Learning Hierarchical Orthogonal Prototypes for Generalized Few-Shot 3D Point Cloud Segmentation

Supplementary Material

Guide to the Supplementary Material

This supplementary material provides:

I. Additional Related Work

Dynamic prototypes and pseudo embeddings. Recent works explore dynamic prototype adaptation and distillation-based refinement for few-shot point cloud segmentation, improving prototype robustness under sparse supervision [1], [2]. For generalized few-shot 3D segmentation, pseudo-embedding strategies further leverage unlabeled context to mitigate base–novel confusion [3]. In contrast, HOP3D targets base–novel interference with an explicit dual-orthogonality design that jointly stabilizes optimization in gradient space (HOP-Grad) and preserves class-subspace geometry in prototype space (HOP-Rep), which is complementary to dynamic prototype refinement and pseudo-embedding approaches.

Orthogonality in optimization and gradients. Orthogonal gradient projection has been studied to mitigate interference and forgetting in continual and multi-task learning. GEM constrains current updates so that losses on stored samples from previous tasks do not increase [4], while OGD stores gradient directions from earlier tasks and projects new-task updates onto their orthogonal complement to preserve past knowledge [5]. PCGrad resolves gradient conflicts by removing opposing components between task gradients [6]. Different from these settings, **HOP-Grad** is tailored to the base–novel stability–plasticity trade-off in GFS-3DS: we construct a compact orthonormal basis from *base-task gradients* after Phase 1 and apply projection *only in Phase 2* to filter base-aligned components from few-shot novel updates. Moreover, unlike OGD/GEM/PCGrad which operate in a task-sequential or multi-task optimization setting, HOP-Grad serves as a targeted intervention for base–novel adaptation and is coupled with HOP-Rep so that gradient-space constraints align with the prototype-induced class-subspace geometry, whereas prior works typically impose orthogonality in a single space (either gradients or representations) without such coupling.

Orthogonality in representation and prototypes. Orthogonality has been widely adopted to encourage decorrelated representations and improve generalization, e.g., via orthogonal/DeCov-style regularization on features or weights [7], [8]. More closely related, POP/OPNet learns *orthogonal class prototypes* for generalized few-shot *2D* segmentation to reduce base–novel confusion by structuring prototypes into near-orthogonal directions [9]. In contrast, **HOP-Rep** is designed for generalized few-shot *3D* point cloud segmentation and introduces an explicit *sequential subspace* decomposition: we first project point features onto the *base-prototype subspace* to obtain a base-aligned component and a residual, and then further project the residual onto *novel prototypes*, yielding base-aligned, novel-aligned, and remaining residual components for prediction. This differs from POP-style *per-class* prototype projection in 2D segmentation: HOP-Rep performs a *subspace-level* residual cascade tailored to dense 3D prediction, explicitly routing features through base and novel subspaces to better isolate base and novel learning.

Entropy-based objectives for certainty and balance. Entropy minimization encourages confident predictions on unlabeled or weakly labeled data and has been used in semi-supervised learning and adaptation [10], [11]. Complementarily, marginal-entropy maximization (often framed as information maximization) discourages degenerate solutions and promotes balanced predictions across classes [12]. Recent transductive/test-time adaptation methods often optimize these objectives at inference time [11]. In contrast, **HOP-Ent** integrates conditional-entropy minimization and marginal-entropy maximization *into Phase 2 training* for few-shot 3D segmentation, jointly refining novel predictions without any post-training or test-time optimization, and complements our dual-orthogonality design.

II. Algorithms and Pseudocode

This section provides detailed algorithmic descriptions of the **HOP3D** framework. We first formalize the Gram–Schmidt-based gradient subspace construction employed by **HOP-Grad**, followed by the hierarchical representation decomposition in **HOP-Rep**. Core PyTorch-style implementations are also included for completeness.

A. Orthogonal Gradient Projection (HOP-Grad)

The **HOP-Grad** module aims to preserve base-class knowledge by constraining the optimization of shared parameters to remain within the orthogonal complement of the gradient subspace induced by the base task.

Gradient Subspace Construction via Gram–Schmidt.
After Phase 1 training, we collect a sequence of T gradient vectors $\mathcal{G} = \{g^{(t)}\}_{t=1}^{T}$, where $g^{(t)} \in \mathbb{R}^d$ denotes the gradient of shared parameters computed at the t-th sampling step. To obtain an orthonormal basis for the subspace spanned by these gradients, we apply the Gram–Schmidt process.

Specifically, the basis vectors u_k are constructed iteratively as follows:

1) **Initialization:** Set $u'_1 = g^{(1)}$ and normalize it as
$$u_1 = \frac{u'_1}{\|u'_1\|_2 + \epsilon}, \tag{1}$$
where $\epsilon = 10^{-8}$ is a small constant for numerical stability.
2) **Orthogonalization:** For each subsequent gradient $g^{(k)}$ ($k = 2, \ldots, T$), subtract its projections onto the previously computed basis vectors:
$$u'_k = g^{(k)} - \sum_{j=1}^{k-1} \langle g^{(k)}, u_j \rangle u_j, \tag{2}$$
where $\langle \cdot, \cdot \rangle$ denotes the inner product.
3) **Normalization:** If $\|u'_k\|_2 > \epsilon$, the new basis vector is defined as
$$u_k = \frac{u'_k}{\|u'_k\|_2}. \tag{3}$$

This procedure yields an orthonormal matrix $B = [u_1, \ldots, u_r] \in \mathbb{R}^{d \times r}$, where $r \le T$ corresponds to the effective rank of the collected gradient set.

Gradient Projection in Phase 2.
During Phase 2 adaptation, for any gradient g computed from the novel-class objectives, we project it as
$$\tilde{g} = g - B(B^\top g). \tag{4}$$
The projected gradient $\tilde{g}$ is orthogonal to the base gradient subspace, thereby preventing updates from interfering with the representations learned during Phase 1. The complete procedure for constructing the base gradient subspace and performing online projection during Phase 2 is summarized in Algorithm 1.

B. Hierarchical Orthogonal Decomposition (HOP-Rep)

The **HOP-Rep** module enforces a hierarchical decomposition of point features into orthogonal semantic components. Instead of learning a single shared embedding space, **HOP3D** explicitly separates base and novel representations to reduce semantic interference.

As summarized in Algorithm 2, the embedded point feature f is first decomposed into a base-aligned component f_b and an initial residual $r^{(0)}$. Novel-class prototypes are subsequently optimized within the subspace spanned by this residual, ensuring that novel adaptation proceeds in a representation space orthogonal to base semantics.

Algorithm 1: Basis Construction and Projection (**HOP-Grad**)

```
Input: Base model $\Theta_{\text{base}}$, base dataset $\mathcal{D}_{\text{base}}$, sample count $T$.
Output: Orthonormal basis $B$, projected gradient $\tilde{g}$.
// Stage 1: Basis Construction
1  Initialize $\mathcal{G} \leftarrow \emptyset$;
2  for $t = 1$ to $T$ do
3      $g^{(t)} \leftarrow \nabla_\phi \mathcal{L}_{\text{base}}$;
4      $\mathcal{G} \leftarrow \mathcal{G} \cup \{g^{(t)}\}$;
5      Zero gradients;
6  end
7  $B \leftarrow$ GramSchmidt($\mathcal{G}$);
// Stage 2: Online Projection
8  for each training step in Phase 2 do
9      $g \leftarrow \nabla_\phi \mathcal{L}_{\text{novel}}$;
10     $\tilde{g} \leftarrow g - B(B^\top g)$;
11     $\phi \leftarrow$ Optimizer($\phi, \tilde{g}$);
12 end
```

Algorithm 2: Hierarchical Orthogonal Decomposition (**HOP-Rep**)

```
Input: Point $x$, feature $f_\theta(x)$, base prototypes $\{\hat{s}_{b,k}\}$, novel prototypes $\{\hat{s}_{n,\ell}\}$.
Output: Logits $z(x)$.
1  $f_b(x) = \sum_{k=1}^{K_b} \langle f_\theta(x), \hat{s}_{b,k} \rangle \hat{s}_{b,k}$;
2  $r^{(0)}(x) = f_\theta(x) - f_b(x)$;
3  $f_n(x) = \sum_{\ell=1}^{K_n} \langle r^{(0)}(x), \hat{s}_{n,\ell} \rangle \hat{s}_{n,\ell}$;
4  $r^{(1)}(x) = r^{(0)}(x) - f_n(x)$;
5  $z_b = h_b(f_b(x))$;
6  $z_n = h_n(f_n(x), r^{(1)}(x))$;
7  $z(x) = [z_b, z_n]$;
```

III. Implementation Details and Reproducibility

This section elaborates on the implementation details of HOP3D, with a particular focus on how the mathematical formulations presented in the Methodology section of the main paper are instantiated in our code.

A. Network Architecture and Representation Heads

As described in the main paper, our framework utilizes PTv3 [13] as the backbone to extract point-wise embeddings $f_\theta(x) \in \mathbb{R}^{256}$. The hierarchical decomposition in **HOP-Rep** is implemented through specialized projection heads:

- **Prototype Storage**: The base prototypes $\hat{S}_b$ and novel prototypes $\hat{S}_n$ are stored as `nn.Parameter` tensors. In the forward pass, these are normalized via

`F.normalize` to yield the orthonormal bases used in the projection equations.

- **Hierarchical MLPs**: The classification heads $h_b(\cdot)$ and $h_n(\cdot)$ are instantiated as three-layer MLPs. Specifically, h_b processes the base-aligned component $f_b(x)$, while h_n takes the concatenation of the novel-aligned component $f_n(x)$ and the final residual $r^{(1)}(x)$ as input.

B. Optimization Strategy and Gradient Projection

The training of HOP3D follows a strict two-stage schedule. During Phase 2, we implement the gradient projection $\tilde{g} = g - B(B^\top g)$ as an *inplace* modification of the parameter gradients. Specifically, after the backward pass but before the optimizer step, we flatten the gradients of the protected parameters, project them onto the orthogonal complement of the basis B (constructed from T samples), and reshape them back to their original dimensions. This ensures that updates for novel classes do not disrupt the base optimization manifold.

C. Hyperparameter Summary

To support reproducibility, Table I summarizes the key hyperparameters and their corresponding notations. Following the main paper, we use a shared pretraining learning rate across benchmarks, while adaptation rates are tuned separately to reflect dataset differences: 1×10^{-3} for ScanNet200 and 7×10^{-3} for ScanNet++.

TABLE I
SUMMARY OF KEY HYPERPARAMETERS AND THEIR CORRESPONDING SYMBOLS IN THE METHODOLOGY.

Description	Symbol	ScanNet200	ScanNet++
Optimization Parameters			
Base pretraining LR (Phase 1)	η_{P1}	6×10^{-3}	6×10^{-3}
Novel adaptation LR (Phase 2)	η_{novel}	1×10^{-3}	7×10^{-3}
Base micro-tuning LR (Phase 2)	η_{base}	1×10^{-4}	1×10^{-4}
Backbone adaptation LR (Phase 2)	$\eta_{backbone}$	5×10^{-5}	5×10^{-5}
Gradient sample count (T)	T	500	500
Loss Weights			
Orthogonality weight (Phase 1)	$\lambda_{\text{orth}}^{(1)}$	10.0	10.0
Orthogonality weight (Phase 2)	$\lambda_{\text{orth}}^{(2)}$	0.1	0.1
Conditional entropy weight	λ_{cond}	0.1	0.1
Marginal entropy weight	λ_{marg}	0.2	0.2
Data Pipeline			
Voxel size	δ	0.02 m	0.02 m
Pseudo-label selection threshold	τ_{PS}	0.6	0.6
Adaptive infilling threshold	τ_{AI}	0.9	0.9

D. Code and Reproducibility

To ensure reproducibility, we provide: **`HOP3D.py`**, implementing the core logic in Section II. Full code will be released upon acceptance.

IV. Benchmark Details and Consistency

As discussed in Section III-A (Experimental Setup) of the main paper, we evaluate HOP3D on the Generalized Few-Shot 3D Segmentation (GFS-3DS) task using two large-scale benchmarks. To ensure a fair and rigorous comparison with the current state-of-the-art, we strictly follow the benchmark settings and class-split protocols established by GFS-VL [14], which are built upon ScanNet200 [15] and ScanNet++ [16].

Benchmark Protocol. Following the strategy proposed in GFS-VL [14], categories are selected based on their occurrence counts to ensure adequate representation while maintaining the long-tail characteristics of real-world scenes. The selection process and frequency thresholds adopted from the baseline are summarized as follows:

- **ScanNet200**: Consistent with GFS-VL, 57 classes with occurrence counts exceeding 100 were retained. The 12 most frequent classes are designated as base classes, and the remaining 45 serve as novel classes.
- **ScanNet++**: Similarly, 30 classes with occurrence counts exceeding 80 were retained. The top 12 classes form the base set, while the remaining 18 constitute the novel set.

By adopting these standardized benchmarks, we provide a direct comparison with previous methods under identical data distributions and semantic splits.

Detailed Class Lists. The specific class lists used in our experiments, which are identical to those in the GFS-VL [14] benchmark, are provided below:

- **ScanNet200 Benchmark**:
  - `Base Classes (12)`: [‘refrigerator’, ‘desk’, ‘curtain’, ‘bookshelf’, ‘bed’, ‘table’, ‘window’, ‘cabinet’, ‘door’, ‘chair’, ‘floor’, ‘wall’]
  - `Novel Classes (45)`: [‘trash can’, ‘ceiling’, ‘doorframe’, ‘object’, ‘shelf’, ‘sink’, ‘picture’, ‘backpack’, ‘couch’, ‘box’, ‘pillow’, ‘radiator’, ‘mirror’, ‘whiteboard’, ‘lamp’, ‘toilet’, ‘book’, ‘monitor’, ‘towel’, ‘tv’, ‘clothes’, ‘coffee table’, ‘office chair’, ‘nightstand’, ‘bag’, ‘dresser’, ‘toilet paper’, ‘recycling bin’, ‘kitchen cabinet’, ‘bathtub’, ‘telephone’, ‘plant’, ‘stool’, ‘keyboard’, ‘shoe’, ‘jacket’, ‘shower curtain’, ‘armchair’, ‘microwave’, ‘computer tower’, ‘bathroom vanity’, ‘kitchen counter’, ‘shower wall’, ‘paper towel dispenser’, ‘file cabinet’]
- **ScanNet++ Benchmark**:
  - `Base Classes (12)`: [‘wall’, ‘floor’, ‘door’, ‘ceiling’, ‘table’, ‘window’, ‘box’, ‘ceiling lamp’, ‘light switch’, ‘cabinet’, ‘chair’, ‘heater’]
  - `Novel Classes (18)`: [‘monitor’, ‘bottle’, ‘whiteboard’, ‘office chair’, ‘doorframe’, ‘keyboard’, ‘window frame’, ‘mouse’, ‘paper’, ‘blinds’, ‘trash can’, ‘telephone’, ‘book’, ‘shelf’, ‘sink’, ‘windowsill’, ‘bag’, ‘smoke detector’]

These benchmarks represent the most challenging scenarios for GFS-3DS due to their high semantic granularity and scene diversity. Maintaining consistency with established baselines ensures that the improvements achieved by HOP3D are attributable to our hierarchical orthogonalization and regularized adaptation, rather than variations in data partitioning.

V. Extended Visualization and Analysis

In this section, we provide a detailed interpretation of the prototype similarity matrices presented in the main paper and

refrigerator	desk	curtain	bookshelf	bed	table
window	cabinet	door	chair	floor	wall

desk	sink	picture	backpack	couch
box	pillow	bed	radiator	mirror
whiteboard	lamp	curtain	toilet	book
monitor	bookshelf	towel	tv	clothes
coffee table	office chair	nightstand	bag	dresser
toilet paper	recycling bin	kitchen cabinet	refrigerator	bathtub
telephone	plant	stool	keyboard	shoe
jacket	shower curtain	armchair	microwave	computer tower
bathroom vanity	kitchen counter	shower wall	paper towel dispenser	file cabinet

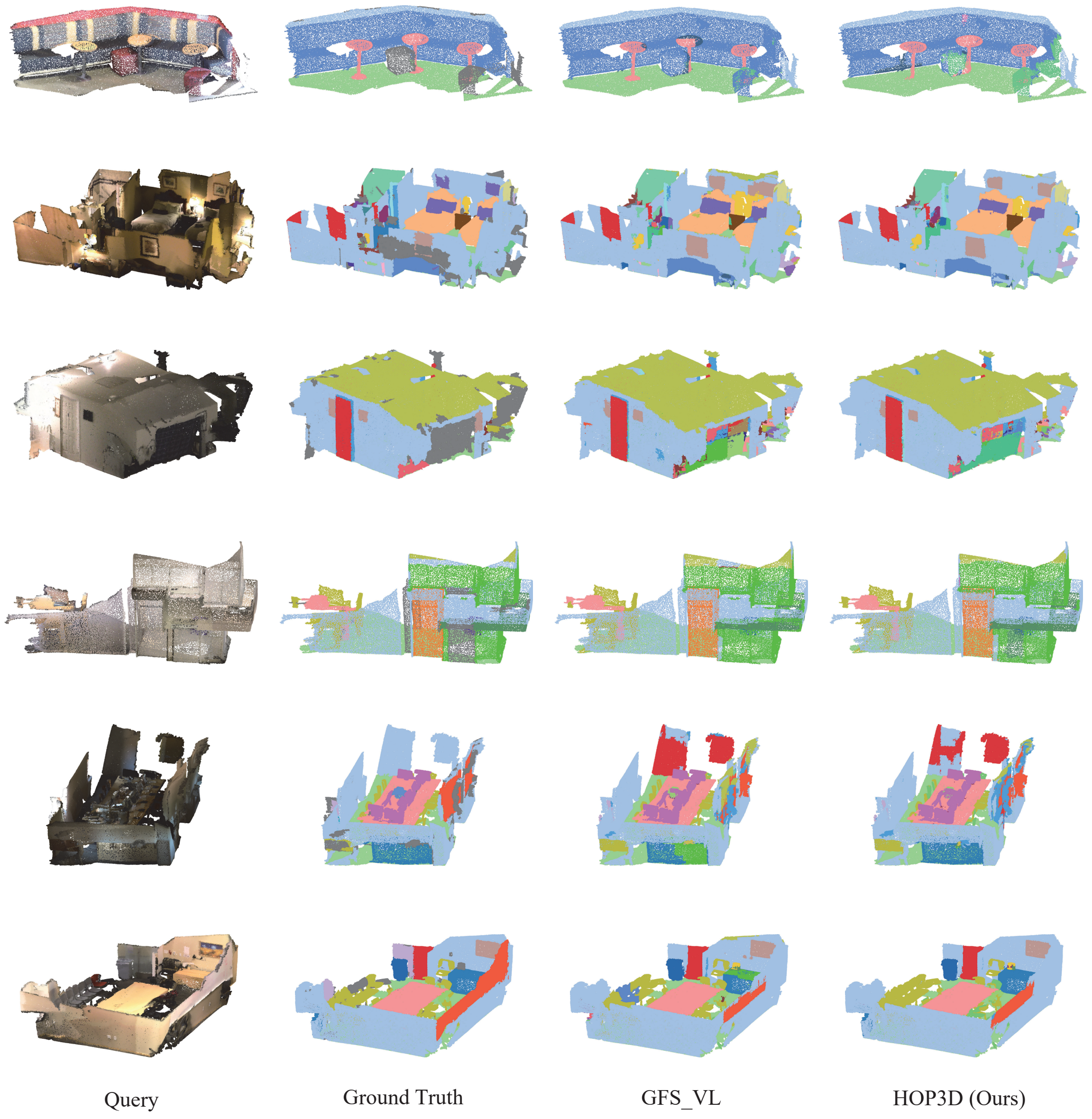

Fig. 1. **Qualitative comparison on ScanNet200.** Results show HOP3D outperforms GFS_VL [14] in GFS-3DS. The top legend covers 57 categories: the first two rows indicate 12 base classes, and the subsequent nine rows denote 45 novel classes. HOP3D demonstrates sharper segmentation and stronger generalization across diverse scenes.

present an extensive qualitative gallery to further demonstrate the practical advantages of HOP3D.

A. Deep Interpretation of Prototype Orthogonality

In **Figure 5** of the main paper, we visualized the cosine similarity matrices of learned prototypes across different training phases. Here, we elaborate on the physical implications of these visualizations and how they validate the core design of **HOP-Rep**:

Phase 1: Base-Class Subspace Initialization. In the Phase 1 matrix of **Figure 5**, the near-identity structure among base-class prototypes indicates that our orthogonality regularizer successfully enforces the prototypes to span a mutually independent semantic subspace. This initialization provides a "clean" geometric foundation, ensuring that base classes are well-separated before the adaptation to novel categories.

Phase 2: Mitigating Semantic Leakage. A critical challenge in GFS-3DS is "prototype warping," where newly learned novel prototypes drift into the pre-trained base-class manifold. As shown in the base-novel similarity matrices in **Figure 5**, HOP3D maintains near-zero inter-group correlation. This confirms that **HOP-Rep**, by decomposing features in the residual space, effectively isolates novel-class embeddings. This structural decoupling prevents "semantic leakage," a common failure mode where novel objects are misclassified as geometrically similar base classes.

Resistance to Prototype Collapse. In traditional few-shot settings, sparse supervision often leads to "prototype collapse," where novel-class representations become highly correlated. The sparse nature of our novel-novel similarity matrix in **Figure 5** demonstrates that the synergy between **HOP-Net** and **HOP-Ent** ensures that each novel prototype represents a distinct and compact semantic region, significantly improving the discriminability of the model.

B. Extensive Qualitative Gallery on ScanNet200

To further illustrate the robustness of our framework, we provide additional full-scene qualitative comparisons between the state-of-the-art baseline GFS-VL [14] and HOP3D.

Improved Boundary Integrity. In the first row of Figure 1, GFS-VL struggles to distinguish between functionally related objects (e.g. , misclassifying *sink* points as *kitchen counter*). By preserving the unique geometric residuals of novel objects via **HOP-Rep**, our model maintains much sharper and more accurate transitions between adjacent categories.

Calibrated Predictions in Ambiguous Regions. The second and third rows highlight the impact of the entropy-based regularizer (**HOP-Ent**). While the baseline often exhibits "over-prediction"—erroneously labeling large background or ambiguous regions as a specific novel class—HOP3D produces more balanced and confident predictions. This calibration effectively suppresses false positives, leading to a segmentation map that is much cleaner and more visually coherent.

C. Conclusion of Analysis

The combination of representation-level interpretation and full-scene visualization provides strong evidence that HOP3D effectively addresses the stability-plasticity trade-off. By jointly controlling the subspace structure (**HOP-Rep**) and the optimization dynamics (**HOP-Grad**, **HOP-Ent**), our framework achieves superior performance in complex, generalized few-shot 3D point cloud segmentation scenarios.

References

[1] Jie Liu, Wenzhe Yin, Haochen Wang, et al., "Dynamic prototype adaptation with distillation for few-shot point cloud segmentation," in *Int. Conf. 3D Vision (3DV)*, 2024, pp. 810–819.

[2] Qianguang Zhao, Dongli Wang, Yan Zhou, et al., "Few to big: Prototype expansion network via diffusion learner for point cloud few-shot semantic segmentation," *arXiv preprint arXiv:2509.12878*, 2025.

[3] Chih-Jung Tsai, Hwann-Tzong Chen, and Tyng-Luh Liu, "Pseudo-embedding for generalized few-shot 3D segmentation," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*. 2024, pp. 383–400, Springer.

[4] David Lopez-Paz and Marc'Aurelio Ranzato, "Gradient episodic memory for continual learning," in *Adv. Neural Inf. Process. Syst. (NeurIPS)*, 2017.

[5] Mehrdad Farajtabar, Navid Azizan, Alex Mott, et al., "Orthogonal gradient descent for continual learning," in *Int. Conf. Artif. Intell. Stat. (AISTATS)*, 2020.

[6] Tianhe Yu, Saurabh Kumar, Abhishek Gupta, et al., "Gradient surgery for multi-task learning," in *Adv. Neural Inf. Process. Syst. (NeurIPS)*, 2020.

[7] Naman Bansal, Xiaohan Chen, Zhangyang Wang, et al., "Can we gain more from orthogonality regularization in convolutional neural networks?," in *Adv. Neural Inf. Process. Syst. (NeurIPS)*, 2018.

[8] Michael Cogswell, Faruk Ahmed, Ross Girshick, et al., "Reducing overfitting in deep networks by decorrelating representations," in *Int. Conf. Learn. Represent. (ICLR)*, 2016.

[9] Sun-Ao Liu, Yiheng Zhang, Zhaofan Qiu, et al., "Learning orthogonal prototypes for generalized few-shot semantic segmentation," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2023.

[10] Yves Grandvalet and Yoshua Bengio, "Semi-supervised learning by entropy minimization," in *Adv. Neural Inf. Process. Syst. (NeurIPS)*, 2004.

[11] Dequan Wang, Evan Shelhamer, Shaoteng Liu, et al., "TENT: Fully test-time adaptation by entropy minimization," in *Int. Conf. Learn. Represent. (ICLR)*, 2021.

[12] Ryan Gomes, Andreas Krause, and Pietro Perona, "Discriminative clustering by regularized information maximization," in *Adv. Neural Inf. Process. Syst. (NeurIPS)*, 2010.

[13] Xiaoyang Wu, Li Jiang, Peng-Shuai Wang, et al., "Point transformer V3: Simpler, faster, stronger," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2024.

[14] Zhaochong An, Guolei Sun, Yun Liu, et al., "Generalized few-shot 3D point cloud segmentation with vision-language model," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2025.

[15] David Rozenberszki, Or Litany, and Angela Dai, "Language-grounded indoor 3D semantic segmentation in the wild," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, 2022.

[16] Chandan Yeshwanth, Yueh-Cheng Liu, Matthias Nissner, et al., "ScanNet++: A high-fidelity dataset of 3D indoor scenes," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, 2023.