Supplementary Material

In Section A, we provide more data from our linear SVM experiments (as outlined in Section 4.3) for classification performance under limited supervision. In Section B we derive the theoretical relationship between f_{ψ} and ξ. In Section C, we provide ablation experiments to validate our choice of $J = 32$ in the main paper.

A. Linear SVM with Limited Labels

In this section we provide additional information regarding the Linear SVM experiment from Section 4.3 for the case of limited labeled data. Our setup matches the setup used in [42]: We use a DGCNN backbone pre-trained on ShapeNet. We generate a set of smaller datasets given as a percentage of the original ModelNet40 training split. To generate a dataset, we first randomly sample one labeled point cloud from each class (ensuring there is at least one label of each category). Then we sample the labeled data uniformly randomly across the entire training set until the desired percentage of labeled data is reached with respect to the original dataset size. Note that since ModelNet40 has 9843 training objects, training an SVM on 1% of labels would mean roughly 1-4 examples per class.

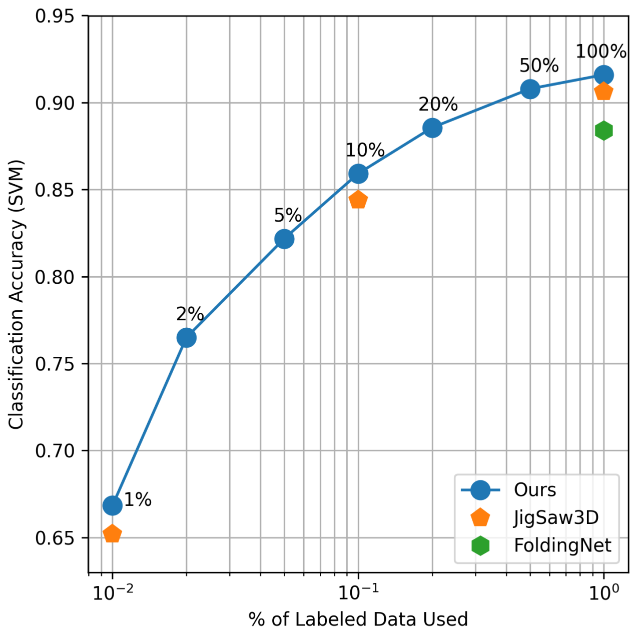

Figure 5. **Linear SVM Performance with Limited Labels:** With only 50% of the training data, our method outperforms Jigsaw3D [42] trained on 100% of the data (90.9 vs. 90.6). Additionally, our method outperforms FoldingNet [52] while using 5 times fewer labels (88.6 vs. 88.4).

Figure 5 shows the test accuracy (best of 3 runs) of a Linear SVM trained with varying amounts of labels, from $1-100\%$, using the frozen features of a pre-trained DGCNN. Surprisingly, our method trained on 50% of the data outperforms Jigsaw3D [42] trained on 100% of the data (90.9 vs. 90.6). In general, we gain a couple points in accuracy relative to Jigsaw3D, for example, at 1% data (66.8 vs. 65.2) and 10% data (85.9 vs. 84.4). For reference, we also place FoldingNet's SVM accuracy trained on 100% of the data [52] on the same graph. Our method outperforms FoldingNet while using a fifth as many labels (88.6 vs. 88.4).

B. Parametric Bottleneck as KL-Divergence Minimization

In this section, we provide more mathematical intuition for Equations 5, 6, and 7 in the main paper.

First, recall from Equation 1 that we redefined the logit matrix $\mathcal{S}$ of a segmentation network f_{ψ} in terms of a set of joint log probabilities of points $\boldsymbol{p}_i \in \mathcal{P}$ and latent binary variables $c_{ij} \in \mathcal{C}$. We restate this equation here but with added notation showing the dependence on parameters $\boldsymbol{\psi}$,

$$\boldsymbol{s}_{ij} \overset{\text{def}}{=} \log p(\boldsymbol{p}_i, c_{ij} = 1; \boldsymbol{\psi}) \tag{12}$$

Under this interpretation, $f_{\boldsymbol{\psi}}$ defines the log joint distribution over observed 3D points and latent binary correspondence variables (*i.e.* complete log likelihood), which we can marginalize over to compute the model evidence with respect to $\boldsymbol{\psi}$,

$$p(\mathcal{P};\boldsymbol{\psi}) = \prod_{i=1}^{N} p(\boldsymbol{p}_i;\boldsymbol{\psi}) = \prod_{i=1}^{N}\sum_{j=1}^{J} p(\boldsymbol{p}_i, c_{ij}=1;\boldsymbol{\psi}) = \prod_{i=1}^{N}\sum_{j=1}^{J}\exp(\boldsymbol{s}_{ij}) \tag{13}$$

Directly maximizing Equation 13 for $\boldsymbol{\psi}$ would not produce a meaningful result. Instead, we maximize the data likelihood with respect to a parametric generative model (GMM) whose specific parameter set $\boldsymbol{\Theta}$ can be analytically derived, as we will show in this section, as the result of an optimization problem.

Let $q_{\boldsymbol{\psi}} \overset{\text{def}}{=} p(\boldsymbol{p}, \mathbf{c};\boldsymbol{\psi}) = \exp\left(f_{\boldsymbol{\psi}}\right)$ and $p_{\boldsymbol{\Theta}} \overset{\text{def}}{=} p(\boldsymbol{p}, \mathbf{c};\boldsymbol{\Theta})$. Our choice of ξ functionally minimizes the KL-Divergence between $q_{\boldsymbol{\psi}}$ and $p_{\boldsymbol{\Theta}}$ with respect to $\boldsymbol{\Theta}$. Restating in mathematical form,

$$\xi(\mathcal{P},\mathcal{S}) = \underset{\boldsymbol{\Theta}}{\operatorname{argmin}}\, D_{KL}(q_{\boldsymbol{\psi}} \,||\, p_{\boldsymbol{\Theta}}) \tag{14}$$

Equation 14 provides additional theoretical insight into the "parametric bottleneck" concept: given a neural network parameterized by $\boldsymbol{\psi}$ and a parametric model parameterized by $\boldsymbol{\Theta}$, the purpose of ξ is to squeeze $\boldsymbol{\psi}$ into the specific set of parameters $\boldsymbol{\Theta}$ of the parametric model such that the KL-Divergence between these two distributions is minimized. To prove that the specific form of ξ solves the optimization in Equation 14, we first expand the KL-Divergence,

$$\begin{aligned}
D_{KL}(q_{\boldsymbol{\psi}} \,||\, p_{\boldsymbol{\Theta}}) &= -\int_{\boldsymbol{p}} \sum\nolimits_{\mathbf{c}} p(\boldsymbol{p}, \mathbf{c};\boldsymbol{\psi}) \log \frac{p(\boldsymbol{p}, \mathbf{c};\boldsymbol{\Theta})}{p(\boldsymbol{p}, \mathbf{c};\boldsymbol{\psi})} d\boldsymbol{p} &(15)\\
&= -H(q_{\boldsymbol{\psi}}) - \int_{\boldsymbol{p}} \sum\nolimits_{\mathbf{c}} p(\boldsymbol{p}_i, \mathbf{c};\boldsymbol{\psi}) \log p(\boldsymbol{p}_i, \mathbf{c};\boldsymbol{\Theta}) d\boldsymbol{p} &(16)\\
&= -H(q_{\boldsymbol{\psi}}) - \int_{\boldsymbol{p}} p(\boldsymbol{p}) \sum\nolimits_{\mathbf{c}} p(\mathbf{c}|\boldsymbol{p};\boldsymbol{\psi}) \log p(\boldsymbol{p}_i, \mathbf{c};\boldsymbol{\Theta}) d\boldsymbol{p} &(17)\\
&\approx -H(q_{\boldsymbol{\psi}}) - \frac{1}{N} \sum_{\boldsymbol{p}_i \in \mathcal{P}} \sum\nolimits_{\mathbf{c}} p(\mathbf{c}|\boldsymbol{p}_i;\boldsymbol{\psi}) \log p(\boldsymbol{p}_i, \mathbf{c};\boldsymbol{\Theta}) &(18)
\end{aligned}$$

$H(\cdot)$ denotes the entropy functional and the approximation sign is due to the summation over $\boldsymbol{p}_i \in \mathcal{P}$. Since the entropy term doesn't depend on $\boldsymbol{\Theta}$, it can be ignored when minimizing the KL-Divergence. The posterior $p(\mathbf{c}|\boldsymbol{p};\boldsymbol{\psi})$ can be calculated via Bayes' rule (see Equations 2-4 in the main text). As in the main paper, we use the shorthand $\gamma_{ij} \overset{\text{def}}{=} p(c_{ij}=1|\boldsymbol{p}_i;\boldsymbol{\psi})$.

$$\begin{aligned}
\underset{\boldsymbol{\Theta}}{\operatorname{argmin}}\, D_{KL}(q_{\boldsymbol{\psi}} \,||\, p_{\boldsymbol{\Theta}}) &= \underset{\boldsymbol{\Theta}}{\operatorname{argmax}} \sum_{\boldsymbol{p}_i \in \mathcal{P}} \sum\nolimits_{\mathbf{c}} p(\mathbf{c}|\boldsymbol{p}_i;\boldsymbol{\psi}) \log p(\boldsymbol{p}_i, \mathbf{c};\boldsymbol{\Theta}) &(19)\\
&= \underset{\boldsymbol{\Theta}}{\operatorname{argmax}} \sum_{\boldsymbol{p}_i \in \mathcal{P}} \sum_{j=1}^{J} \gamma_{ij} \log\left\{p(\boldsymbol{p}_i|c_{ij}=1;\boldsymbol{\Theta}) p(c_{ij}=1;\boldsymbol{\Theta})\right\} &(20)\\
&= \underset{\boldsymbol{\Theta}}{\operatorname{argmax}} \sum_{\boldsymbol{p}_i \in \mathcal{P}} \sum_{j=1}^{J} \gamma_{ij} \left\{\log \mathcal{N}(\boldsymbol{p}_i;\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j) + \log \pi_j\right\} &(21)
\end{aligned}$$

The optimization problem in Equation 21 shares the general underlying form of the M Step in a traditional Expectation Maximization (EM) algorithm [11] for optimizing the set of GMM parameters $\boldsymbol{\Theta}$ given responsibilities γ_{ij} computed in a previous E Step. Using the method of Lagrange multipliers to ensure $\sum_j^J \pi_j = 1$, the gradient of Equation 21 with respect to each $\boldsymbol{\mu}_j$, $\boldsymbol{\Sigma}_j$, and π_j can be easily computed, set to zero, and solved, yielding the following analytical expressions,

$$\pi_j = \frac{1}{N}\sum_{i=1}^{N}\gamma_{ij} \tag{22}$$

$$\boldsymbol{\mu}_j = \frac{1}{N\pi_j}\sum_{i=1}^{N}\gamma_{ij}\boldsymbol{p}_i \tag{23}$$

$$\boldsymbol{\Sigma}_j = \frac{1}{N\pi_j}\sum_{i=1}^{N}\gamma_{ij}(\boldsymbol{p}_i - \boldsymbol{\mu}_j)(\boldsymbol{p}_i - \boldsymbol{\mu}_j)^T \tag{24}$$

Noting that $\gamma_{ij} = \frac{p(\boldsymbol{p}_i, c_{ij}=1;\boldsymbol{\psi})}{p(\boldsymbol{p}_i;\boldsymbol{\psi})} = \frac{\exp(\boldsymbol{s}_{ij})}{\sum_{j'}^{J}\exp(\boldsymbol{s}_{ij'})}$ via Equations 2-4 in the main text, we can substitute the latter expression for γ_{ij} into Equations 22-24 to obtain the exact form of Equations 5, 6, and 7. Thus in computing these expressions, ξ functionally maps the output of a point-wise classification network $f_{\boldsymbol{\psi}}$ to the minimum KL-Divergence GMM parameter set $\boldsymbol{\Theta}$, under the interpretation that $f_{\boldsymbol{\psi}}$'s output $\mathcal{S}$ comprises joint likelihoods of observed and latent data.

C. Ablation Experiments for J, the Number of Gaussian Components

On average, it seems that a larger J relative to N helps produce better representations. However, there are a few drawbacks: numerical stability, computational cost, and memory consumption during pre-training. For our ablation, we pre-trained on ShapeNet for various J, trained a linear SVM on the ModelNet40 training split using features learned during the pre-training stage, and report accuracy on the ModelNet40 test split. Times and memory consumption are reported from a single NVIDIA Titan RTX, with batch size equal to 32. Results are shown in Table 4. Larger J incurs larger compute and memory costs for marginal downstream accuracy gains. We use $J = 32$ in the experiments in our paper as it performed the best while having reasonable memory and compute costs.

J	Memory Consumption	Time/Epoch (MM:SS)	Test Accuracy (SVM)
2	2.1GB	1:48	88.51
4	2.1GB	2:09	89.00
8	2.1GB	2:15	89.20
16	2.2GB	2:21	89.37
32	**2.2GB**	**3:30**	**90.30**
64	2.3GB	3:36	90.27
128	2.4GB	5:01	89.89
256	3.4GB	9:10	90.14
512	4.5GB	16:32	89.04

Table 4. **Ablation Study for Various J**