A. Implementation details

In this section, we discuss detailed derivations and descriptions of SetVAE presented in Section 3 and Section 4.

A.1. KL Divergence of Initial Set Distribution

This section provides proof that the KL divergence between the approximate posterior and the prior of the initial elements in Eq. (10) and (13) is a constant.

Following the definition in Eq. (8) and Eq. (9), we decompose the prior as $p(\mathbf{z}^{(0)}) = p(n)p(\mathbf{z}^{(0)}|n)$ and the approximate posterior as $q(\mathbf{z}^{(0)}|\mathbf{x}) = \delta(n)q(\mathbf{z}^{(0)}|n, \mathbf{x})$, where $\delta(n)$ is defined as a delta function centered at $n = |\mathbf{x}|$. Here, the conditionals are given by

$$p(\mathbf{z}^{(0)}|n) = \prod_{i=1}^{n} p(\mathbf{z}_i^{(0)}), \tag{26}$$

$$q(\mathbf{z}^{(0)}|n, \mathbf{x}) = \prod_{i=1}^{n} q(\mathbf{z}_i^{(0)}|\mathbf{x}). \tag{27}$$

As described in the main text, we set the elementwise distributions identical, $p(\mathbf{z}_i^{(0)}) = q(\mathbf{z}_i^{(0)}|\mathbf{x})$. This renders the conditionals equal,

$$p(\mathbf{z}^{(0)}|n) = q(\mathbf{z}^{(0)}|n, \mathbf{x}). \tag{28}$$

Then, the KL divergence between the approximate posterior and the prior in Eq. (10) is written as

$$\begin{aligned}
&\mathrm{KL}(q(\mathbf{z}^{(0)}|\mathbf{x})||p(\mathbf{z}^{(0)})) \\
&\quad = \mathrm{KL}(\delta(n)q(\mathbf{z}^{(0)}|n, \mathbf{x})||p(n)p(\mathbf{z}^{(0)}|n)) \qquad (29) \\
&\quad = \mathrm{KL}(\delta(n)p(\mathbf{z}^{(0)}|n)||p(n)p(\mathbf{z}^{(0)}|n)), \qquad (30)
\end{aligned}$$

where the second equality comes from the Eq. (28). From the definition of KL divergence, we can rewrite Eq. (30) as

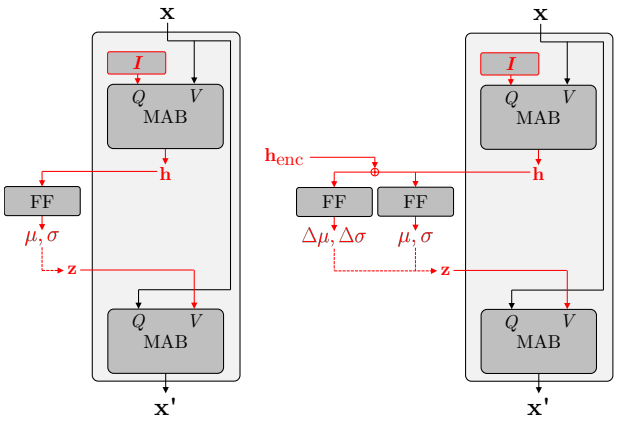
$$\begin{aligned}
&\mathrm{KL}(q(\mathbf{z}^{(0)}|\mathbf{x})||p(\mathbf{z}^{(0)})) \\
&\quad = \mathbb{E}_{\delta(n)p(\mathbf{z}^{(0)}|n)}\left[\log \frac{\delta(n)p(\mathbf{z}^{(0)}|n)}{p(n)p(\mathbf{z}^{(0)}|n)}\right] \qquad (31) \\
&\quad = \mathbb{E}_{\delta(n)p(\mathbf{z}^{(0)}|n)}\left[\log \frac{\delta(n)}{p(n)}\right], \qquad (32)
\end{aligned}$$

As the logarithm in Eq. (32) does not depend on $\mathbf{z}^{(0)}$, we can take it out from the expectation over $p(\mathbf{z}^{(0)}|n)$ as follows:

$$\begin{aligned}
&\mathrm{KL}(q(\mathbf{z}^{(0)}|\mathbf{x})||p(\mathbf{z}^{(0)})) \\
&\quad = \mathbb{E}_{\delta(n)}\left[\mathbb{E}_{p(\mathbf{z}^{(0)}|n)}\left[\log \frac{\delta(n)}{p(n)}\right]\right] \qquad (33) \\
&\quad = \mathbb{E}_{\delta(n)}\left[\log \frac{\delta(n)}{p(n)}\right], \qquad (34)
\end{aligned}$$

which can be rewritten as

$$\mathbb{E}_{\delta(n)}[\log \delta(n) - \log p(n)]. \tag{35}$$



(a) Generation (b) Inference

Figure 11: The detailed structure of Attentive Bottleneck Layer during sampling (for generation) and inference (for reconstruction).

The expectation over the delta function $\delta(n)$ is simply an evaluation at $n = |\mathbf{x}|$. As δ is defined over a discrete random variable n, its probability mass at the center $|\mathbf{x}|$ equals 1. Therefore, $\log \delta(n)$ at $n = |\mathbf{x}|$ reduces to $\log 1 = 0$, and we obtain

$$\mathrm{KL}(q(\mathbf{z}^{(0)}|\mathbf{x})||p(\mathbf{z}^{(0)})) = -\log p(|\mathbf{x}|). \tag{36}$$

As discussed in the main text, we model $p(n)$ using the empirical distribution of data cardinality. Thus, $p(|\mathbf{x}|)$ only depends on data distribution, and $-\mathrm{KL}(q(\mathbf{z}^{(0)}|\mathbf{x})||p(\mathbf{z}^{(0)}))$ in Eq. (10) can be omitted from optimization.

A.2. Implementation of SetVAE

Attentive Bottleneck Layer. In Figure 11, we provide the detailed structure of Attentive Bottleneck Layer (ABL) that composes the top-down generator of SetVAE (Section 4). We share the parameters in ABL for generation and inference, which is known to be effective in stabilizing the training of hierarchical VAE [14, 26].

During generation (Figure 11a), $\mathbf{z}$ is sampled from a Gaussian prior $\mathcal{N}(\mu, \sigma)$ (Eq. (19)). To predict μ and σ from $\mathbf{h}$, we use an elementwise fully-connected (FC) layer, of which parameters are shared across elements of $\mathbf{h}$. During inference, we sample the latent variables from the approximate posterior $\mathcal{N}(\mu + \Delta\mu, \sigma \cdot \Delta\sigma)$, where the correction factors $\Delta\mu, \Delta\sigma$ are predicted from the bottom-up encoding $\mathbf{h}_{\text{enc}}$ by an additional FC layer. Note that the FC for predicting μ, σ is shared for generation and inference, but the FC that predicts $\Delta\mu, \Delta\sigma$ is used only for inference.

Slot Attention in ISAB and ABL. SetVAE discovers subset representation via projection attention in ISAB (Eq. (6)) and ABL (Eq. (18)). However, with a basic attention mechanism, the projection attention may ignore some parts of input by simply not attending to them. To prevent this, in both ISAB and ABL, we change the projection attention to Slot Attention [21].

Specifically, plain projection attention[4] (Eq. (4)) treats input $\mathbf{x} \in \mathbb{R}^{n \times d}$ as key (K) and value (V), and uses a set of inducing points $\mathbf{I} \in \mathbb{R}^{m \times d}$ as query (Q). First, it obtains the attention score matrix as follows:

$$A = \frac{QK^{\mathrm{T}}}{\sqrt{d}} \in \mathbb{R}^{m \times n}. \tag{37}$$

Each row index of A denotes an inducing point, and each column index denotes an input element. Then, the value set V is aggregated using A. With $\mathrm{Softmax}_{\mathrm{axis}=d}(\cdot)$ denoting softmax normalization along d-th axis, the plain attention normalizes each row of A, as follows:

$$\mathrm{Att}(Q, K, V) = WV \in \mathbb{R}^{m \times d}, \tag{38}$$

$$\text{where } W = \mathrm{Softmax}_{\mathrm{axis}=2}(A) \in \mathbb{R}^{m \times n}. \tag{39}$$

As a result, an input element can get zero attention if every query suppresses it. To prevent this, Slot Attention normalizes each column of A and computes weighted mean:

$$\mathrm{SlotAtt}(Q, K, V) = W'V, \tag{40}$$

$$\text{where } W'_{ij} = \frac{A'_{ij}}{\sum_{l=1}^{n} A'_{il}} \text{ for } A' = \mathrm{Softmax}_{\mathrm{axis}=1}(A). \tag{41}$$

As attention coefficients across a row sum up to 1 after softmax, slot attention guarantees that an input element is not ignored by every inducing point.

With the adaptation of Slot Attention, we observe that inducing points often attend to distinct subsets of the input to produce $\mathbf{h}$, as illustrated in the Figure 7 and Figure 8 of the main text. This is similar to the observation of [21] that the competition across queries encouraged segmented representations (slots) of objects from a multi-object image. A difference is that unlike in [21] where the queries are noise vectors, we design the query set as a learnable parameter $\mathbf{I}$. Also, we do not introduce any refinement steps to the projected set $\mathbf{h}$ to avoid the complication of the model.

B. Experiment Details

This section discusses the detailed descriptions and additional results of experiments in Section 6 in the main paper.

[4]For simplicity, we explain with single-head attention instead of MultiHead.

B.1. ShapeNet Evaluation Metrics

We provide descriptions of evaluation metrics used in the ShapeNet experiment (Section 6 in the main paper). We measure standard metrics including coverage (COV), minimum matching distance (MMD), and 1-nearest neighbor accuracy (1-NNA) [1, 30]. Following recent literature [13], we omit Jensen-Shannon Divergence (JSD) [1] because it does not assess the fidelity of each point cloud. To measure the similarity $D(\mathbf{x}, \mathbf{y})$ between point clouds $\mathbf{x}$ and $\mathbf{y}$, we use Chamfer Distance (CD) (Eq. (24)) and Earth Mover's Distance (EMD), where the EMD is defined as:

$$\mathrm{EMD}(\mathbf{x}, \mathbf{y}) = \min_{\pi} \sum_{i} \|\mathbf{x}_i - \mathbf{y}_{\pi(i)}\|_2. \tag{42}$$

Let S_g be the set of generated point clouds and S_r be the set of reference point clouds with $|S_r| = |S_g|$.

Coverage (COV) measures the percentage of reference point clouds that is a nearest neighbor of at least one generated point cloud, computed as follows:

$$\mathrm{COV}(S_g, S_r) = \frac{|\{\mathrm{argmin}_{\mathbf{y} \in S_r} D(\mathbf{x}, \mathbf{y}) | \mathbf{x} \in S_g\}|}{|S_r|}. \tag{43}$$

Minimum Matching Distance (MMD) measures the average distance from each reference point cloud to its nearest neighbor in the generated point clouds:

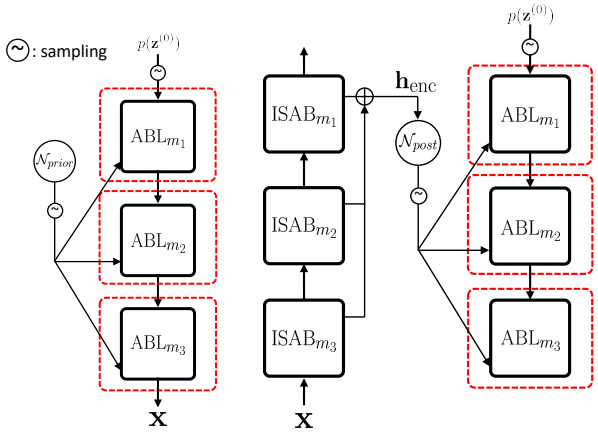
$$\mathrm{MMD}(S_g, S_r) = \frac{1}{|S_r|} \sum_{\mathbf{y} \in S_r} \min_{\mathbf{x} \in S_g} D(\mathbf{x}, \mathbf{y}). \tag{44}$$

1-Nearest Neighbor Accuracy (1-NNA) assesses whether two distributions are identical. Let $S_{-\mathbf{x}} = S_r \cup S_g - \{\mathbf{x}\}$ and $N_{\mathbf{x}}$ be the nearest neighbor of $\mathbf{x}$ in $S_{-\mathbf{x}}$. With $\mathbf{1}(\cdot)$ an indicator function:

$$\text{1-NNA}(S_g, S_r) = \frac{\sum_{\mathbf{x} \in S_g} \mathbf{1}(N_{\mathbf{x}} \in S_g) + \sum_{\mathbf{y} \in S_r} \mathbf{1}(N_{\mathbf{y}} \in S_r)}{|S_g| + |S_r|}. \tag{45}$$

B.2. Hierarchical Disentanglement

This section describes an evaluation protocol used in Figure 9 in the main paper. To investigate the latent representations learned at each level, we employed Linear Discriminant Analysis (LDA) as simple layer-wise classifiers. The classifiers take the latent variable at each layer $\mathbf{z}^l$, $\forall l \in [1, L]$ as an input, and predict the identity and position of two digits (in 4×4 quantized grid) respectively in Set-MultiMNIST dataset. To this end, we first train the SetVAE in the training set of Set-MultiMNIST. Then we train the LDA classifiers using the validation dataset, where the input latent variables are sampled from the posterior distribution of SetVAE (Eq. (12)). We report the training accuracy of the classifiers at each layer in Figure 9.



(a) Generation (b) Inference

Figure 12: Structure of Vanilla SetVAE without hierarchical priors and subset reasoning in generator.

B.3. Ablation study

In this section, we provide details of the ablation study presented in Table 3 of the main text.

Baseline As baselines, we use a SetVAE with unimodal Gaussian prior over the initial set elements, and a non-hierarchical, Vanilla SetVAE presented in Section 3.

To implement a SetVAE with unimodal prior, we only change the initial element distribution $p(\mathbf{z}_i^{(0)})$ from MoG (Eq. (21)) to a multivariate Gaussian with a diagonal covariance matrix $\mathcal{N}(\mu^{(0)}, \sigma^{(0)})$ with learnable $\mu^{(0)}$ and $\sigma^{(0)}$. This approach is adopted in several previous works in permutation-equivariant set generation [30, 16, 21].

To implement a Vanilla SetVAE, we employ a bottom-up encoder same to our full model and make the following changes to the top-down generator. As illustrated in Figure 12, we remove the subset relation in the generator by fixing the latent cardinality to 1 and employing a global prior $\mathcal{N}(\mu_1, \sigma_1)$ with $\mu_1, \sigma_1 \in \mathbb{R}^{1\times d}$ for all ABL. To compute permutation-invariant $\mathbf{h}_{\text{enc}} \in \mathbb{R}^{1\times d}$, we aggregate every elements of $\mathbf{h}$ from all levels of encoder network by average pooling. During inference, $\mathbf{h}_{\text{enc}}$ is provided to every ABL in the top-down generator.

Evaluation metric For the ablation study of SetVAE on the Set-MultiMNIST dataset, we measure the generation quality in image space by rendering each set instance to 64×64 binary image based on the occurrence of a point in a pixel bin. To measure the generation performance, we compute Frechet Inception Distance (FID) score [9] using the output of the penultimate layer of a VGG11 network trained from scratch for MultiMNIST image classification into 100 labels (00-99). Given the channel-wise mean μ_g, μ_r and covariance matrix Σ_g, Σ_r of generated and reference set of images respectively, we compute FID as follows:

$$d^2 = \|\mu_g - \mu_r\|^2 + \text{Tr}(\Sigma_g + \Sigma_r - 2\sqrt{\Sigma_g\Sigma_r}). \quad (46)$$

To train the VGG11 network, we replace the first conv layer to take single-channel inputs, and use the same MultiMNIST train set as in SetVAE. We use the SGD optimizer with Nesterov momentum, with learning rate 0.01, momentum 0.9, and L2 regularization weight 5e-3 to prevent overfitting. We train the network for 10 epochs using batch size 128 so that the training top-1 accuracy exceeds 95%.

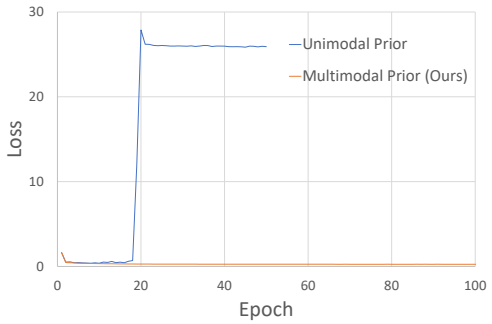


Figure 13: Training loss curves from SetVAE with multimodal and unimodal initial set trained on Set-MultiMNIST dataset.

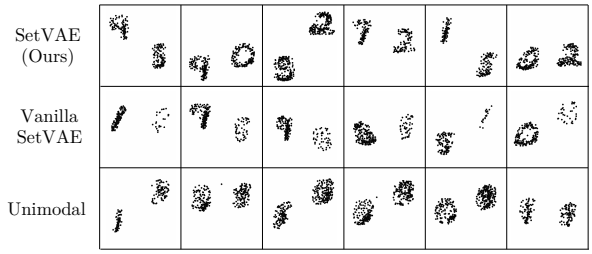


Figure 14: Samples from SetVAE and its ablated version trained on Set-MultiMNIST dataset.

C. More Qualitative Results

Ablation study This section provides additional results of the ablation study, which corresponds to the Table 3 of the main paper. We compare the SetVAE with two baselines: SetVAE with a unimodal prior and the one using a single global latent variable (*i.e.*, Vanilla SetVAE).

Figure 13 shows the training loss curves of SetVAE and the unimodal prior baseline on the Set-MultiMNIST dataset. We observe that training of the unimodal baseline is unstable compared to SetVAE which uses a 4-component MoG. We conjecture that a flexible initial set distribution provides a cue for the generator to learn stable subset representations.

In Figure 14, we visualize samples from SetVAE and the two baselines. As the training of unimodal SetVAE was un-

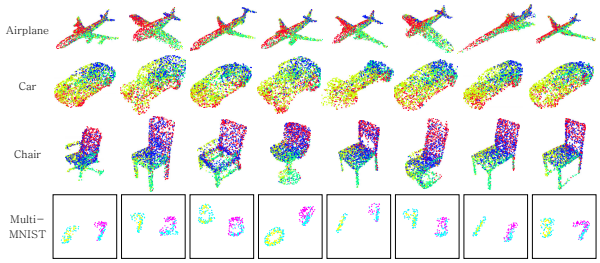


Figure 15: Color-coded mixture assignments on output sets.

stable, we provide the results from a checkpoint before the training loss diverges (third row of Figure 14). The Vanilla SetVAE without hierarchy (second row of Figure 14) focuses on modeling the left digit only and fails to assign a balanced number of points. This failure implies that multi-level subset reasoning in the generative process is essential in faithfully modeling complex set data such as Set-MultiMNIST.

Role of mixture initial set Although the multi-modal prior is not a typical choice, we emphasize that it marginally adds complexity to the model since it only introduces the additional learnable mixture parameters $(\pi_k^{(0)}, \mu_k^{(0)}, \sigma_k^{(0)})$. Despite the simplicity, we observe that mixture prior is effective in stabilizing training, especially when there are clearly separating modes in data such as in the Set-MultiMNIST dataset (Fig. 15). Still, the choice of the initial prior is flexible and orthogonal to our contribution.

ShapeNet results Figure 16 presents the generated samples from SetVAE on ShapeNet, Set-MNIST and Set-MultiMNIST datasets, extending the results in Figure 4 of the main text. As illustrated in the figure, SetVAE generates point sets with high diversity while capturing thin and sharp details (*e.g.* engines of an airplane and legs of a chair, *etc.*).

Cardinality disentanglement Figure 17 presents the additional results of Figure 5 in the main paper, which illustrates samples generated by increasing the cardinality of the initial set $\mathbf{z}^{(0)}$ while fixing the hierarchical latent variables $\mathbf{z}^{(1:L)}$. As illustrated in the figure, SetVAE is able to disentangle the cardinality of a set from the rest of its generative factors, and is able to generalize to unseen cardinality while preserving the disentanglement.

Notably, SetVAE can retain the disentanglement and generalize even to a high cardinality (100k) as well. Figure 18 presents the comparison to PointFlow with varying cardinality, which extends the results of the Figure 6 in the main paper. Unlike PointFlow that exhibits degradation and blurring of fine details, SetVAE retains the fine structure of the generated set even for extreme cardinality.

Coarse-to-fine dependency In Figure 19 and Figure 20, we provide additional visualization of encoder and generator attention, extending the Figure 7 and Figure 8 in the main text. We observe that SetVAE learns to attend to a subset of points consistently across examples. Notably, these subsets often have a bilaterally symmetric structure or correspond to semantic parts. For example, in the top level of the encoder (rows marked level 1 in Figure 19), the subsets include wings of an airplane, legs & back of a chair, or wheels & rear wing of a car (colored red).

Furthermore, SetVAE extends the subset modeling to multiple levels with a top-down increase in latent cardinality. This allows SetVAE to encode or generate the structure of a set in various granularities, ranging from global structure to fine details. Each column in Figure 19 and Figure 20 illustrates the relations. For example, in level 3 of Figure 19, the bottom-up encoder partitions an airplane into fine-grained parts such as an engine, a tip of the tail wing, *etc*. Then, going bottom-up to level 1, the encoder composes them to fuselage and symmetric pair of wings. As for the top-down generator in Figure 20, it starts in level 1 by composing an airplane via the coarsely defined body and wings. Going top-down to level 3, the generator descends into fine-grained subsets like an engine and tail wing.

D. Architecture and Hyperparameters

Table 4 provides the network architecture and hyperparameters of SetVAE. In the table, $\text{FC}(d, f)$ denotes a fully-connected layer with output dimension d and nonlinearity f. $\text{ISAB}_m(d, h)$ denotes an ISAB_m with m inducing points, hidden dimension d, and h heads (in Section 2.2). $\text{MoG}_K(d)$ denotes a mixture of Gaussian (in Eq. (21)) with K components and dimension d. $\text{ABL}_m(d, d_z, h)$ denotes an ABL_m with m inducing points, hidden dimension d, latent dimension d_z, and h heads (in Section 4). All MABs used in ISAB and ABL uses fully-connected layer with bias as FF layer.

In Table 5, we provide detailed training hyperparameters. For all experiments, we used Adam optimizer with first and second momentum parameters 0.9 and 0.999, respectively and decayed the learning rate linearly to zero after 50% of the training schedule. Following [26], we linearly annealed β from 0 to 1 during the first 2000 epochs for ShapeNet datasets, 40 epochs for Set-MNIST, and 50 epochs for Set-MultiMNIST dataset.

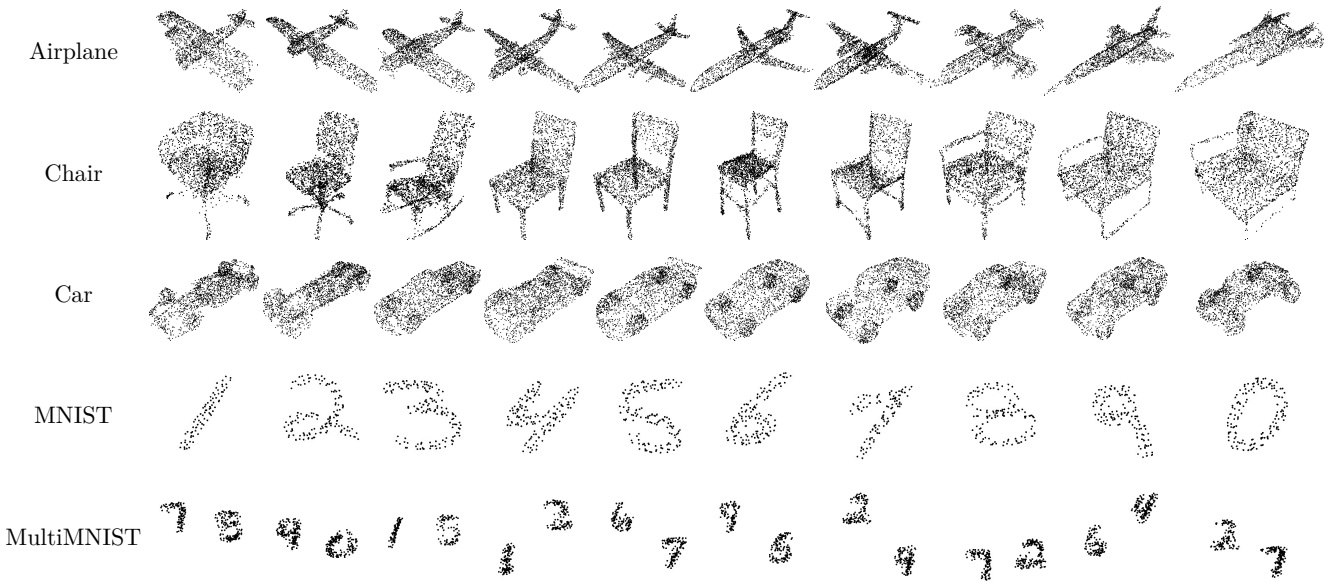


Figure 16: Additional examples of generated point clouds from SetVAE.

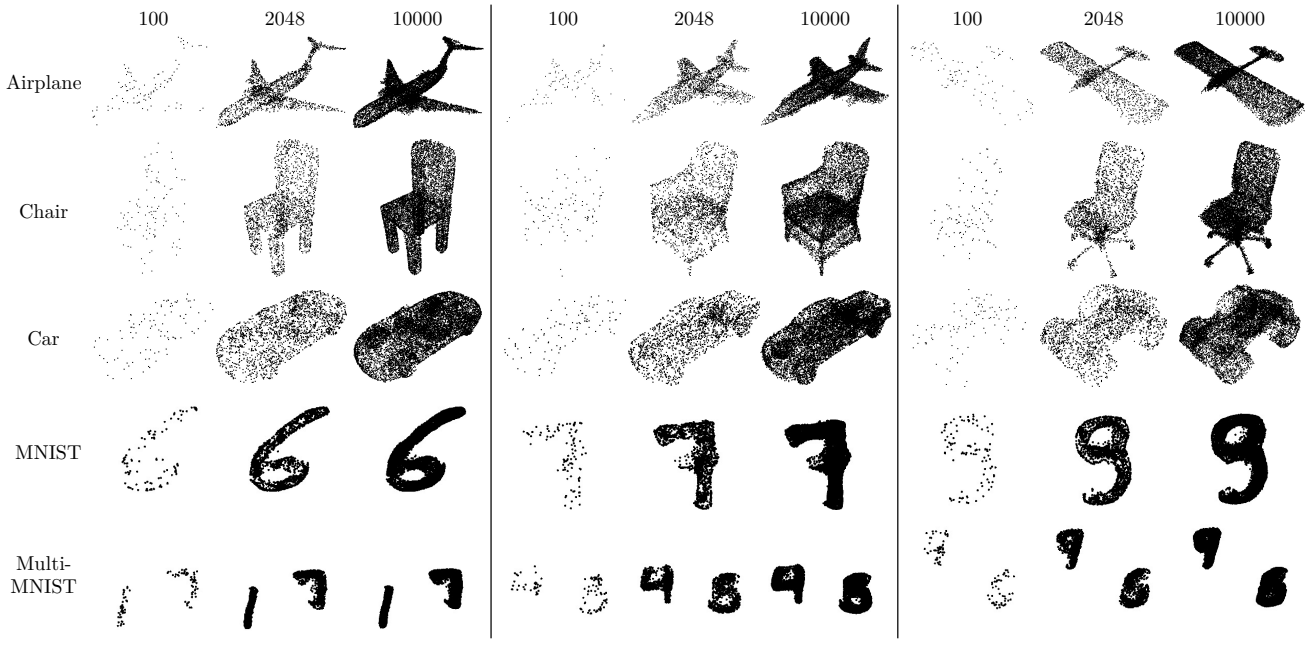


Figure 17: Additional examples demonstrating cardinality generalization of SetVAE.

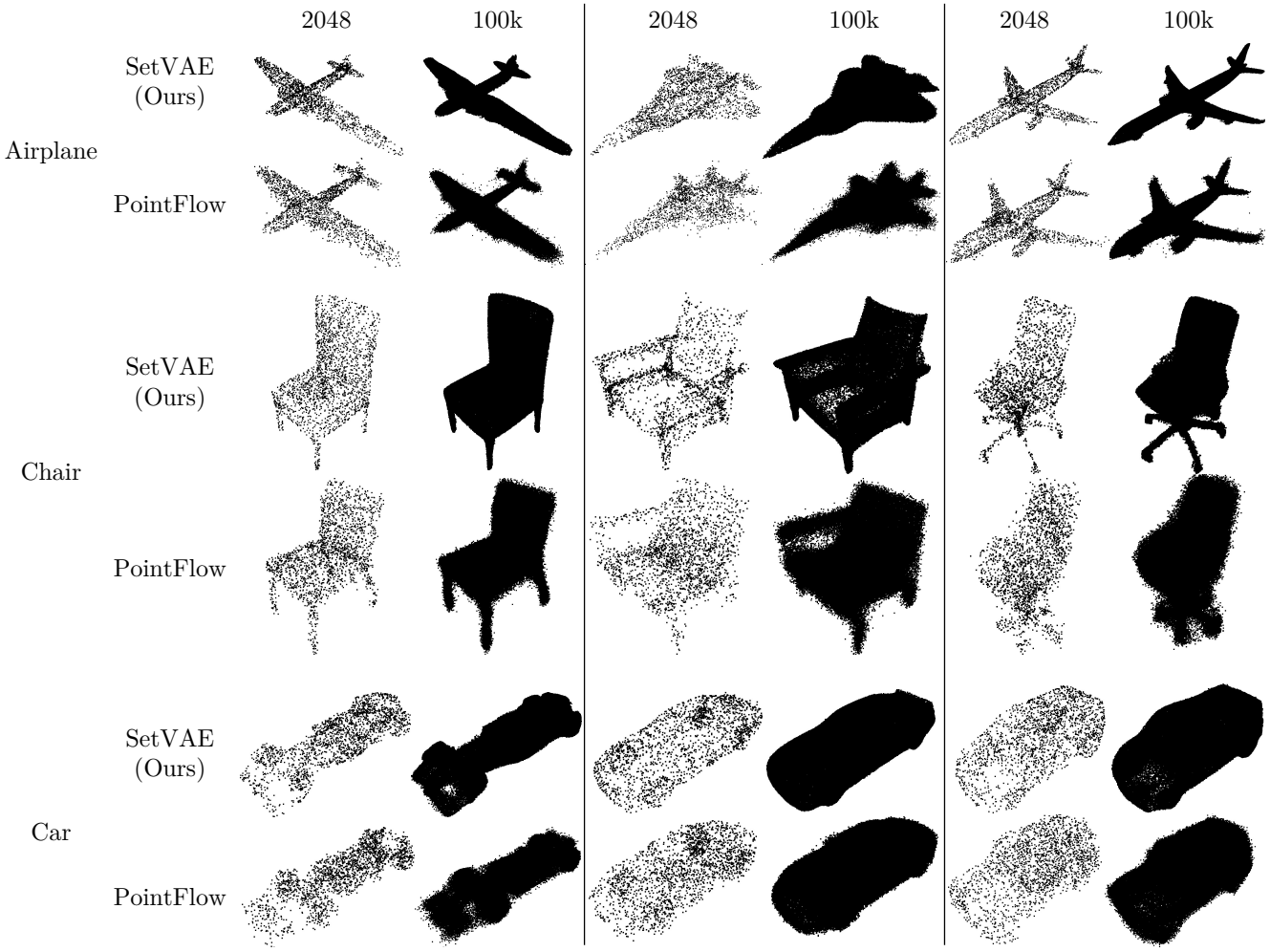


Figure 18: More examples in high-cardinality setting, compared with PointFlow.

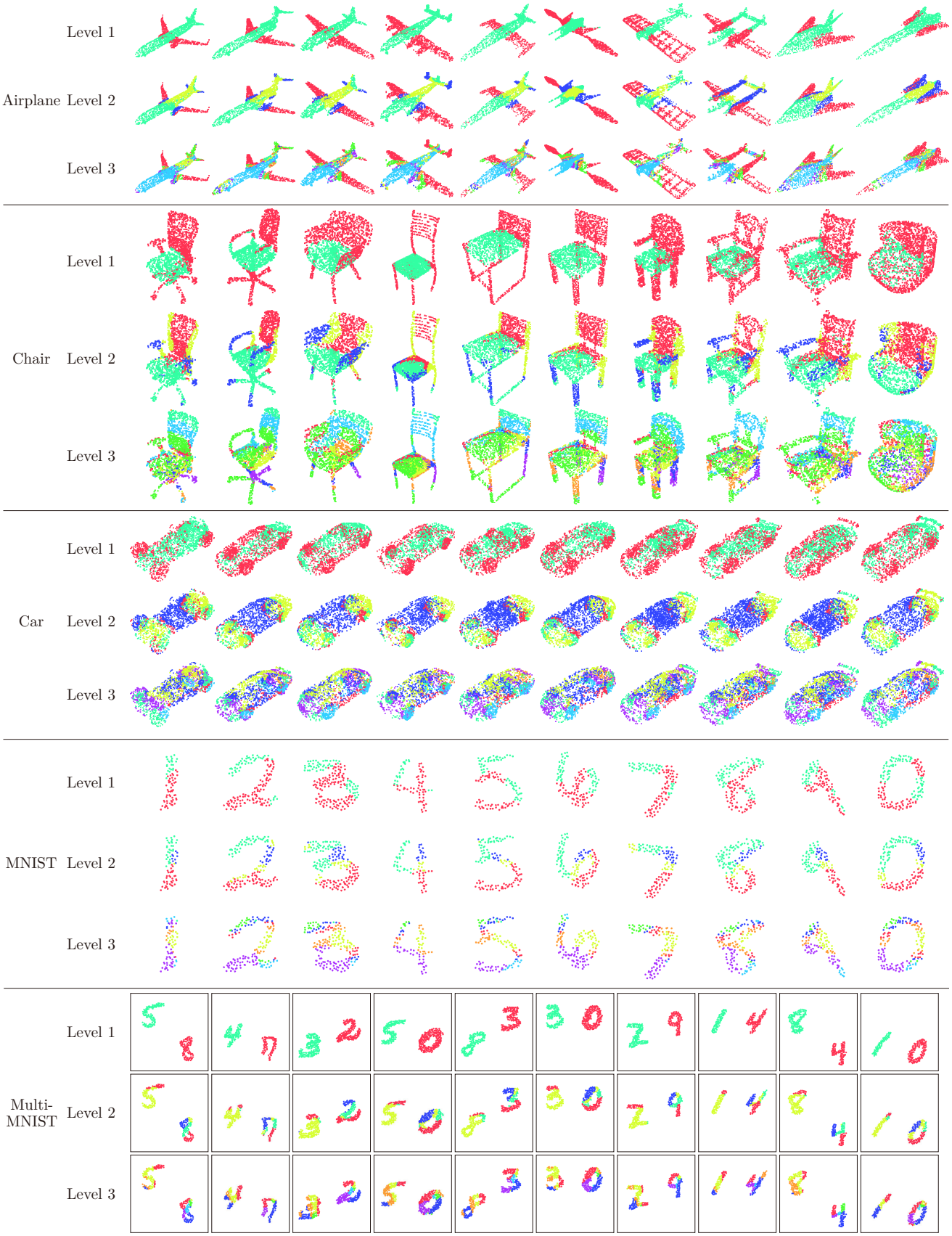


Figure 19: More examples of color-coded encoder attention.

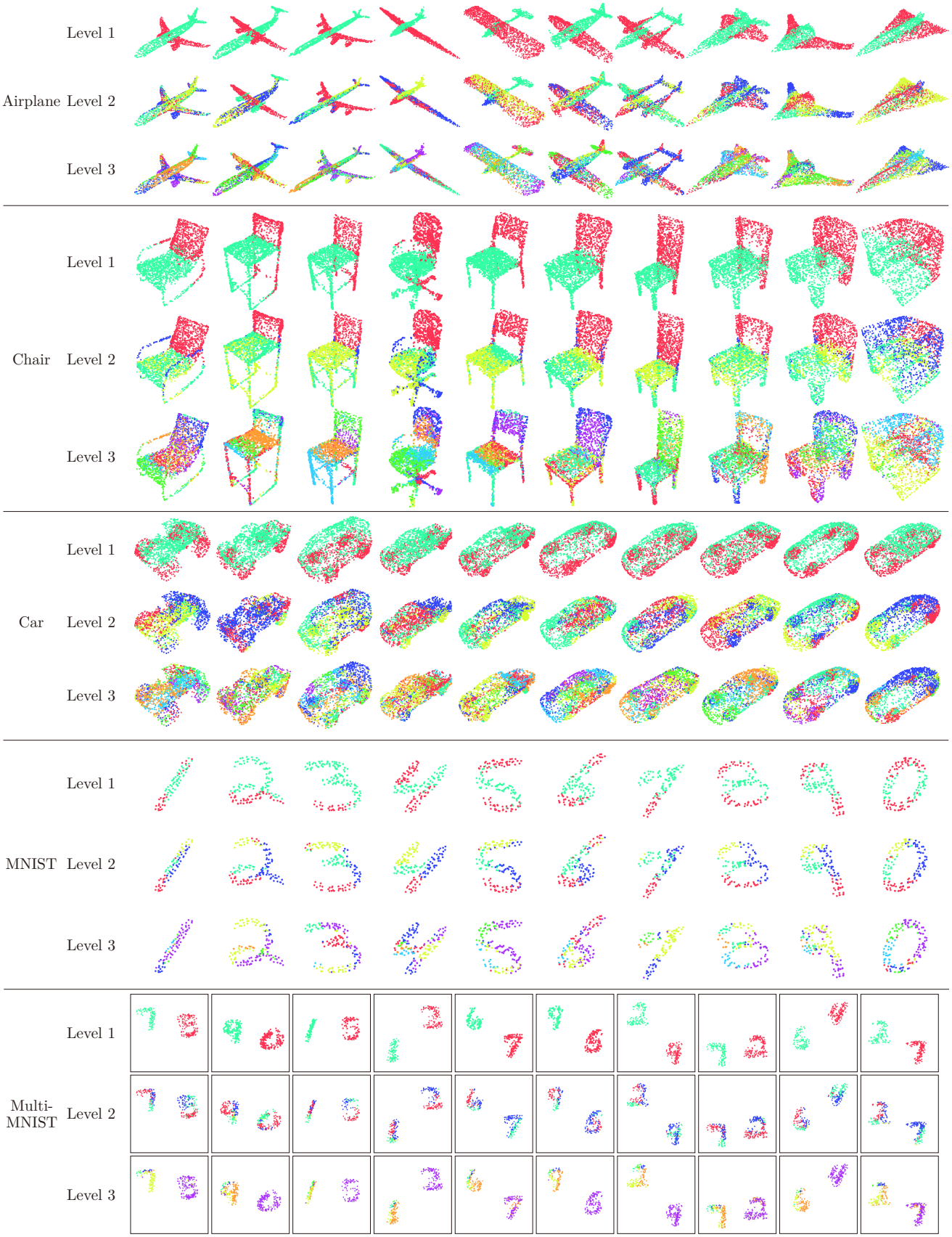


Figure 20: More examples of color-coded generator attention.

Table 4: Detailed network architectures used in our experiments.

ShapeNet		**Set-MNIST**		**Set-MultiMNIST**	
Encoder	**Generator**	**Encoder**	**Generator**	**Encoder**	**Generator**
Input: FC$(64, -)$	Initial set: MoG$_4(32)$	Input: FC$(64, -)$	Initial set: MoG$_4(32)$	Input: FC$(64, -)$	Initial set: MoG$_{16}(64)$
ISAB$_{32}(64, 4)$	ABL$_1(64, 16, 4)$	ISAB$_{32}(64, 4)$	ABL$_2(64, 16, 4)$	ISAB$_{32}(64, 4)$	ABL$_2(64, 16, 4)$
ISAB$_{16}(64, 4)$	ABL$_1(64, 16, 4)$	ISAB$_{16}(64, 4)$	ABL$_4(64, 16, 4)$	ISAB$_{16}(64, 4)$	ABL$_4(64, 16, 4)$
ISAB$_8(64, 4)$	ABL$_2(64, 16, 4)$	ISAB$_8(64, 4)$	ABL$_8(64, 16, 4)$	ISAB$_8(64, 4)$	ABL$_8(64, 16, 4)$
ISAB$_4(64, 4)$	ABL$_4(64, 16, 4)$	ISAB$_4(64, 4)$	ABL$_{16}(64, 16, 4)$	ISAB$_4(64, 4)$	ABL$_{16}(64, 16, 4)$
ISAB$_2(64, 4)$	ABL$_8(64, 16, 4)$	ISAB$_2(64, 4)$	ABL$_{32}(64, 16, 4)$	ISAB$_2(64, 4)$	ABL$_{32}(64, 16, 4)$
ISAB$_1(64, 4)$	ABL$_{16}(64, 16, 4)$		Output: FC$(2, \tanh)$		Output: FC$(2, \tanh)$
ISAB$_1(64, 4)$	ABL$_{32}(64, 16, 4)$		(Output $+ 1)/2$		(Output $+ 1)/2$
	Output: FC$(3, -)$				

Table 5: Detailed training hyperparameters used in our experiments.

	ShapeNet	**Set-MNIST**	**Set-MultiMNIST**
Minibatch size	128	64	64
Training epochs	8000	200	200
Learning rate	1e-3, linear decay to zero after half of training		
β (Eq. (25))	1.0, annealed (-2000epoch)	0.01, annealed (-50epoch)	0.01, annealed (-40epoch)