A. Implementation Details

For ULIP [38] and ULIP-2 [39], we adopt PointBERT [41] as the point-cloud encoder backbone. For OpenShape [16], we follow the official configuration and use its scaled PointBERT variant with 32.1M parameters, as reported in Table J of the Appendix. For Uni3D, we employ the giant model, whose point encoder contains 1,016.5M parameters. All pretrained weights are obtained directly from their public GitHub repositories. We report the zero-shot recognition accuracy of these large 3D models as the baselines for comparison, and include Point-Cache [24] for completeness.

To describe point clouds, rather than relying on a single prompt such as "a point cloud object of a {`class`}", we follow ULIP [38] and Point-PRC [23] and use 64 diverse text templates. Each template produces a textual description that is encoded into an embedding, and the 64 embeddings are averaged to obtain a class-level representation.

We further incorporate additional semantic detail only in the ModelNet-C evaluation. In this setting, the 64 generic templates are concatenated with 50 GPT-generated class-specific descriptions so that each category is provided with richer and more tailored semantics. For all subsequent evaluations, we use only the 64 generic templates without any class-specific augmentation. Even with this simplified setting, our method still yields competitive performance, which shows that it does not depend on complex or dataset-specific semantic information. This also indicates that the approach is easy to transfer across datasets without the need to generate separate prompts for each of them.

B. Derivation of Eq. (7) and Eq. (12)

B.1. Eq. (7): Textual prototype MAP

We derive the MAP estimator of the textual prototype $\boldsymbol{\nu}_c$ in Eq. (7). For each class c, an LLM produces M paraphrased prompts with embeddings $\{\mathbf{z}_{c,1}, \dots, \mathbf{z}_{c,M}\}$, from which we compute the empirical mean and scatter:

$$\bar{\mathbf{z}}_c = \frac{1}{M}\sum_{i=1}^{M}\mathbf{z}_{c,i}, \quad \mathbf{S}_c = \sum_{i=1}^{M}(\mathbf{z}_{c,i}-\bar{\mathbf{z}}_c)(\mathbf{z}_{c,i}-\bar{\mathbf{z}}_c)^\top. \tag{15}$$

We treat $\bar{\mathbf{z}}_c$ as a sufficient statistic summarizing the M paraphrases and model the latent textual prototype $\boldsymbol{\nu}_c$ as a Gaussian variable with prior

$$p(\boldsymbol{\nu}_c) = \mathcal{N}\big(\boldsymbol{\nu}_c \,\big|\, \mathbf{0},\, \beta^2\mathbf{I}\big), \tag{16}$$

where β^2 controls the prior variance.

Conditioned on $\boldsymbol{\nu}_c$, we assume that the empirical mean $\bar{\mathbf{z}}_c$ is drawn from a Gaussian whose covariance shrinks with the number of paraphrases:

$$p(\bar{\mathbf{z}}_c \mid \boldsymbol{\nu}_c) = \mathcal{N}\big(\bar{\mathbf{z}}_c \,\big|\, \boldsymbol{\nu}_c,\, \tfrac{1}{M}\mathbf{S}_c\big). \tag{17}$$

By Bayes' rule, the posterior over $\boldsymbol{\nu}_c$ is

$$p(\boldsymbol{\nu}_c \mid \bar{\mathbf{z}}_c) \propto p(\bar{\mathbf{z}}_c \mid \boldsymbol{\nu}_c)\, p(\boldsymbol{\nu}_c). \tag{18}$$

Taking the negative log and omitting constants independent of $\boldsymbol{\nu}_c$ gives

$$\begin{aligned} -\log p(\boldsymbol{\nu}_c \mid \bar{\mathbf{z}}_c) = {} & \frac{1}{2\beta^2}\boldsymbol{\nu}_c^\top\boldsymbol{\nu}_c \\ & + \frac{M}{2}(\bar{\mathbf{z}}_c - \boldsymbol{\nu}_c)^\top\mathbf{S}_c^{-1}(\bar{\mathbf{z}}_c - \boldsymbol{\nu}_c) \\ & + \text{const}. \end{aligned} \tag{19}$$

Expanding the second term, we obtain:

$$\begin{aligned} -\log p(\boldsymbol{\nu}_c \mid \bar{\mathbf{z}}_c) = {} & \frac{1}{2}\boldsymbol{\nu}_c^\top\big(\beta^{-2}\mathbf{I} + M\mathbf{S}_c^{-1}\big)\boldsymbol{\nu}_c \\ & - \boldsymbol{\nu}_c^\top\big(M\mathbf{S}_c^{-1}\bar{\mathbf{z}}_c\big) + \text{const}, \end{aligned} \tag{20}$$

which matches the canonical Gaussian form in $\boldsymbol{\nu}_c$ with precision

$$\boldsymbol{\Lambda}_c = \beta^{-2}\mathbf{I} + M\mathbf{S}_c^{-1}, \tag{21}$$

and natural parameter

$$\boldsymbol{\eta}_c = M\mathbf{S}_c^{-1}\bar{\mathbf{z}}_c. \tag{22}$$

Thus the posterior over $\boldsymbol{\nu}_c$ is Gaussian,

$$p(\boldsymbol{\nu}_c \mid \bar{\mathbf{z}}_c) = \mathcal{N}\big(\boldsymbol{\nu}_c \,\big|\, \boldsymbol{\nu}_c^{\text{MAP}},\, \boldsymbol{\Sigma}_{\nu_c}\big), \tag{23}$$

with

$$\boldsymbol{\Sigma}_{\nu_c} = \boldsymbol{\Lambda}_c^{-1} = \big(\beta^{-2}\mathbf{I} + M\mathbf{S}_c^{-1}\big)^{-1}, \tag{24}$$

$$\boldsymbol{\nu}_c^{\text{MAP}} = \boldsymbol{\Sigma}_{\nu_c}\boldsymbol{\eta}_c = \big(\beta^{-2}\mathbf{I} + M\mathbf{S}_c^{-1}\big)^{-1}M\mathbf{S}_c^{-1}\bar{\mathbf{z}}_c. \tag{25}$$

Since $\mathbf{S}_c$ in Eq. (3) is an unnormalized scatter matrix, its global scale can be absorbed into M without changing the relative weighting between the prior and data terms. Under this convention, simplifying the common scalar factor yields the compact expression used in the main paper:

$$\boldsymbol{\nu}_c^{\text{MAP}} = \big(\beta^{-2}\mathbf{I} + M\mathbf{S}_c^{-1}\big)^{-1}\mathbf{S}_c^{-1}\bar{\mathbf{z}}_c, \tag{26}$$

which gives Eq. (7).

B.2. Eq. (12): Geometric distribution update

We next derive the recursive update in Eq. (12) for a fixed class c, omitting the class index when unambiguous. At test-time step t, the geometric parameters are:

$$\Theta_t = \{\boldsymbol{\mu}_t, \boldsymbol{\Sigma}_t\}, \tag{27}$$

and the prior is given by the previous posterior:

$$p(\Theta_t) = p(\Theta_{t-1} \mid \mathbf{x}_{t-1}). \tag{28}$$

Under the Gaussian assumptions in Eq. (8) and Eq. (10), the mean evolves according to:

$$\boldsymbol{\mu}_t \sim \mathcal{N}(\boldsymbol{\mu}_{t-1}, \boldsymbol{\Sigma}_{t-1}), \tag{29}$$

$$\mathbf{x}_t \mid \boldsymbol{\mu}_t \sim \mathcal{N}(\boldsymbol{\mu}_t, \boldsymbol{\Sigma}). \tag{30}$$

Applying Bayes' rule yields:

$$p(\boldsymbol{\mu}_t \mid \mathbf{x}_t) \propto p(\mathbf{x}_t \mid \boldsymbol{\mu}_t)\, p(\boldsymbol{\mu}_t). \tag{31}$$

Taking the negative log (up to constants independent of $\boldsymbol{\mu}_t$) gives:

$$\begin{aligned} -\log p(\boldsymbol{\mu}_t \mid \mathbf{x}_t) = &\frac{1}{2}(\boldsymbol{\mu}_t - \boldsymbol{\mu}_{t-1})^\top \boldsymbol{\Sigma}_{t-1}^{-1}(\boldsymbol{\mu}_t - \boldsymbol{\mu}_{t-1}) \\ &+ \frac{1}{2}(\mathbf{x}_t - \boldsymbol{\mu}_t)^\top \boldsymbol{\Sigma}^{-1}(\mathbf{x}_t - \boldsymbol{\mu}_t). \end{aligned} \tag{32}$$

Expanding the quadratic terms leads to:

$$\begin{aligned} -\log p(\boldsymbol{\mu}_t \mid \mathbf{x}_t) = &\frac{1}{2}\boldsymbol{\mu}_t^\top (\boldsymbol{\Sigma}_{t-1}^{-1} + \boldsymbol{\Sigma}^{-1})\boldsymbol{\mu}_t \\ &- \boldsymbol{\mu}_t^\top \left(\boldsymbol{\Sigma}_{t-1}^{-1}\boldsymbol{\mu}_{t-1} + \boldsymbol{\Sigma}^{-1}\mathbf{x}_t\right) + \text{const}, \end{aligned} \tag{33}$$

which matches the canonical Gaussian form with precision:

$$\boldsymbol{\Lambda}_t = \boldsymbol{\Sigma}_{t-1}^{-1} + \boldsymbol{\Sigma}^{-1}, \tag{34}$$

and natural parameter:

$$\boldsymbol{\eta}_t = \boldsymbol{\Sigma}_{t-1}^{-1}\boldsymbol{\mu}_{t-1} + \boldsymbol{\Sigma}^{-1}\mathbf{x}_t. \tag{35}$$

Thus the posterior is Gaussian with parameters:

$$\boldsymbol{\Sigma}_t = \boldsymbol{\Lambda}_t^{-1} = \left(\boldsymbol{\Sigma}_{t-1}^{-1} + \boldsymbol{\Sigma}^{-1}\right)^{-1}, \tag{36}$$

$$\boldsymbol{\mu}_t = \boldsymbol{\Sigma}_t\, \boldsymbol{\eta}_t = \boldsymbol{\Sigma}_t\left(\boldsymbol{\Sigma}^{-1}\mathbf{x}_t + \boldsymbol{\Sigma}_{t-1}^{-1}\boldsymbol{\mu}_{t-1}\right). \tag{37}$$

Restoring the class index and substituting the class-specific covariance $\boldsymbol{\Sigma}^c$, we obtain:

$$\boldsymbol{\mu}_t^c = \boldsymbol{\Sigma}_t^c\left[(\boldsymbol{\Sigma}^c)^{-1}\mathbf{x}_t + (\boldsymbol{\Sigma}_{t-1}^c)^{-1}\boldsymbol{\mu}_{t-1}^c\right], \tag{38}$$

$$\boldsymbol{\Sigma}_t^c = \left[(\boldsymbol{\Sigma}_{t-1}^c)^{-1} + (\boldsymbol{\Sigma}^c)^{-1}\right]^{-1}, \tag{39}$$

which gives Eq. (12) in the main paper.

C. Additional Results

Robustness evaluation. Tables G, H, and I present the recognition accuracy under different corruption settings. Overall, `BayesMM` consistently improves performance over the baseline models (ULIP, ULIP-2, O-Shape, and Uni3D) as well as the Point-Cache variants.

On S-OBJ_ONLY-C (Table G), `BayesMM` consistently outperforms Point-Cache, improving the average accuracy by 3.0% to 4.5%, and achieves a 4% to 7% gain over the original backbone models, demonstrating clear improvements in robustness. On S-OBJ_BG-C (Table H), which introduces background clutter, `BayesMM` achieves 2.5% to 3.5% higher average accuracy than Point-Cache and up to 6% to 8% improvement over the original models, indicating strong generalization in more challenging scenes. On the most challenging split S-PB_T50-RS-C (Table I), `BayesMM` increases the average accuracy by 4.5% to 5.0% over Point-Cache and by 7% to 10% over the original backbones, demonstrating that our approach can effectively handle severe corruptions and partial observations, significantly enhancing model robustness. These consistent gains across settings highlight the effectiveness of `BayesMM` in improving robustness under diverse and severe corruptions.

Memory usage and inference throughput. Table J reports the memory consumption across ModelNet-C, Omni3D, and O-LVIS. While Point-Cache exhibits comparable or slightly higher memory usage on smaller datasets, its parameter footprint grows substantially as the number of object categories increases, particularly on O-LVIS. In contrast, `BayesMM` maintains a consistently lightweight profile, indicating that the robustness improvements introduced by our method come with only negligible memory overhead. Table K presents the inference throughput measured on S-OBJ_ONLY. The throughput reduction introduced by `BayesMM` remains within 2% to 5% across all evaluated backbones, indicating that the additional operations do not significantly affect runtime. The resulting throughput decrease remains modest relative to the robustness enhancements achieved by `BayesMM`.

Table G. Comparison of recognition accuracy on S-OBJ_ONLY-C, which contains seven types of corruptions. Results are reported at corruption severity level 2. Each clean point cloud contains 1024 points. SONN refers to ScanObjectNN.

Method	**Original Data**	**Corruption Type**							**Avg.**
	SONN	Add Global	Add Local	Drop Global	Drop Local	Rotate	Scale	Jitter	
ULIP [38]	49.05	31.50	34.77	51.29	38.38	48.36	44.58	36.83	41.85
+ Point-Cache (Global)	52.15	35.80	37.01	54.39	41.82	49.74	45.09	**40.28**	44.54
+ Point-Cache (Hierarchical)	52.15	32.01	**38.04**	54.56	45.27	50.95	45.96	39.24	44.77
+ `BayesMM`	**54.04**	**39.24**	**38.04**	**55.42**	**46.30**	**52.15**	**47.68**	35.11	**45.75**
ULIP-2 [39]	42.00	40.45	41.31	37.69	30.29	38.21	44.45	22.89	37.16
+ Point-Cache (Global)	48.19	49.05	46.30	45.09	37.18	41.65	44.41	25.99	42.24
+ Point-Cache (Hierarchical)	51.98	49.05	46.30	**48.88**	40.45	**45.78**	45.09	25.99	44.19
+ `BayesMM`	**52.67**	**54.04**	**48.02**	45.78	**42.34**	44.75	**45.96**	**28.57**	**45.52**
O-Shape [16]	53.18	49.91	46.30	52.15	36.66	46.64	46.82	30.81	45.31
+ Point-Cache (Global)	56.80	56.45	51.98	54.56	40.45	51.81	49.23	37.69	49.90
+ Point-Cache (Hierarchical)	58.69	59.04	53.01	55.94	41.82	51.12	48.54	39.41	50.95
+ `BayesMM`	**61.96**	**61.10**	**55.76**	**59.03**	**48.54**	**54.90**	**53.35**	**40.96**	**54.44**
Uni3D [48]	65.58	62.65	56.45	60.07	49.40	61.62	56.11	43.55	56.93
+ Point-Cache (Global)	70.05	65.06	59.38	63.68	54.39	63.34	60.07	51.29	60.91
+ Point-Cache (Hierarchical)	70.22	65.40	58.00	64.20	54.91	61.96	62.13	53.18	61.25
+ `BayesMM`	**71.60**	**69.53**	**60.06**	**64.54**	**60.07**	**68.33**	**63.86**	**52.32**	**63.79**

Table H. Comparison of recognition accuracy on S-OBJ_BG-C, which includes seven types of corruptions. Results are reported at corruption severity level 2. Each clean point cloud contains 1024 points.

Method	**Original Data**	**Corruption Type**							**Avg.**
	SONN	Add Global	Add Local	Drop Global	Drop Local	Rotate	Scale	Jitter	
ULIP [38]	45.96	27.19	25.82	45.61	34.25	40.96	40.10	30.98	36.36
+ Point-Cache (Global)	48.88	30.46	30.46	49.05	39.59	44.92	42.17	31.84	39.68
+ Point-Cache (Hierarchical)	49.74	28.23	30.12	48.71	40.45	43.55	40.28	**34.42**	39.44
+ `BayesMM`	**52.67**	**33.05**	**34.08**	**50.43**	**41.65**	**48.53**	**45.78**	31.50	**41.81**
ULIP-2 [39]	48.19	40.62	38.90	39.24	32.36	41.14	42.86	21.17	38.04
+ Point-Cache (Global)	52.50	48.19	45.09	46.82	39.07	46.64	48.02	**26**.51	44.10
+ Point-Cache (Hierarchical)	54.73	**51.64**	**47.16**	**50.95**	39.76	**53.01**	**51.81**	22.72	**46.47**
+ `BayesMM`	**56.80**	50.77	46.82	49.40	**40.45**	50.26	49.57	25.47	46.19
O-Shape [16]	55.94	49.40	48.19	52.67	42.51	48.88	47.16	31.84	47.08
+ Point-Cache (Global)	59.72	57.49	51.12	59.72	48.71	56.11	54.22	35.28	52.80
+ Point-Cache (Hierarchical)	62.65	58.00	51.64	59.55	47.85	54.91	53.36	36.49	53.06
+ `BayesMM`	**64.72**	**60.41**	**54.90**	**61.62**	**52.32**	**60.41**	**57.14**	**38.21**	**55.09**
Uni3D [48]	60.24	58.00	52.32	51.64	44.23	58.00	51.81	39.24	51.94
+ Point-Cache (Global)	63.86	**66.27**	**57.83**	56.11	50.77	61.62	56.11	44.23	57.10
+ Point-Cache (Hierarchical)	62.82	64.72	57.14	58.52	50.43	60.93	**59.55**	46.30	57.55
+ `BayesMM`	**68.50**	66.26	54.39	**60.58**	**55.07**	**65.23**	58.86	**49.57**	**59.06**

Table I. Comparison of corruption generalization on S-PB_T50-RS-C, the most challenging split of ScanObjectNN. Each clean point cloud is represented by 1024 points. SONN denotes ScanObjectNN.

Method	**Original Data**	**Corruption Type**							**Avg.**
	SONN	Add Global	Add Local	Drop Global	Drop Local	Rotate	Scale	Jitter	
ULIP [38]	29.29	19.26	18.39	30.99	23.91	27.48	26.34	21.44	24.64
+ Point-Cache (Global)	32.37	22.87	20.85	33.31	27.90	30.85	<u>28.63</u>	24.53	27.66
+ Point-Cache (Hierarchical)	<u>32.48</u>	<u>23.46</u>	<u>22.69</u>	<u>34.70</u>	<u>31.75</u>	<u>33.00</u>	28.28	<u>25.05</u>	<u>28.93</u>
+ `BayesMM`	**40.52**	**29.53**	**25.92**	**39.21**	**33.59**	**35.74**	**32.44**	**24.67**	**33.18**
ULIP-2 [39]	33.38	30.29	29.42	28.24	24.91	28.56	30.22	12.98	27.25
+ Point-Cache (Global)	40.28	<u>36.40</u>	33.80	35.39	30.88	33.66	35.01	18.36	32.97
+ Point-Cache (Hierarchical)	<u>42.40</u>	35.70	<u>34.42</u>	<u>37.75</u>	<u>34.21</u>	<u>36.26</u>	<u>36.09</u>	<u>19.12</u>	<u>34.49</u>
+ `BayesMM`	**46.31**	**41.29**	**37.82**	**40.46**	**34.57**	**39.73**	**37.51**	**16.53**	**36.78**
O-Shape [16]	41.12	32.41	35.60	37.80	27.34	36.61	35.22	18.88	33.12
+ Point-Cache (Global)	42.16	40.32	37.58	42.02	33.76	41.53	<u>38.24</u>	24.12	37.47
+ Point-Cache (Hierarchical)	<u>43.72</u>	<u>40.91</u>	<u>39.24</u>	<u>43.03</u>	<u>35.22</u>	<u>43.06</u>	37.40	<u>25.05</u>	<u>38.45</u>
+ `BayesMM`	**50.52**	**49.51**	**43.64**	**49.53**	**41.22**	**47.11**	**45.31**	**30.29**	**44.39**
Uni3D [48]	46.04	48.23	37.99	36.75	31.47	44.00	37.37	28.66	38.46
+ Point-Cache (Global)	50.28	<u>52.57</u>	<u>42.23</u>	42.61	36.29	47.22	39.83	33.48	43.06
+ Point-Cache (Hierarchical)	<u>51.13</u>	51.67	41.88	<u>44.59</u>	<u>38.79</u>	<u>49.03</u>	<u>41.05</u>	<u>34.70</u>	<u>44.10</u>
+ `BayesMM`	**57.04**	**59.30**	**45.70**	**49.10**	**44.41**	**53.16**	**48.37**	**37.86**	**49.17**

Table J. Memory usage (MB) comparison across different datasets. Numbers below each dataset name indicate the number of classes. Point-Cache denotes the hierarchical variant.

Method	ModelNet-C (40)	Omni3D (216)	O-LVIS (1156)	#Params (M)
ULIP-2	**1,556**	**1,558**	**1,556**	85.7
+ Point-Cache	**1,556**	**1,558**	1,566	85.7
+ **BayesMM**	<u>1,560</u>	<u>1,560</u>	<u>1,562</u>	85.7
OpenShape	**7,056**	**7,058**	**7,116**	2,571.9
+ Point-Cache	<u>7,058</u>	<u>7,062</u>	7,150	2,571.9
+ **BayesMM**	7,076	7,080	<u>7,084</u>	2,571.9

Table K. Inference throughput (samples per second) on S-OBJ_ONLY. All experiments are conducted with a batch size of 1 on an RTX 3090 GPU. Point-Cache refers to the hierarchical variant of Point-Cache here.

Method	ULIP	ULIP2	OpenShape	Uni3D
Vanilla	**11.25**	**11.25**	**8.60**	**7.72**
+ Point-Cache [24]	<u>11.17</u>	<u>11.17</u>	<u>8.57</u>	<u>7.62</u>
+ **BayesMM**	10.90	10.91	8.28	7.41