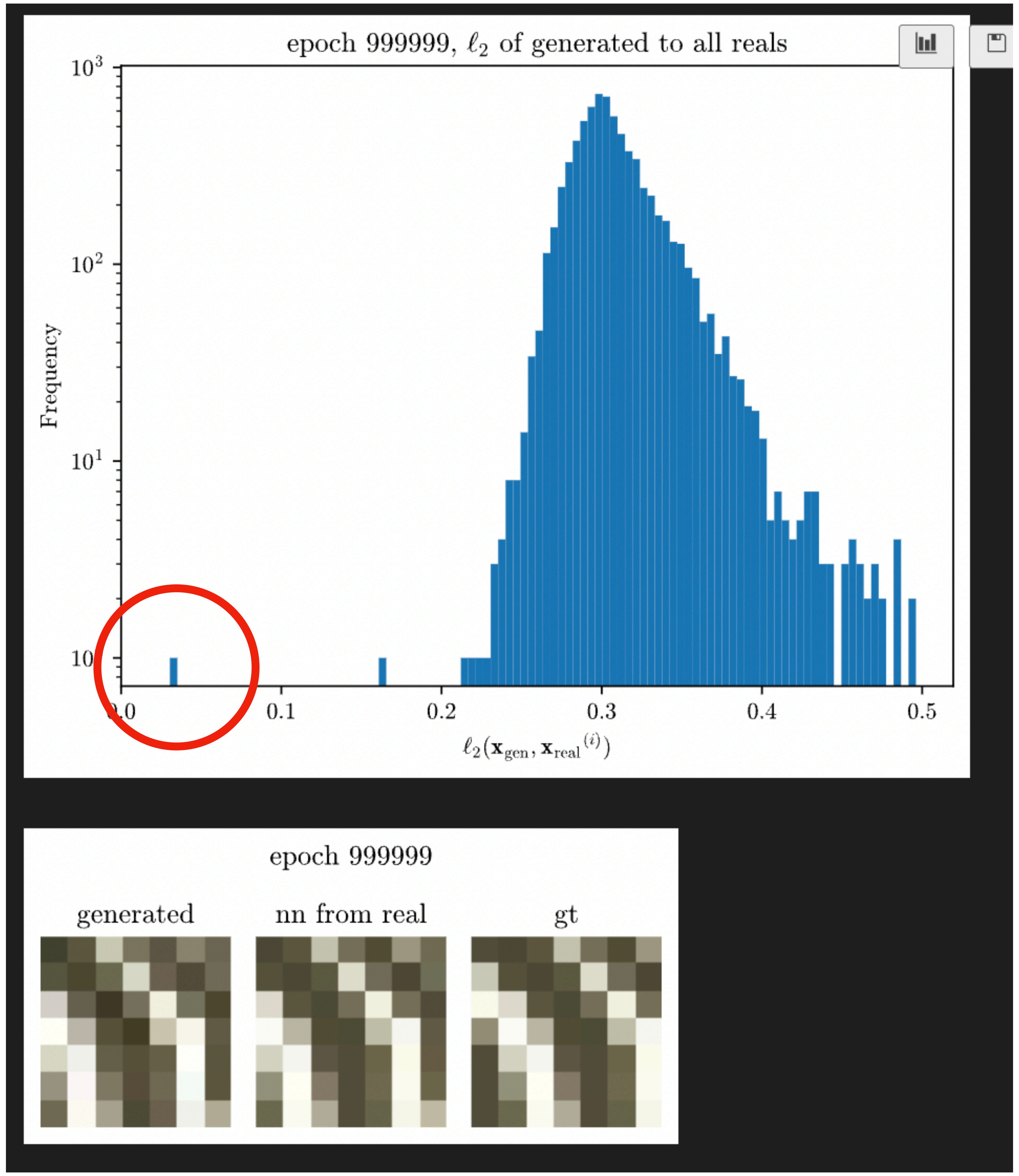
It's a evolving progress



3. Verify Memorization on Real Patches

Train 0.5 to 1.0 for Upsample

It's a evolving progress



3. Verify Memorization on Real Patches

Train 0.5 to 1.0 for Upsample

Even for untextured patch