Appendix

A. Adversarial Dynamics Embedding

The details of ADE derivations were originally provided in (Dai et al., 2019). We include relevant details here for completeness, since we make explicit use of these techniques in our training methods.

Given the EBM with $p_f(x) \propto \exp(f(x))$, ADE considers the augmented model

$$p_f(x,v) \propto \exp\left(f(x) - \frac{\lambda}{2} v^\top v\right), \tag{A.1}$$

where v is the auxiliary momentum variable. It has been proved in Dai et al. (2019) that the MLE of (A.1) is the same as the original model, *i.e.*,

$$\operatorname*{argmax}_f \widehat{\mathbb{E}}_{\mathcal{D}}\left[\log \int p(x,v)\, dv\right] = \operatorname*{argmax}_f \widehat{\mathbb{E}}_{\mathcal{D}}\left[\log p_f(x)\right]. \tag{A.2}$$

We then apply the primal-dual view of the MLE to the augmented model, leading to

$$\max_f \min_{q(x,v)} \widehat{\mathbb{E}}\left[f(x)\right] - \mathbb{E}_{q(x,v)}\left[f(x) - \frac{\lambda}{2} v^\top v - \log q(x,v)\right], \tag{A.3}$$

where $q(x,v)$ is the dual sampler in the exponential family of distributions.

To ensure that the dual sampler $q(x,v)$ is flexible and tractable, ADE utilizes the dynamics embedding parametrization. Specifically, we consider the Hamiltonian dynamics embedding as an example. In this setting, the sample first comes from an initial distribution $\left(x^0, v^0\right) \sim q_\omega^0(x,v)$, and then moves according to

$$(x', v') = \mathbf{L}_{f,\eta}(x,v) := \begin{pmatrix} v^{\frac{1}{2}} = v + \frac{\eta}{2}\nabla_x f(x) \\ x' = x + \eta v^{\frac{1}{2}} \\ v' = v^{\frac{1}{2}} + \frac{\eta}{2}\nabla_x f(x') \end{pmatrix}, \tag{A.4}$$

where η is defined as the leapfrog stepsize. After T iterations, we obtain

$$\left(x^T, v^T\right) = \mathbf{L}_{f,\eta} \circ \mathbf{L}_{f,\eta} \circ \ldots \circ \mathbf{L}_{f,\eta}\left(x^0, v^0\right), \tag{A.5}$$

where $\mathbf{L}_{f,\eta}$ can be one layer of the neural network. Together with the initial distribution q_ω^0, we obtain the parametrization of $q(x,v)$ with learnable parameters (η, ω). As justified in Dai et al. (2019), this parametrization is flexible and has a tractable density,

$$q^T\left(x^T, v^T\right) = q_\omega^0\left(x^0, v^0\right). \tag{A.6}$$

By plugging this parametrization into (A.3), we obtain the final objective,

$$\max_{f\in\mathcal{F}} \min_{\omega,\eta} \ell(f,\omega) := \widehat{\mathbb{E}}_{\mathcal{D}}[f] - \mathbb{E}_{(x^0,v^0)\sim q_\omega^0(x,v)}\left[f\left(x^T\right) - \frac{\lambda}{2}\left\|v^T\right\|_2^2\right] - H\left(q_\omega^0\right). \tag{A.7}$$

Dai et al. (2019) also introduces the Langevin and generalized Hamiltonian dynamics embedding for dual density parametrization in ADE. For more details, please refer to Dai et al. (2019).

A.1. Dynamics Embedding Distribution Parametrization

We now present the concrete implementation of the dual sampler $q(x_{1:n}, v|\theta)$ used in our paper. Following the ADE technique introduced above, parametrization of $q(x_{1:n}, v|\theta)$ is separated into parameterizing the initial distribution and the dynamics embedding. In our real-data experiment, we use the block RNN to parametrize $q_\omega^0(x_{1:n}, v|\theta)$, as introduced in Appendix F.1. Block RNN is simply a design choice; other alterantives such as normalizing flows can also parametrize $q_\omega^0(x_{1:n}, v|\theta)$, as shown in our synthetic-data experiment.

The output of the RNN is then treated as the starting sample to which T Hamiltonian/Langevin dynamics updates are applied, *i.e.*,

$$\left(x_{1:n}^{T}, v^{T}\right) = \mathbf{L} \circ \mathbf{L} \circ \ldots \circ \mathbf{L}\left(x_{1:n}^{0}, v^{0}\right), \tag{A.8}$$

where $\mathbf{L}$ can be either a Hamiltonian layer or a Langevin layer as specalized below,

$$\textbf{Hamiltonian layer:} \quad \left(x'_{1:n}, v'\right) = \mathbf{L}_{f,\eta}\left(x_{1:n}, v\right) := \left(\begin{array}{c} v_i^{\frac{1}{2}} = v_i + \frac{\eta}{2}\nabla_x f\left(x_i;\theta\right) \\ x'_i = x_i + \eta v_i^{\frac{1}{2}} \\ v'_i = v_i^{\frac{1}{2}} + \frac{\eta}{2}\nabla_x f\left(x'_i;\theta\right) \end{array} \middle| \, i = 1, \ldots, n \right), \tag{A.9}$$

with $v' = \{v'_i\}_{i=1}^n$.

$$\textbf{Langevin layer:} \quad \left(x'_{1:n}, v'\right) = \mathbf{L}^{\xi}_{f,\eta}\left(x_{1:n}\right) := \left(\begin{array}{c} v'_i = \xi_i + \frac{\eta}{2}\nabla_x f\left(x_i;\theta\right) \\ x'_i = x_i + v'_i \end{array} \middle| \, i = 1, \ldots, n \right), \tag{A.10}$$

with $\xi = \{\xi_i\}_{i=1}^n$, $\xi_i \sim q_\omega\left(\xi\right)$ and $v' = \{v'_i\}_{i=1}^n$.

Finally, following Theorem 4 in Dai et al. (2019), we obtain the parametrized dual sampler with tractable density as

$$\textbf{Hamiltonian embedding:} \qquad q^T\left(x_{1:n}^T, v^T|\theta\right) = q^0\left(x_{1:n}^0, v^0|\theta\right), \tag{A.11}$$

$$\textbf{Langevin embedding:} \qquad q^T\left(x_{1:n}^T, \left\{v^t\right\}_{t=1}^T|\theta\right) = q^0\left(x_{1:n}^0, \xi^0|\theta\right) \prod_{t=1}^{T-1} q_{\omega_i}\left(\xi^t\right). \tag{A.12}$$

By plugging this into the primal-dual view objective (21), we are able to learn the paramters in $\mathcal{EBP}$s and dual samplers.

B. Derivation of Special Cases of $\mathcal{EBP}$s

In this section, we provide the details for instantiating (un)conditional $\mathcal{EBP}$s to other specific models.

B.1. Latent Variable Representation of Gaussian Processes

We consider the latent variable model specified in (8), *i.e.*,

$$\theta \sim \mathcal{N}\left(\mathbf{0}, I_d\right), \tag{B.1}$$

$$f_w\left(x, t; \theta\right) = \frac{1}{2\sigma^2}\left\|x - \theta^\top \phi\left(t\right)\right\|^2. \tag{B.2}$$

To show that the marginal distribution follows

$$p\left(x_{t_1:t_n} | \left\{t_i\right\}_{i=1}^n\right) = \mathcal{N}\left(0, K\left(t_{1:,n}\right) + \sigma^2 I_n\right),$$

we integrate out of θ, *i.e.*,

$$p\left(x_{t_1:t_n} | \left\{t_i\right\}_{i=1}^n\right) = \int \prod_{i=1}^n p\left(x|t_i, \theta\right) p\left(\theta\right) d\theta = \mathcal{N}\left(0, \sigma^2 I_n + \phi\left(t_{1:n}\right)^\top \phi\left(t_{1:n}\right)\right),$$

where the last equation come from the integration of Gaussians. This shows that $\mathcal{GP}$s under the latent variable parametrization are a special case of $\mathcal{EBP}$s.

B.2. Latent Variable Representation of Student-t Processes

We consider the latent variable model specified in (10), *i.e.*,

$$\alpha \sim \mathcal{N}\left(\mathbf{0}, I_d\right), \tag{B.3}$$

$$\beta^{-1} \sim \Gamma\left(\frac{\nu}{2}, \frac{\gamma}{2}\right), \tag{B.4}$$

$$f_w\left(x, t; \theta\right) = \frac{\gamma\left\|x - \sqrt{\frac{\beta(\nu-2)}{\gamma}}\alpha^\top \phi\left(t\right)\right\|^2}{2\sigma^2\left(\nu-2\right)\beta}, \tag{B.5}$$

To show the marginal distribution follows

$$p\left(x_{t_1:t_n}|\left\{t_i\right\}_{i=1}^n\right)=\mathcal{T}\left(\nu,0,K\left(t_{1:n}\right)+\beta I_n\right),$$

we integrate out of $\theta=(\alpha,\beta)$. By integrating over α, we have

$$\begin{aligned}p\left(x_{t_1:t_n}|\beta,\left\{t_i\right\}_{i=1}^n\right) &= \int\prod_{i=1}^n p\left(x_{t_i}|t_i,\alpha,\beta\right)p\left(\alpha\right)d\alpha\\ &= \mathcal{N}\left(\mathbf{0},\left(\frac{\beta}{\gamma}\left(\nu-2\right)\right)\left(\underbrace{\sigma^2 I_n+\phi\left(t_{1:n}\right)^\top\phi\left(t_{1:n}\right)}_{\tilde{K}\left(t_{1:n}\right)}\right)\right),\end{aligned}$$

and by integrating over β, we have

$$\begin{aligned}p\left(x_{t_1:t_n}|\left\{t_i\right\}_{i=1}^n\right) &= \int p\left(x_{t_1:t_n}|\beta,\left\{t_i\right\}_{i=1}^n\right)p\left(\beta\right)d\beta\\ &\propto \int\exp\left(-\frac{1+\frac{\gamma X^\top\tilde{K}^{-1}X}{(\nu-2)}}{2\beta}\right)\beta^{-\frac{\nu}{2}-1}d\beta\\ &\propto \left(1+\frac{\gamma X^\top\tilde{K}^{-1}X}{(\nu-2)}\right)^{\frac{\nu+n}{2}}=\mathcal{T}\left(\nu,0,\tilde{K}\right)\end{aligned}$$

This shows that $\mathcal{TP}$s under the latent variable parametrization are a special case of $\mathcal{EBP}$s.

B.3. Topic Models

The topic models, including Bayesian sets (Ghahramani & Heller, 2005), probabilistic latent semantic index (Hofmann, 1999), latent Dirichlet allocation (LDA) family (Blei et al., 2003; Blei & Lafferty, 2007; Blei & McAuliffe, 2007), and replicated softmax (Hinton & Salakhutdinov, 2009), are also special cases of the unconditional $\mathcal{EBP}$s.

Here we consider the original LDA and replicated softmax as examples of directed and undirected topic models respectively. Other models follow a similar consideration.

Latent Dirichlet Allocation LDA is a representative of the directed topic model, which treats a document as a set of words. The model defines each component in (12) as

$$\begin{aligned}p\left(t_i\right) &= \mathcal{D}ir\text{-}\mathcal{M}ulti\left(\alpha\right),\\ p\left(x_{t_i}|t_i\right) &= \mathcal{M}ulti\left(\Phi^\top\mathbf{1}_{t_i}\right),\quad i=1,\ldots,n,\end{aligned}$$

where $t_i\in\{1,\ldots,d\}$ denotes the topic of each word and follows the Dirichlet-Multinomial distribution. The x_i is a k-dimensional one-hot encoding for each word in the vocabulary. Φ is a $d\times k$ matrix where each row denotes the distribution of words in one topic.

Replicated Softmax The replicated softmax (Hinton & Salakhutdinov, 2009) is proposed as an undirected topic model, which defines the joint distribution in (12) as a restricted Boltzmann machine with a linear potential function, *i.e.*,

$$p\left(x_{1:n},\theta\right)\propto\exp\left(-\sum_{i=1}^n\left(\theta^\top W x_i+b^\top x_i\right)-a^\top\theta\right),\tag{B.6}$$

where $\theta\in\{0,1\}^d$ can be seen as the latent topic assignment. Each x_i is a k-dimensional indication vector with only one element equals to 1 and the rest equal to 0. Together $\{x_i\}_{i=1}^n$ denote the one-hot encodings for the observed words in the document.

As a result, LDA and replicated softmax are special realizations of uncondtional $\mathcal{EBP}$s, where the index variables are either explicit defined or integrated out.

C. Proof Details of Theorem 2

Theorem 2 *If $n \geqslant m \geqslant 1$, and prior is exchangeable and consistent, then the marginal distribution $p(x_{1:n})$ will be exchangeable and consistent.*

Proof We simply verify the consistency of $p(x_{1:n})$ under the consistency of $p(\{t_i\}_{i=1}^m)$.

$$\begin{aligned}
& \int p(x_{1:n})\, dx_{m+1:n} \\
= & \int\int p(x_{t_1:t_n}|\{t_i\}_{i=1}^n)\, p(\{t_i\}_{i=1}^n)\, d\{t_i\}_{i=1}^n\, dx_{m+1:n} \\
= & \int \left(\int p(x_{t_1:t_n}|\{t_i\}_{i=1}^n)\, dx_{m+1:n}\right) p(\{t_i\}_{i=1}^n)\, d\{t_i\}_{i=1}^n \\
= & \int p(x_{t_1:t_m}|\{t_i\}_{i=1}^m) \left(\int p(\{t_i\}_{i=1}^n)\, dt_{m+1:n}\right) dt_{1:m} \\
= & \int p(x_{t_1:t_m}|\{t_i\}_{i=1}^m)\, p(\{t_i\}_{i=1}^m)\, dt_{1:m} = p(x_{1:m}).
\end{aligned}$$

The exchangeability of $p(x_{1:n})$ directly comes from the exchangeablity of $p(x_{t_1:t_n})$ and $p(\{t_i\}_{i=1}^n)$,

$$\begin{aligned}
p(x_{1:n}) &= \int p(x_{t_1:t_n}|\{t_i\}_{i=1}^n)\, p(\{t_i\}_{i=1}^n)\, d\{t_i\}_{i=1}^n \\
&= \int p(\pi(x_{t_1:t_n})|\pi(\{t_i\}_{i=1}^n))\, p(\pi(\{t_i\}_{i=1}^n))\, d\{t_i\}_{i=1}^n \\
&= p(\pi(x_{t_1:t_n}))
\end{aligned}$$

■

D. More Details of Inference

D.1. Further Neural Collapsed Inference for Unconditional $\mathcal{EBP}$s

In the main text, we introduce the neural collapsed inference for $\mathcal{EBP}$s to eliminate the posterior inference of $\{t_i\}_{i=1}^n$. We can further exploit the neural collapsed inference idea to eliminate θ.

Recall the model with $\{t_i\}_{i=1}^n$ collapsed,

$$\begin{aligned}
p_{w'}(x_{1:n}|\theta) &= \prod_{i=1}^n \int p_w(x_{t_i}|\theta, t_i)\, p(t_i)\, dt_i \\
&\propto \prod_{i=1}^n \int \exp\left(f_w(x_{t_i}, t_i; \theta) - Z(f_w, t_i; \theta) + h_v(t_i)\right) dt_i \\
&\approx \prod_{i=1}^n \frac{1}{Z(f_{w'}; \theta)} \exp\left(f_{w'}(x_i; \theta)\right).
\end{aligned}$$

We further consider $p(\theta) \propto \exp(g_u(\theta))$, then we have

$$\begin{aligned}
& \prod_{i=1}^n \int p_{w'}(x_{1:n}|\theta)\, p(\theta)\, d\theta \\
\propto & \int \exp\left(\sum_{i=1}^n f_{w'}(x_i; \theta) + g_u(\theta)\right) d\theta \\
\approx & \frac{1}{Z(w'')} \exp\left(\sum_{i=1}^n f_{\tilde{w}'}\left(x_i; \sum_{i=1}^n \phi_{u'}(x_i)\right)\right) := p_{w''}(x_{1:n}),
\end{aligned} \tag{D.1}$$

where we denote $w'' = \{\tilde{w}', u'\}$. The last approximation comes from the fact that

1) the integration over θ leads to a distribution over $x_{1:n} \in \otimes^n \Omega$;

2) since $p(x_{1:n})$ is an integration of latent variable model, it should be exchangeable.

Therefore, we consider deepsets (Zaheer et al., 2017) in the reparametrized EBM as a special choice of (D.1), which satifies these two conditions in order to approximate the integrated model.

The attention mechanism is another choice for the reparametrized EBM, *i.e.*,

$$p_{w''}(x_{1:n}) = \frac{1}{Z(w'')} \exp\left(\sum_{i=1}^{n} \texttt{attn}_{\tilde{w}'}(x_i; x_{1:n})\right),$$

where $\texttt{attn}_{w''}(x; x_{1:n}) = \sum_{i=1}^{n} \frac{\exp\left(\phi(x)^\top \phi(x_i)\right) h(x_i)}{\sum_{j=1}^{n} \exp\left(\phi(x)^\top \phi(x_j)\right)}$ with w'' denoting the parameters in $\phi(\cdot)$ and $h(\cdot)$.

We can apply ADE to such neural collapsed reparametrization if the task only concerns set generation (i.e., does not use θ). We take the deepsets parametrization (D.1) as an example (the attention parametrization follows similarly). Specifically, we learn the parameters in w'' via

$$\max_{w''} \min_{q\left(\{x_i\}_{i=1}^n, v\right)} \widehat{\mathbb{E}}_{\mathcal{D}} \left[\sum_{i=1}^{n} f_{\tilde{w}'}\left(x_i; \sum_{i=1}^{n} \phi_{u'}(x_i)\right)\right] - \mathbb{E}_{q\left(\{x_i\}_{i=1}^n, v\right)} \left[\sum_{i=1}^{n} f_{\tilde{w}'}\left(x_i; \sum_{i=1}^{n} \phi_{u'}(x_i)\right) - \frac{\lambda}{2} v^\top v\right] - H\left(q\left(\{x_i\}_{i=1}^n, v\right)\right). \quad \text{(D.2)}$$

Following the ADE technique, we can parametrize the initialization distribution $q_\omega^0\left(\left\{x_i^0\right\}_{i=1}^n, v^0\right)$ using an RNN and refine $q\left(\{x_i\}_{i=1}^n, v\right)$ with learnable Hamiltonian/Langevin dynamics as introduced in Section A.

Connection to Gibbs Point Processes ($\mathcal{GPP}$s) and Determinantal Point Processes ($\mathcal{DPP}$s): In fact, if we adopt the most general model collapsed unconditional $\mathcal{EBP}$ for arbitrary n, we obtain

$$p_{w''}(x_{1:n}) = \frac{1}{Z_{w''}} \exp\left(f_{w''}(x_{1:n})\right), \quad \text{(D.3)}$$

which includes the Gibbs point processes (Dereudre, 2019) with $f_{w''}$ satisfying several mild conditions for the regularity of $\mathcal{GPP}$. Particularly, the determinatal point processes (Lavancier et al., 2015; Kulesza et al., 2012) can be instantiated from $\mathcal{GPP}$ by setting the potential function to be $\log\det$ of some kernel function, which can also be parametrized by neural network, *e.g.*, Xie et al. (2016).

Therefore, the proposed algorithm can straightforwardly applied for (deep) $\mathcal{GPP}$s and $\mathcal{DPP}$s learning. It should be emphasized that by exploiting the proposed primal-dual MLE framework, we automatically obtain a deep neural network parametrized dual sampler with the learned model simultaneously, which can be used in inference and bypass the notorious sampling difficulty in $\mathcal{GPP}$ and $\mathcal{DPP}$.

D.2. Inference for Conditional $\mathcal{EBP}$

In the conditional $\mathcal{EBP}$ setting where the index $\{t_i\}_{i=1}^n$ are given for any cardinality n, we are modeling:

$$p\left(x_{t_1:t_n} \middle| \{t_i\}_{i=1}^n\right) = \int \prod_{i=1}^{n} p(x|\theta, t_i) p(\theta) d\theta \quad \text{(D.4)}$$

For simplicity, we use capital letters to denote the set. We denote T_{train}, X_{train} as the observed sets of points, and T_{test}, X_{test} as the un-observed inputs and targets. For the rest of this section, we denote $T = T_{train} \cup T_{test}$ and $X = X_{train} \cup X_{test}$. Then the predictive model is to infer

$$p(X_{test}|T, X_{train}) = \frac{p(X_{train}, X_{test}|T)}{\sum_{X'} p(X_{train}, X'|T)} = \frac{p(X_{train}, X_{test}|T)}{p(X_{train}|T)} = \frac{p(X_{train}, X_{test}|T)}{p(X_{train}|T_{train})} \quad \text{(D.5)}$$

where the last equal sign is due to the consistency condition of stochastic processes. Below we briefly review several existing conditional neural processes.

D.2.1. REVIEW OF CONDITIONAL NEURAL PROCESSES

The learning of processes is generally done via maximizing the marginal likelihood $p(X|T)$. The following neural processes, however, perform learning by maximizing the predictive distribution $p(X|T, X_{train})$ directly.

Conditional Neural Processes (Garnelo et al., 2018a) directly parametrize the conditional distribution $p(X|T, X_{train})$ as:

$$p(X|T, X_{train}) = \prod_{i=1}^{|X|} \mathcal{N}(x_{t_i}|\mu_{t_i}, \sigma_{t_i}) \tag{D.6}$$

where μ_{t_i} and σ_{t_i} are outputs of some neural network $g(t_i, T_{train}, X_{train})$ with permutation invariance.

Neural Processes (Garnelo et al., 2018b) model the distribution of $p(X|T)$ as

$$p(X|T) = \int \prod_{i=1}^{|X|} \mathcal{N}(x_{t_i}|g(t_i, \theta), \sigma)p(\theta)d\theta, \tag{D.7}$$

where $g(t_i, \theta)$ can be learned using ELBO:

$$\log p(X|T) \geqslant \mathbb{E}_{q(\theta|T,X)}\Big[\sum_{i=1}^{|X|} \log \mathcal{N}(x_{t_i}|g(t_i, \theta), \sigma) + \log p(\theta) - \log q(\theta|T, X)\Big]. \tag{D.8}$$

During learning, however, Garnelo et al. (2018b) again performs MLE on the predictive model:

$$\log p(X_{pred}|T, X_{context}) \geqslant \mathbb{E}_{q(\theta|T,X)}\Big[\sum_{x_{t_j} \in X_{pred}} \log \mathcal{N}(x_{t_j}|g(t_j, \theta), \sigma) + \log p(\theta|T_{context}, X_{context}) - \log q(\theta|T, X)\Big], \tag{D.9}$$

where $X_{context}$ denotes a subset of X_{train} and $X_{pred} = X_{train} \setminus X_{context}$, that are randomly splitted from X_{train}. The $p(\theta|T_{context}, X_{context})$ is the true posterior after observing $T_{context}, X_{context}$. This posterior then serves as the prior of the predictive model according to Bayes' rule. However, since $p(\theta|T_{context}, X_{context})$ is not tractable in general, Garnelo et al. (2018b) uses $q(\theta|T_{context}, X_{context})$ to approximate this term instead.

Attentive Neural Processes (Kim et al., 2019) is an extension to Neural Processes (Garnelo et al., 2018b) by using self-attention to parametrize the variational posterior. Besides θ from the variational posterior, their model also adds the deterministic context embedding r computed using attention, together with location x_{t_i}, into the mean function $g(x_{t_i}, r, \theta)$. Other than these differences, the training procedure is the same as Garnelo et al. (2018b), which still uses Gaussian for observation modeling.

D.2.2. LEARNING CONDITIONAL $\mathcal{EBP}$

The proposed $\mathcal{EBP}$s exploits flexible energy-based model, intead of Gaussian distributions in $\mathcal{NP}$s and theirs variants:

$$p(X|T, \theta) \propto \exp\left(f(X, T; \theta)\right) p(\theta) \propto \exp\left(\sum_{i=1}^{|X|} f(x_{t_i}, t_i; \theta)\right). \tag{D.10}$$

The ELBO then becomes:

$$\log p(X|T) \geqslant \mathbb{E}_{q(\theta|T,X)}\Big[\sum_{i=1}^{|X|} f(x_{t_i}, t_i; \theta)) - A(\theta, t_i) + \log p(\theta) - \log q(\theta|T, X)\Big], \tag{D.11}$$

where $A(\theta, t) = \log \int \exp(f(x, t; \theta))dx$ is the log partition function.

Using the ADE technique, we have $A(\theta, t) = \max_{q'(x|\theta,t)} \mathbb{E}_{q'(x|\theta,t)} [f(x, t; \theta)] + H(q')$. By plugging this into the ELBO, we arrive at the learning objective which tries to maximize the marginal likelihood w.r.t.

$$\max_{q,h} \min_{q'} \mathbb{E}_{q(\theta|T,X)} \Big[\log p(\theta) - \log q(\theta|T, X) + \sum_{i=1}^{|X|} f(x_{t_i}, t_i; \theta)) - \mathbb{E}_{x\sim q'} f(x, t_i; \theta) - H(q') \Big]. \tag{D.12}$$

Different from the family of neural processes, we are able to directly optimize for $p(X|T)$ using the above objective. We can easily learn the predictive distribution $p(X|T, X_{train})$ similar to other neural processes by replacing the prior with the variational posterior from the observed set T_{train}, X_{train}.

D.2.3. Prediction using $\mathcal{EBP}$s

Without loss of generality, we illustrate the prediction of a single point (t, x) given the training set, namely $p(x|t, X_{train}, T_{train})$.

As is pointed out in Eq (D.5), $p(x|t, X_{train}, T_{train}) \propto p(x, X_{train}|t, T_{train})$. We use the variational posterior (denoted as $q(\theta)$ below for simplicity) to approximate the lower bound:

$$\begin{aligned} \log p(x, X_{train}|t, T_{train}) &= \log \left(\int p(\theta) \exp\left(f(x, t; \theta) - A(\theta, t)\right) \prod_{i=1}^{|X_{train}|} \exp\left(f(x_{t_i}, t_i; \theta)) - A(\theta, t_i)\right) d\theta \right) \\ &\geqslant \max_{q(\theta)} \mathbb{E}_{q(\theta)} \left[f(x, t; \theta)\right] + \mathbb{E}_{q(\theta)} \left[G(\theta, t, X_{train}, T_{train})\right], \end{aligned} \tag{D.13}$$

where

$$G(\theta, t, X_{train}, T_{train}) = \log p(\theta) - \log q(\theta) - A(\theta, t) + \sum_{i=1}^{|X_{train}|} f(x_{t_i}, t_i; \theta)) - A(\theta, t_i) \tag{D.14}$$

In practice, we use $q(\theta|X_{train}|T_{train})$ to replace the above $p(\theta)$, which results in the predictive distribution approximated by:

$$\begin{aligned} p(x|t, T_{train}, X_{train}) &\geqslant \exp\left(\mathbb{E}_{q(\theta)} \left[f(x, t; \theta)\right]\right) \cdot \exp\left(\mathbb{E}_{q(\theta)} G(\theta, t, X_{train}, T_{train})\right) \\ &\propto \exp\left(\mathbb{E}_{q(\theta)} \left[f(x, t; \theta)\right]\right) \\ &\simeq \exp\left(\mathbb{E}_{q(\theta|X_{train}, T_{train})} \left[f(x, t; \theta)\right]\right) \end{aligned} \tag{D.15}$$

E. Additional Related Work

The proposed $\mathcal{EBP}$s bridge the gap between stochastic processes and models of exchangeable data. We summarize these two related topics below:

Stochastic Processes. Exploiting stochastic processes for conditional distribution modeling has a long line of research, starting from Gaussian processes for regression (Williams & Rasmussen, 1996) to being generalized to classification (Opper & Winther, 2000) and ordinal regression (Chu & Ghahramani, 2005). One of the major bottlenecks of $\mathcal{GP}$s is the memory and computational costs — $\mathcal{O}\left(N^2\right)$ in memory and $\mathcal{O}\left(N^3\right)$ in computation respectively, where N denotes the number of training samples. Although low-rank and sparse approximations have been proposed (Williams & Seeger, 2000; Quiñonero-Candela & Rasmussen, 2005; Snelson & Ghahramani, 2007; Titsias, 2009; Hensman et al., 2013) to reduce these costs, flexibilty of $\mathcal{GP}$s and its variants remains a critical issue to be addressed.

Student-t processes are derived by adding the inverse-Wishart process prior to the kernel function (Shah et al., 2014), allowing the kernel function to adapt to data. As we discussed in Section 3.2, student-t processes essentially rescales each dimention in the likelihood model, and thus still has restricted modeling flexibilty. Another line of research to further improve the flexibility of $\mathcal{GP}$s is to introduce structures or deep compositions into kernels, such as Duvenaud et al. (2013); Damianou & Lawrence (2012); Bui et al. (2016); Wilson et al. (2016); Al-Shedivat et al. (2017). Even though neural networks can be incorporated in constructing these kernels, the likelihood estimations of these models are still restricted to known distributions, limiting their modeling flexibility.

Neural processes (Garnelo et al., 2018b) and its varaints (Garnelo et al., 2018a; Kim et al., 2019; Louizos et al., 2019) introduce neural networks to stochatic processes beyond $\mathcal{GP}$s. However, their likelihoods are still restricted to known distributions, *e.g.*, Gaussian. Besides the modeling restriction, these models are learned by maximizing log-*predictive* distribution, rather than the log-*marginal* distibution, which may lead to suboptimial solutions.

The most competitive model w.r.t. $\mathcal{EBP}$s is the implicit processes (Ma et al., 2018), which uses the implicit models as the likelihood in (4). However, due to the intractability of the implicit likelihood, $\mathcal{GP}$s are introduced for variational inference, which negatively impacts the initially designed flexility in implicit processes, as demonstrated in Section G.1.

Exchangeable Probabilistic Models. The generative models for exchangeable data is a separate research topic which mostly relies on the De Finetti's Theorem. Bayesian sets (Ghahramani & Heller, 2005) considers the latent variable model with Bernoulli distribution likelihood and beta prior for a set of binary data. The conjugacy of Bernoulli and beta distributions leads to tractability in Bayesian inference, but limits the flexibility. Topic models, *e.g.*, pLSI (Hofmann, 1999), LDA family (Blei et al., 2003; Blei & Lafferty, 2007; Blei & McAuliffe, 2007), and replicated softmax (Hinton & Salakhutdinov, 2009) generalize Bayesian sets by introducing more complicated local latent variables, but the likelihood is still restricted to simple distributions.

To make distribution modeling more flexible, neural networks have been introduced to likelihood estimation of latent variable models (Edwards & Storkey, 2016). However, this work only exploits the analytic form of parametrization within known distributions. Korshunova et al. (2018); Yang et al. (2019) use normalizing flows to improve modeling flexibility of exchangeable data. Flow-based models, compared to $\mathcal{EBP}$s, still restrict the underlying distribution (by requiring the transformations to be invertable), and therefore cannot fully utilize the expressiveness of neural networks.

The Gibbs point processes (Dereudre, 2019), including Poission point processes, Hawkes point processes (Hawkes, 1971) and determinantal point processes (Lavancier et al., 2015; Kulesza et al., 2012) as special cases, is also an alternative flexible model for exchangeable data. As we discussed in Appendix D.1, by the neural collapsed inference technique, we reduce the collapsed $\mathcal{EBP}$ to $\mathcal{GPP}$, which implies the flexibility of the proposed $\mathcal{EBP}$. Moreover, this connection also highlights that the proposed primal-dual ADE can be used for $\mathcal{GPP}$ and $\mathcal{DPP}$ estimation.

F. Experiment Details

F.1. Architecture and Training Details

As preluded in Section D, we use deepsets (Zaheer et al., 2017) to parametrize $q(\theta|x_{1:n})$ as a diagonal Gaussian. Specifically, each input x_i undergoes 1D-convolutions with kernel size 1 and filter sizes $\{128, 256\}$ for synthetic-data experiments and filter sizes $\{128, 256, 256, 512\}$ for image completion and point-cloud generation. The 1D-convolution layers are interleaved with ReLU, followed by max-pooling across inputs in the embedding space. We then use the reparametrization trick (Kingma & Welling, 2013) to sample θ from the resulting Gaussian. To compute the energy of a single input $f(x_i; \theta)$, each x_i is concatenated with θ and followed by fully connected layers of $\{128, 64, 1\}$ neurons each interleaved with ReLU. The energy of the set, $f(x_{1:n}; \theta)$, is the average energy of all inputs in the set.

To obtain the ADE initialization distribution $q^0\left(x^0_{1:n}|\theta\right)$ on synthetic data, we first use a 2-layer hyper-network (Ha et al., 2017) with 256 hidden neurons each to output the parameters for a 10-layer normalizing planar flow model (Rezende & Mohamed, 2015). The planar flow then takes both θ (learned from T_{train} and X_{train}) and the target indices T_{test} as inputs to produce $q^0\left(x^0_{t1:t_n}|\theta\right)$. For the synthetic experiments, we directly use $q^0\left(x^0_{1:n}|\theta\right)$ as the sampler output without performing additional Hamiltonian/Langevin dynamics, which is sufficient to capture the synthetic data distributions.

To obtain $q\left(x_{1:n}, v|\theta\right)$ for image completion and point-cloud generation, we first parametrize the initialization distribution $q^0\left(x^0_{1:n}|\theta\right)$ using an RNN where each recurrent LSTM block consists of an MLP with $\{64, 128, 512\}$ hidden neurons interleaved with ReLU. Each LSTM block outputs the mean and variance of a diagonal Gaussian of dimension k times d, where k (block size) is the number of elements generated at once, and d is the dimension of each element. For image completion, k equals the number of pixels in a row of an image (*e.g.*, 28 for MNIST and 32 for CelebA), and d equals the number of channels of a pixel (1 for MNIST and 3 for CelebA). For point-cloud generation, we have $k = 512$ and $d = 3$. We considered a range of k values from 32 to 2048, and selected k based on the best completion/generation performance. The number of recurrent RNN blocks equals the total number of elements in the final generated set (784 for MNIST, 1024 for CelebA, and 2048 for point clouds) divided by k. Using the RNN output as the initialization distribution $q^0\left(x^0_{1:n}|\theta\right)$, we then perform $T = 20$ steps of Langevin dynamics similar to Xie et al. (2016) with step size $\eta = 0.1$ and $\xi_i \sim \mathcal{N}\left(0, 0.05\right)$

while clipping $\nabla_x f$ to 0.1.

We use spectral normalization in training the energy function and batch normalization in training the sampler. We use Adam with learning rate 10^{-4} to optimize all of our models. We set $\beta_1 = 0.5$ for image completion and $\beta_1 = 0.0$ for the synthetic experiments and for point-cloud generation. The coefficient of $H(q(x_{1:n}, v|\theta))$ in (1) is set to 10^{-5}. All tasks are trained until convergence using batch size 64 on a single NVIDIA V100 GPU with 32 GB memory.

F.2. Point-Cloud Preprocessing

Following Achlioptas et al. (2017), we use shapes from ShapeNet (Wu et al., 2015) that are axis aligned and centered into the unit sphere. For the point-cloud classification task, we apply random rotations along the gravity axis following Achlioptas et al. (2017); Yang et al. (2019), and also normalize shapes from ModelNet40 (Wu et al., 2015) the same way. For generation, the model is trained on the official training split and evaluated on the test split of ShapeNet (where the train-validation-test splits are 70%-20%-10%) similar to Yang et al. (2019).

F.3. Point-Cloud Model Selection

We follow the model selection protocol of Achlioptas et al. (2017), namely select the model with the smallest JSD and reports other measurements of this model. Measurements are created every 100 epochs. We noticed that the measurement of JSD and COV are relatively robust across different evaluation runs, whereas the MMD measurement (due to its small magnitude, *e.g.*, 10^{-4}) is less robust.

F.4. Point-Cloud Generation Metrics

Following Achlioptas et al. (2017); Yang et al. (2019), we use Chamfer distance (CD) and earth mover's distance (EMD) defined below to measure distance between point clouds.

$$d_{CD}(X, X') = \sum_{x_i \in X} \min_{x'_i \in X'} \|x_i - x'_i\|_2 + \sum_{x'_i \in X'} \min_{x_i \in X} \|x_i - x'_i\|_2, \tag{F.1}$$

$$d_{EMD}(X, X') = \min_{\phi: X \to X'} \sum_{x_i \in X} \|x_i - \phi(x_i)\|_2. \tag{F.2}$$

Similarly, we use Jensen-Shannon Divergence (JSD), Minimum matching distance (MMD), and Coverage (COV) defined below to evaluate generation quallity.

- JSD is computed between the marginal distribution of the entire reference set (p_r) and the marginal distribution of the entire generated set (p_g) of point clouds, specifically,

$$\text{JSD}(p_g, p_r) = \frac{1}{2} D_{KL}(p_r || p_m) + \frac{1}{2} D_{KL}(p_g || p_m)\,, \tag{F.3}$$

  where $p_m = \frac{1}{2}(p_r + p_g)$ and D_{KL} stands for the Kullback-Leibler divergence. JSD only measures similarity at the marginal distribution level, and does not provide insights on the generation quality of each individual point cloud.

- MMD measures the average distance between a point-cloud in the reference set S_r and its closest neighbor in the generated set S_g, namely

$$\text{MMD}(S_g, S_r) = \frac{1}{|S_r|} \sum_{X \in S_r} \min_{X' \in S_g} d(X, X'), \tag{F.4}$$

  where $d(X, X')$ is the distance between two point clouds according to either CD or EMD.

- COV measures the ratio of point clouds in the reference set S_r that are matched to a distinct closest neighbor in the the generated set S_g:

$$\text{COV}(S_g, S_r) = \frac{1}{|S_r|} |\{\arg\min_{X \in S_r} d(X, X') | X' \in S_g\}|. \tag{F.5}$$

  where $d(X, X')$ can again be based on CD or EMD. COV reflects the generation diversity.

G. More Experimental Results

G.1. Additional Comparisons on Synthetic Data

We comapre the proposed $\mathcal{EBP}$s with Gaussian processes ($\mathcal{GP}$s), neural processes ($\mathcal{NP}$s)[4], and variational implicit processes ($\mathcal{VIP}$s)[5].

Figure 7 shows the ground truth data and the normalized probability heatmap for each process. To generate data given an index t_i, we randomly select one of the two modes (*e.g.*, one of the sine waves) as the mean and adds $\epsilon \sim \mathcal{N}(0, 0.1)$ noise to produce x_{t_i}. We plot the heatmap of the learned predictive distribution of each process. For $\mathcal{EBP}$, we use (D.15) to approximate $p(x_{t_i}|t_i)$.

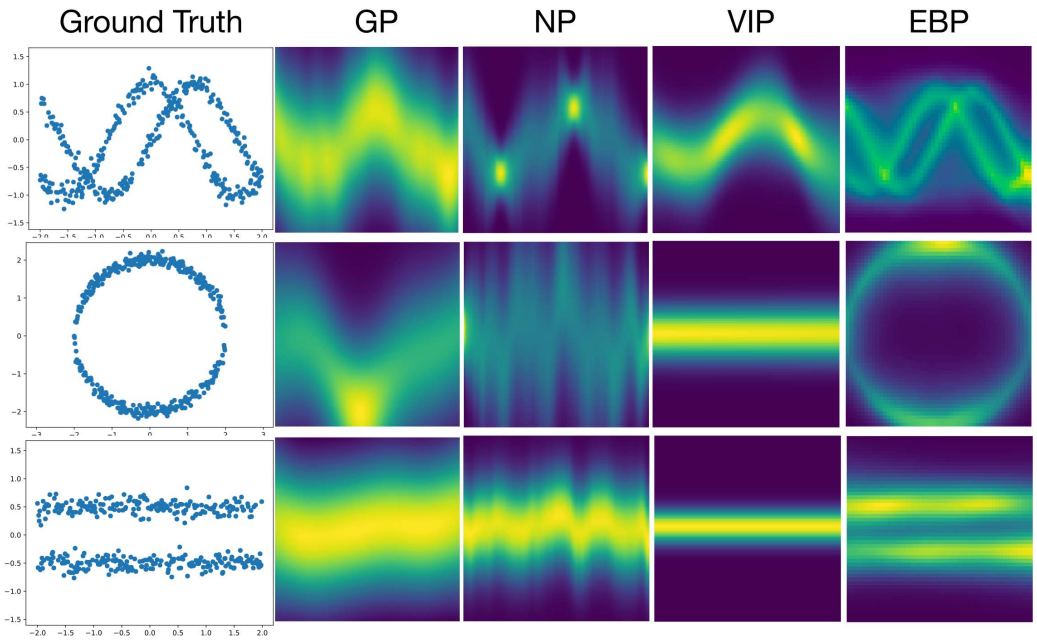

Figure 7. The ground truth data and learned energy functions of $\mathcal{GP}$, $\mathcal{NP}$, $\mathcal{VIP}$, and $\mathcal{EBP}$ (from left to right). $\mathcal{EBP}$ successfully captures multi-modality of the toy data as $\mathcal{GP}$, $\mathcal{NP}$, and $\mathcal{VIP}$ exhibiting only a single mode.

[4] https://github.com/deepmind/neural-processes
[5] https://github.com/LaurantChao/VIP

G.2. Additional Image Completion Results on MNIST

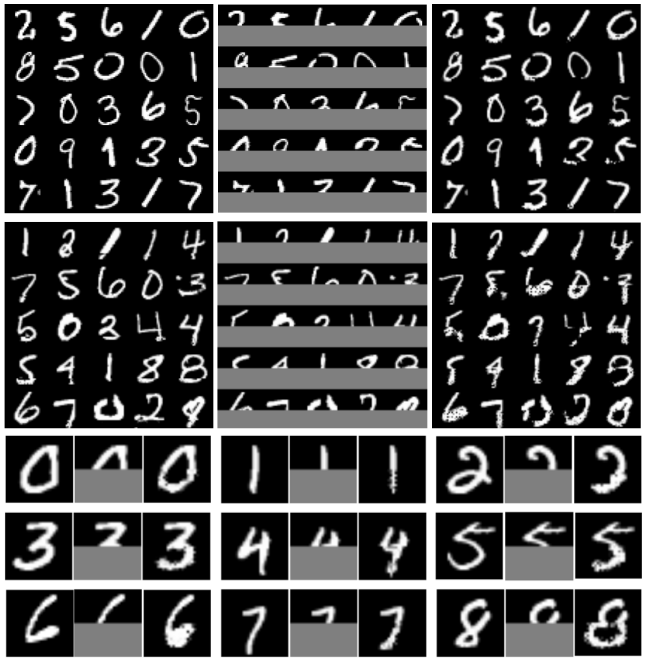

Figure 8. Additional image completion results on MNIST where the top half of the image serves as context.

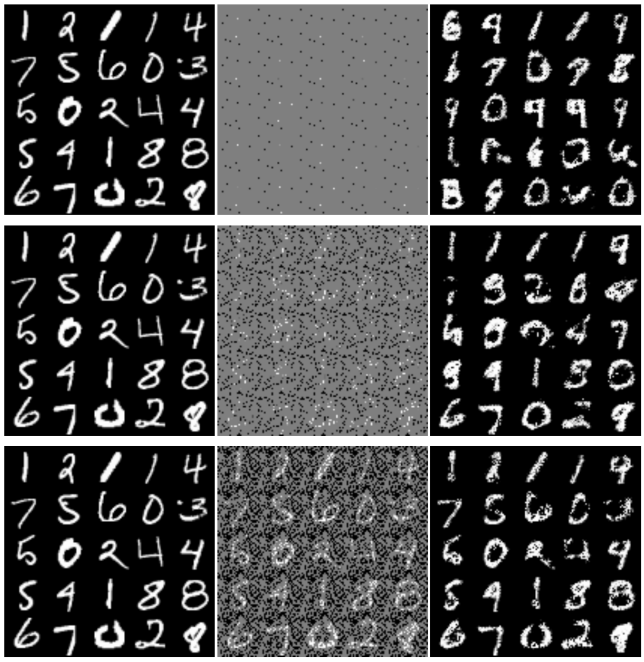

Figure 9. Additional image completion results on MNIST where 10, 100, and 1000 (top to bottom) random pixels serve as context.

Figure 8 shows additional image completion results on MNIST when only the upper-half of an image is given. Figure 9 shows MNIST completion results when 10, 100, and 1000 randomly selected pixels are given.

G.3. Additional Image Completion Results on CelebA

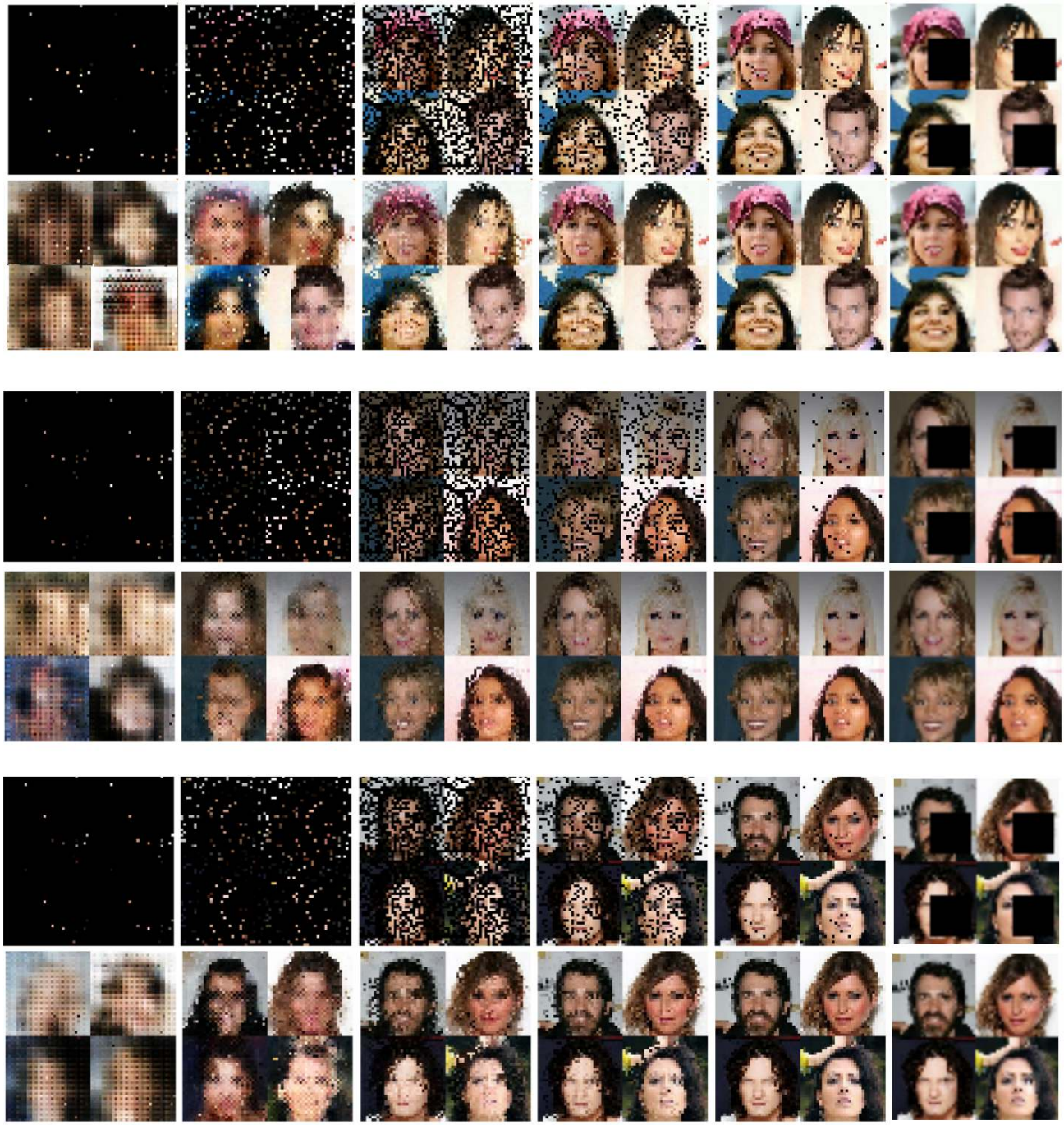

Figure 10. Additional image completion results on CelebA for both contiguous (right-most) and random pixels as context.

Figure 10 shows additional completion results on CelebA where $10, 100, 500, 800, 1000$ randomly selected pixels are given and when a 16x16 square is removed from the original 32x32 image.

G.4. Additional Point-Cloud Generation Restuls

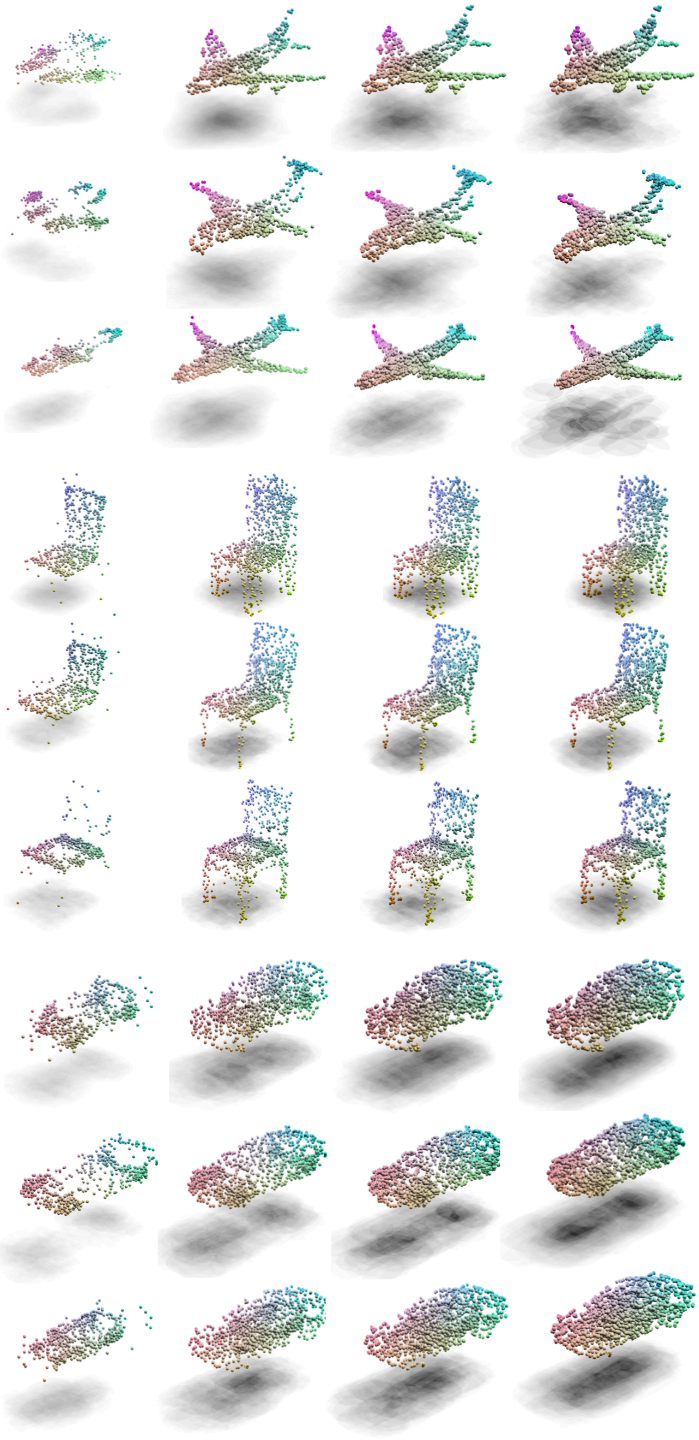

Figure 11. Additional examples of airplane, chair, and car point-cloud generation with 4 RNN blocks of block size 512.

G.5. Additional Point-Cloud Denoising Restuls

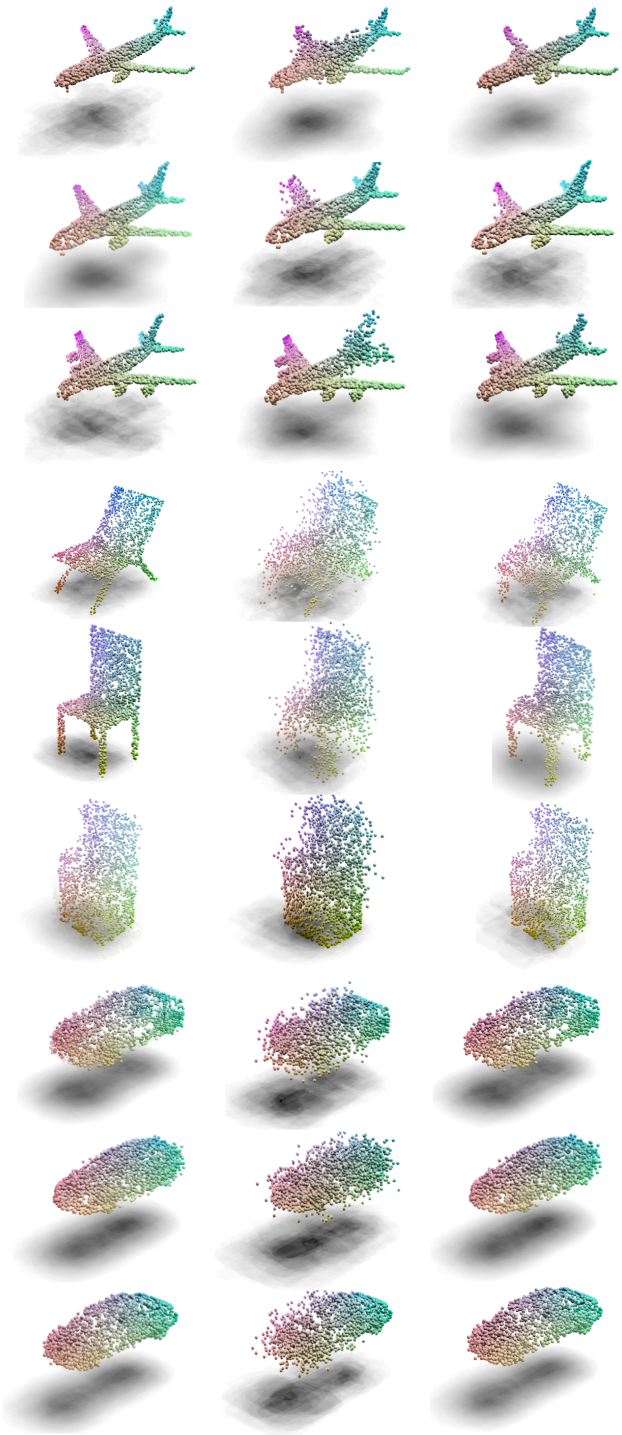

Figure 12. Additional examples of airplane, chair, and car point-cloud denoising. Left to right: original, perturbed, denoised point clouds.