HYPERCAPE: LEARNING HIGHER-ORDER KEYPOINT STRUCTURES FOR CATEGORY-AGNOSTIC POSE ESTIMATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Category-Agnostic Pose Estimation (CAPE) aims to localize semantic keypoints on unseen object categories. Recent CAPE methods increasingly exploit structural relations among keypoints, yet existing formulations predominantly rely on pairwise graphs, which do not explicitly represent group-wise structures jointly formed by multiple semantic keypoints. We propose **HyperCape**, a higher-order keypoint modeling framework that induces a latent hypergraph from contextualized keypoint representations. HyperCape sparsely assigns each keypoint to a subset of latent hyperedges, allowing related keypoints to form shared structural groups while permitting each keypoint to participate in multiple relations. The learned hypergraph not only enables higher-order feature interactions, but also directly guides coordinate refinement by modeling the relative geometry of keypoints within each hyperedge. Experiments on the MP-100 benchmark show that HyperCape outperforms existing graph-based CAPE methods, achieving gains of 1.04%p in mPCK and 2.05%p at PCK@0.05 over the strongest prior method. Qualitative analyses further show that the learned hyperedges form coherent higher-order groups of structurally related keypoints across unseen categories. These results highlight the benefit of learned higher-order structure for accurate category-agnostic keypoint localization. *Code will be released upon publication.*

1 INTRODUCTION

Pose estimation localizes semantically meaningful object keypoints and supports diverse vision applications, including human motion analysis, animal behavior understanding, autonomous systems, and robotic manipulation (Song et al., 2021; Mathis et al., 2018; Song et al., 2019; Liu et al., 2020b). Conventional pose estimation models, however, are typically designed for predefined object categories with fixed keypoint definitions. Applying such models to a new category therefore requires additional category-specific annotations and model adaptation, limiting their scalability to diverse real-world objects. Category-Agnostic Pose Estimation (CAPE) addresses this limitation by learning a unified model that localizes semantic keypoints for object categories unseen during training (Xu et al., 2022a). By extending pose estimation beyond fixed category and keypoint definitions, CAPE provides a more flexible framework for keypoint localization across heterogeneous object categories.

The main challenge in CAPE lies in localizing arbitrary semantic keypoints whose appearance and configuration can vary substantially across unseen categories. Early approaches specify target keypoints through annotated support images and localize them by feature matching or correspondence learning (Xu et al., 2022a; Shi et al., 2023; Chen et al., 2024). More recent methods improve the semantic specification of target keypoints using textual descriptions, instead of visual support features (Rusanovsky et al., 2025; Zhu et al., 2026; Kim et al., 2025). However, accurately localizing a keypoint depends not only on recognizing its individual semantics, but also on understanding how it is related to the other keypoints of the object. Accordingly, recent CAPE methods increasingly incorporate structural reasoning through pairwise graphs. GraphCape (Hirschorn & Avidan, 2024) introduces a manually specified keypoint graph, while subsequent approaches refine or infer pairwise relations from visual observations (Ren et al., 2024; Hirschorn & Avidan, 2026; Rao et al., 2026). Text-based methods such as CapeX (Rusanovsky et al., 2025) and CapeNext (Zhu et al., 2026) similarly combine semantic descriptions with pairwise structural reasoning.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

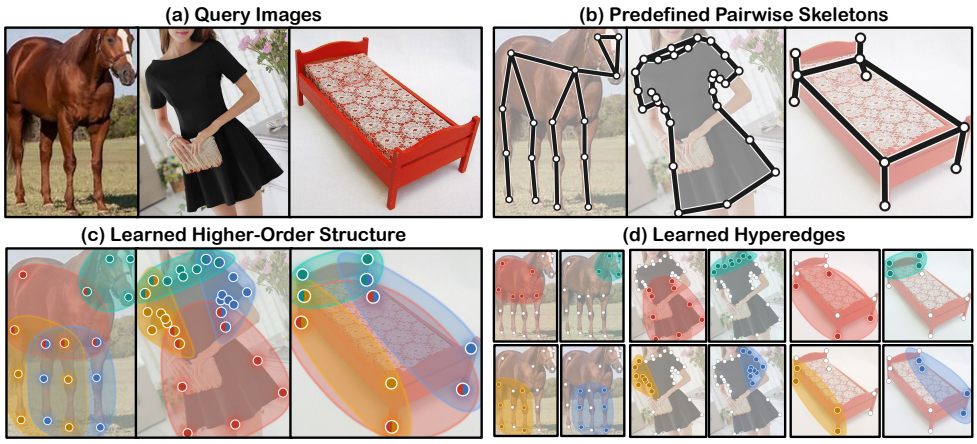


Figure 1: **Pairwise and higher-order structural modeling for CAPE.** A predefined pairwise skeleton represents keypoint dependencies through relations between keypoint pairs. HyperCape instead learns higher-order groups that explicitly organize multiple related keypoints into shared structural contexts.

Notice that pairwise relations provide a natural way to describe local dependencies, but they do not explicitly represent structures jointly formed by multiple semantic keypoints. Such group-wise organization is more naturally represented by higher-order relations that treat multiple keypoints as a shared structural unit (Fig. 1). However, the exact composition of these groups varies across object categories because different categories contain different keypoint sets and semantic parts. This motivates learning higher-order assignments from keypoint representations.

To this end, we propose **HyperCape**, which learns a latent hypergraph over semantic keypoints and uses the resulting higher-order structure throughout keypoint prediction. Given contextualized keypoint representations jointly encoded from the query image and textual descriptions, HyperCape sparsely assigns each keypoint to a small subset of latent hyperedges, allowing related keypoints to form shared structural groups while each keypoint may participate in multiple relations. Each hyperedge then acts as a shared structural context by aggregating information from its member keypoints and propagating the resulting higher-order representation back to them. Beyond feature reasoning, the same learned hypergraph directly guides coordinate refinement by modeling the relative geometry of keypoints within each hyperedge. The learned hypergraph therefore provides a common structural representation for both semantic feature exchange and geometric localization, without relying on a manually specified pairwise skeleton.

We evaluate HyperCape on the MP-100 benchmark (Xu et al., 2022a) under the standard category-agnostic evaluation protocol. HyperCape achieves the strongest overall results on MP-100, improving over the strongest prior method by 1.04%p in mPCK and 2.05%p at PCK@0.05. Qualitative analysis further shows that the learned hyperedges organize structurally related keypoints into coherent higher-order groups across unseen categories. These results demonstrate the effectiveness of learned higher-order structure for category-agnostic keypoint localization.

Our contributions are summarized as follows: **1)** We introduce higher-order structural modeling to CAPE, moving beyond conventional pairwise keypoint graphs to explicitly represent group-wise relations among multiple semantic keypoints. **2)** We propose HyperCape, which induces a latent hypergraph directly from contextualized keypoint representations and uses the learned structure for both higher-order feature reasoning and geometry-aware coordinate refinement, without requiring a manually specified skeleton. **3)** We demonstrate the effectiveness of HyperCape on MP-100, improving over the strongest prior method by 1.04%p in mPCK and 2.05%p at PCK@0.05, together with qualitative analyses of the learned higher-order structures.

2 Related Work

2.1 Category-Agnostic Pose Estimation

Category-Agnostic Pose Estimation (CAPE) aims to localize semantic keypoints in object categories unseen during training. Early approaches formulate CAPE as a few-shot matching problem, transferring keypoint information from support images to a query image through feature matching or

correspondence learning (Xu et al., 2022a; Shi et al., 2023; Chen et al., 2024). ESCAPE (Nguyen et al., 2024) learns cross-category super-keypoints as semantic priors for novel keypoint adaptation. Subsequent methods incorporate structural reasoning through pairwise graphs. SDPNet (Ren et al., 2024) estimates category-specific pairwise structures, while GraphCape (Hirschorn & Avidan, 2024), EdgeCape (Hirschorn & Avidan, 2026), and GenCape (Rao et al., 2026) use manually specified, refined, or learned pairwise graph structures for keypoint reasoning. Recent language-based methods replace support-image features with textual keypoint descriptions (Rusanovsky et al., 2025; Zhu et al., 2026; Kim et al., 2025), combining semantic specification with structural or instance-aware refinement. HyperCape instead learns higher-order relations among text-defined keypoints and exploits them for both feature reasoning and geometry-aware coordinate refinement, without requiring a manually specified pairwise skeleton.

2.2 Hypergraph-Based Higher-Order Modeling in Vision

Hypergraphs provide a natural representation for relations among multiple entities and have been widely studied for higher-order representation learning (Feng et al., 2019; Yadati et al., 2019; Chien et al., 2022). Recent vision models construct or learn hypergraphs over visual tokens for recognition, including patch-based hypergraphs (Han et al., 2023), multi-scale dynamic hypergraphs (Li et al., 2025), and soft or feature-dependent vertex–hyperedge connectivity (Fixelle, 2025; Lei et al., 2026). Hypergraph reasoning has also been explored for human pose estimation, where higher-order dependencies are modeled among predefined human joints (Liu et al., 2020a; Xu et al., 2022b; Zheng & Cao, 2025). These methods demonstrate the benefit of higher-order reasoning for human pose estimation, but assume a fixed set of joints and semantics. HyperCape instead considers category-agnostic keypoint localization, where the keypoint semantics and their structural relations vary across unseen object categories.

3 Method

3.1 Overview

Category-Agnostic Pose Estimation (CAPE) aims to localize semantic keypoints in a query image from an unseen object category. Following prior work (Rusanovsky et al., 2025), we represent the target keypoints using textual descriptions rather than support-image features. Given a query image $I \in \mathbb{R}^{H\times W\times 3}$ and a set of N keypoint descriptions $\mathcal{T} = \{t_i\}_{i=1}^{N}$, where t_i denotes the description of the i-th keypoint, the goal is to predict their corresponding 2D coordinates $\mathbf{P} = [\mathbf{p}_1, \ldots, \mathbf{p}_N]^\top \in \mathbb{R}^{N\times 2}$, where $\mathbf{p}_i \in \mathbb{R}^2$ denotes the predicted coordinate of the i-th keypoint. The query image and keypoint descriptions are first encoded into visual and textual features, respectively, and jointly contextualized through a Transformer encoder, allowing the visual and textual representations to incorporate information from one another. We denote the resulting visual features as $\mathbf{F} \in \mathbb{R}^{S\times d}$, where S denotes the number of visual tokens, and the contextualized keypoint representations as $\mathbf{V} = [\mathbf{v}_1, \ldots, \mathbf{v}_N]^\top \in \mathbb{R}^{N\times d}$.

As illustrated in Fig. 2, HyperCape models keypoint structure through a latent higher-order representation induced from $\mathbf{V}$, rather than pairwise relations. We sparsely assign the N keypoints to M latent hyperedges and construct an incidence matrix $\mathbf{H} \in \mathbb{R}^{N\times M}$, where H_{im} represents the learned membership between the i-th keypoint and the m-th hyperedge (Sec. 3.2). Unlike pairwise relations, each hyperedge represents multiple keypoints as a shared higher-order relation. The induced hypergraph is then used in the Transformer decoder for both feature reasoning and coordinate refinement. Keypoint representations are propagated through shared latent hyperedges (Sec. 3.3), while coordinate predictions are refined using the current relative geometry within each hyperedge (Sec. 3.4). The memberships in $\mathbf{H}$ are shared across decoder layers, whereas feature reasoning and geometric refinement are recomputed as the keypoint representations and coordinates evolve. The remaining components, including visual and textual encoding, proposal generation, and self- and cross-attention, follow CapeX (Rusanovsky et al., 2025).

3.2 Higher-Order Keypoint Modeling

As illustrated in Fig. 2(b), we learn higher-order keypoint relations by associating each contextualized keypoint representation with a subset of M latent hyperedges. Given the contextualized representation $\mathbf{v}_i \in \mathbb{R}^d$ of the i-th keypoint, we compute its hyperedge assignment logits using a

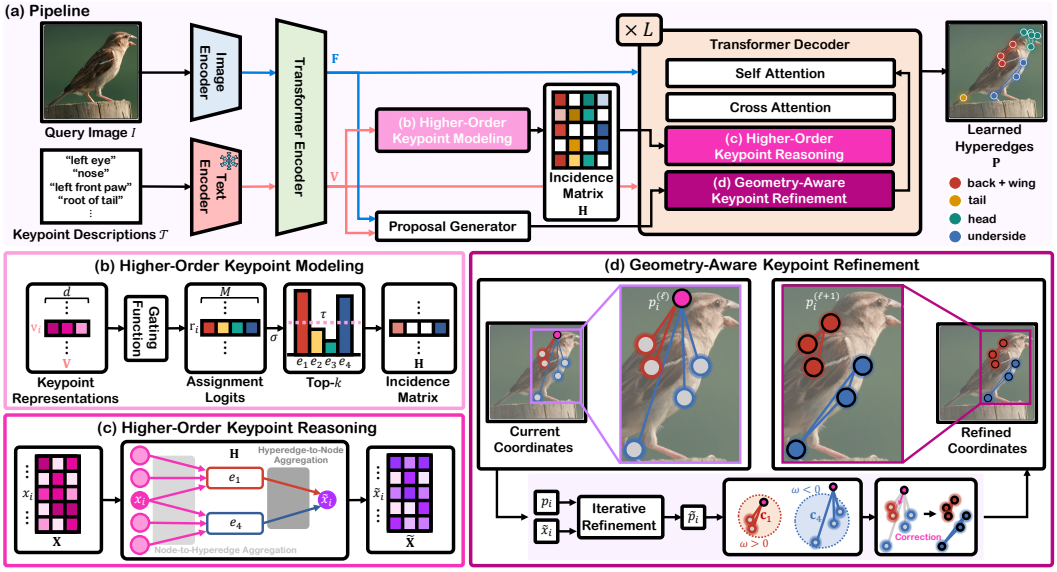


Figure 2: **Overview of HyperCape.** (a) A query image I and keypoint descriptions $\mathcal{T}$ are encoded into $\mathbf{F}$ and $\mathbf{V}$, and refined into keypoint coordinates $\mathbf{P}$ through a Transformer decoder. (b) Each contextualized keypoint representation in $\mathbf{V}$ is sparsely assigned to a subset of M hyperedges, forming the incidence matrix $\mathbf{H}$. (c) Given decoder features $\mathbf{X}^{(\ell)}$, node-to-hyperedge-to-node aggregation produces the higher-order representations $\widetilde{\mathbf{X}}^{(\ell)}$ using $\mathbf{H}$. (d) The same $\mathbf{H}$ guides geometry-aware refinement from $\tilde{\mathbf{p}}_i^{(\ell)}$ to $\mathbf{p}_i^{(\ell+1)}$ using the current hyperedge geometry and signed interactions $\omega_{ijm}^{(\ell)}$.

noisy gating function (Shazeer et al., 2017):

$$\mathbf{r}_i = \mathbf{W}_g \mathbf{v}_i + \boldsymbol{\epsilon}_i \odot \mathrm{Softplus}\left(\mathbf{W}_{\mathrm{noise}} \mathbf{v}_i\right), \tag{1}$$

where $\mathbf{W}_g, \mathbf{W}_{\mathrm{noise}} \in \mathbb{R}^{M \times d}$ are learnable projections, $\boldsymbol{\epsilon}_i \in \mathbb{R}^M$ is sampled from a standard Gaussian distribution during training, and $\odot$ denotes element-wise multiplication. The noise term is removed during inference. The corresponding hyperedge membership scores are obtained as:

$$\boldsymbol{\pi}_i = \sigma(\mathbf{r}_i) \in (0,1)^M, \tag{2}$$

where $\sigma(\cdot)$ denotes the sigmoid function and π_{im} represents the membership strength between the i-th keypoint and the m-th hyperedge. Independent sigmoid scores allow a keypoint to participate in multiple hyperedges, yielding overlapping higher-order relations. To obtain a sparse higher-order structure, we retain the top-k memberships of each keypoint and discard assignments below a threshold τ. The resulting binary mask is defined as:

$$Z_{im} = \mathbb{I}\left[m \in \mathrm{TopK}(\boldsymbol{\pi}_i, k)\right] \cdot \mathbb{I}\left[\pi_{im} \geq \tau\right], \tag{3}$$

where $\mathrm{TopK}(\boldsymbol{\pi}_i, k)$ returns the indices of the k largest membership scores and $\mathbb{I}[\cdot]$ denotes the indicator function. The resulting incidence matrix $\mathbf{H} = [H_{im}] \in \mathbb{R}^{N \times M}$ is constructed as $H_{im} = \pi_{im} Z_{im}$, where nonzero entries indicate the keypoints participating in each hyperedge and their magnitudes represent the corresponding membership strengths. For subsequent reasoning, we define $\mathcal{I}_m = \{i \mid Z_{im} = 1\}$ as the set of keypoints assigned to hyperedge m and $\mathcal{M}_i = \{m \mid Z_{im} = 1\}$ as the set of hyperedges assigned to keypoint i.

3.3 Higher-Order Keypoint Reasoning

As illustrated in Fig. 2(c), we propagate information through the induced higher-order structure using node-to-hyperedge-to-node aggregation (Chien et al., 2022). For the ℓ-th decoder layer, let $\mathbf{X}^{(\ell)} = [\mathbf{x}_1^{(\ell)}, \ldots, \mathbf{x}_N^{(\ell)}]^\top \in \mathbb{R}^{N \times d}$ denote the keypoint representations after self- and cross-attention, where $\mathbf{x}_i^{(\ell)} \in \mathbb{R}^d$ is the representation of the i-th keypoint.

Node-to-Hyperedge Aggregation. For each hyperedge m, we aggregate the representations of its member keypoints according to their learned membership strengths:

$$\mathbf{e}_m^{(\ell)} = \frac{\sum_{i \in \mathcal{I}_m} H_{im} \mathbf{x}_i^{(\ell)}}{\sum_{i \in \mathcal{I}_m} H_{im} + \delta_{\mathrm{agg}}}, \tag{4}$$

where $\delta_{\text{agg}} > 0$ is a small constant for numerical stability. The resulting hyperedge representation $\mathbf{e}_m^{(\ell)} \in \mathbb{R}^d$ summarizes the current features of the keypoints participating in the m-th higher-order relation.

Hyperedge-to-Node Propagation. We then propagate the hyperedge representations back to their member keypoints. The output of the higher-order projection is computed as:

$$\bar{\mathbf{x}}_i^{(\ell)} = \phi\left(\mathbf{W}_{\text{self}}^{(\ell)}\mathbf{x}_i^{(\ell)} + \frac{\sum_{m\in\mathcal{M}_i} H_{im}\mathbf{W}_e^{(\ell)}\mathbf{e}_m^{(\ell)}}{\sum_{m\in\mathcal{M}_i} H_{im} + \delta_{\text{agg}}}\right), \tag{5}$$

where $\mathbf{W}_{\text{self}}^{(\ell)}, \mathbf{W}_e^{(\ell)} \in \mathbb{R}^{d_{\text{ff}}\times d}$ are learnable projections, $\phi(\cdot)$ denotes the activation function, and $\bar{\mathbf{x}}_i^{(\ell)} \in \mathbb{R}^{d_{\text{ff}}}$ is the resulting higher-order representation of the i-th keypoint. This projection replaces the graph-based first projection in the decoder feed-forward module of prior works (Hirschorn & Avidan, 2024; Rusanovsky et al., 2025). The remaining feed-forward operations produce $\tilde{\mathbf{x}}_i^{(\ell)} \in \mathbb{R}^d$, which is used for subsequent coordinate refinement.

3.4 Geometry-Aware Keypoint Refinement

The higher-order reasoning module in Sec. 3.3 uses the learned hypergraph to exchange information among keypoints in feature space. We further use the same higher-order structure to refine keypoint coordinates based on the current geometry of other keypoints within each hyperedge (Fig. 2(d)). The refinement operates on the relative configuration predicted for the current query image, allowing the learned higher-order structure to directly guide coordinate updates.

Let $\mathbf{p}_i^{(\ell)}$ denote the current normalized coordinate of keypoint i entering the ℓ-th decoder layer. Following the iterative refinement scheme of prior works (Shi et al., 2023), we first obtain a feature-based coordinate prediction:

$$\tilde{\mathbf{p}}_i^{(\ell)} = \sigma\left(\sigma^{-1}\left(\mathbf{p}_i^{(\ell)}\right) + f_{\text{coord}}^{(\ell)}\left(\tilde{\mathbf{x}}_i^{(\ell)}\right)\right), \tag{6}$$

where $f_{\text{coord}}^{(\ell)} : \mathbb{R}^d \to \mathbb{R}^2$ denotes the coordinate head. We then characterize the current geometry of each hyperedge from its member keypoints. For hyperedge m, we compute its membership-weighted center and spatial spread as:

$$\mathbf{c}_m^{(\ell)} = \frac{\sum_{i\in\mathcal{I}_m} H_{im}\tilde{\mathbf{p}}_i^{(\ell)}}{\sum_{i\in\mathcal{I}_m} H_{im} + \delta_{\text{geo}}}, \qquad \rho_m^{(\ell)} = \sqrt{\frac{\sum_{i\in\mathcal{I}_m} H_{im}\left\|\tilde{\mathbf{p}}_i^{(\ell)} - \mathbf{c}_m^{(\ell)}\right\|_2^2}{\sum_{i\in\mathcal{I}_m} H_{im} + \delta_{\text{geo}}}}, \tag{7}$$

where $\delta_{\text{geo}} > 0$ is a small constant for numerical stability. For each keypoint pair (i, j) in hyperedge m, we compute a scale-normalized relative distance feature $\gamma_{ijm}^{(\ell)}$ with respect to the current hyperedge spread $\rho_m^{(\ell)}$ and use it to predict a pair-specific interaction weight $\omega_{ijm}^{(\ell)}$:

$$\gamma_{ijm}^{(\ell)} = \log\left(1 + \frac{\left\|\tilde{\mathbf{p}}_j^{(\ell)} - \tilde{\mathbf{p}}_i^{(\ell)}\right\|_2}{\rho_m^{(\ell)} + \delta_{\text{geo}}}\right), \qquad \omega_{ijm}^{(\ell)} = \tanh\left(f_{\text{geo}}^{(\ell)}\left(\tilde{\mathbf{x}}_i^{(\ell)}, \tilde{\mathbf{x}}_j^{(\ell)}, \mathbf{e}_m^{(\ell)}, \gamma_{ijm}^{(\ell)}\right)\right). \tag{8}$$

Here, $f_{\text{geo}}^{(\ell)}$ is an MLP, and $\omega_{ijm}^{(\ell)}$ determines the signed magnitude of the correction along the current displacement between the two keypoints. Because $\omega_{ijm}^{(\ell)}$ is conditioned on the hyperedge representation $\mathbf{e}_m^{(\ell)}$ and the scale-normalized distance $\gamma_{ijm}^{(\ell)}$, the same keypoint pair can interact differently under different higher-order contexts. The output layer of $f_{\text{geo}}^{(\ell)}$ is initialized to zero, so that the geometric correction is initially zero and gradually learned during training. Inspired by relative-displacement coordinate updates (Satorras et al., 2021), hyperedge m provides a geometric correction for keypoint i:

$$\Delta\mathbf{p}_{im}^{(\ell)} = \frac{\sum_{j\in\mathcal{I}_m\setminus\{i\}} H_{jm}\omega_{ijm}^{(\ell)}\left(\tilde{\mathbf{p}}_j^{(\ell)} - \tilde{\mathbf{p}}_i^{(\ell)}\right)}{\sum_{j\in\mathcal{I}_m\setminus\{i\}} H_{jm} + \delta_{\text{geo}}}. \tag{9}$$

Finally, we aggregate the geometric corrections from all hyperedges assigned to keypoint i:

$$\mathbf{p}_i^{(\ell+1)} = \tilde{\mathbf{p}}_i^{(\ell)} + \lambda_{\text{geo}} \frac{\sum_{m \in \mathcal{M}_i} H_{im} \Delta \mathbf{p}_{im}^{(\ell)}}{\sum_{m \in \mathcal{M}_i} H_{im} + \delta_{\text{geo}}}, \tag{10}$$

where λ_{geo} is a fixed hyperparameter controlling the contribution of the geometric refinement. The refined coordinates are clamped to the valid normalized range before entering the next decoder layer. Thus, the learned higher-order structure guides both feature reasoning and coordinate refinement. Further implementation details are provided in Appendix A.4.

3.5 Training Objective

Following CapeX (Rusanovsky et al., 2025), we use the localization objective $\mathcal{L}_{\text{loc}} = \lambda_{\text{heat}} \mathcal{L}_{\text{heat}} + \mathcal{L}_{\text{offset}}$. Here, $\mathcal{L}_{\text{heat}}$ supervises the similarity heatmaps produced by the proposal generator, while $\mathcal{L}_{\text{offset}}$ applies an ℓ_1 loss to the initial proposals and the refined coordinates $\mathbf{p}_i^{(\ell+1)}$ from every decoder layer. To prevent the learned assignments from concentrating on a small number of hyperedges, we additionally regularize hyperedge utilization. For a mini-batch $\mathcal{B}$, let N_b denote the number of valid keypoints in sample b. We define the utilization of hyperedge m as $c_m = \sum_{b \in \mathcal{B}} \sum_{i=1}^{N_b} H_{im}^{(b)}$ and let $\mathbf{c} = [c_1, \ldots, c_M]^\top$. The balancing loss is defined using the squared coefficient of variation:

$$\mathcal{L}_{\text{bal}} = \frac{\text{Var}(\mathbf{c})}{\text{Mean}(\mathbf{c})^2 + \delta_{\text{bal}}}, \tag{11}$$

where $\delta_{\text{bal}} > 0$ is a small constant for numerical stability. The final objective is:

$$\mathcal{L} = \mathcal{L}_{\text{loc}} + \lambda_{\text{bal}} \mathcal{L}_{\text{bal}}, \tag{12}$$

where λ_{bal} controls the strength of the balancing regularization.

4 Experiment

4.1 Experimental Setup

Dataset and Metrics. We evaluate HyperCape on MP-100 (Xu et al., 2022a), which contains over 18K images and 20K annotations spanning 100 object categories, with 8 to 68 keypoints per category. Following the standard protocol, the categories are divided into 70 training, 10 validation, and 20 test categories without overlap, with five predefined splits. We use the keypoint text descriptions of Rusanovsky et al. (2025). For comparability with prior CAPE methods, we follow the standard 1-shot evaluation protocol and sample 200 episodes with 15 queries each for every split. HyperCape does not consume support images; therefore, increasing the number of support examples does not change its input, and we do not separately report 5-shot results. We report the standard PCK@0.2 metric and additionally evaluate PCK at stricter thresholds $\{0.05, 0.1, 0.15, 0.2\}$, whose average is denoted as mPCK, since PCK@0.2 is largely saturated on MP-100. We identify several inconsistent or duplicated textual keypoint descriptions and correct them at the category level before evaluation. The corrected descriptions are used for training and evaluation of HyperCape and the reproduced text-based baselines. The changes are detailed in Appendix D.

Baselines. We use SwinV2-T (Liu et al., 2022) as the visual backbone for HyperCape and compare against image-support methods: POMNet (Xu et al., 2022a), CapeFormer (Shi et al., 2023), ESCAPE (Nguyen et al., 2024), MetaPoint+ (Chen et al., 2024), SDPNet (Ren et al., 2024), X-Pose (Yang et al., 2024), SCAPE (Liang et al., 2024), PPM+CPT (Peng et al., 2024), GraphCape (Hirschorn & Avidan, 2024), EdgeCape (Hirschorn & Avidan, 2026), GenCape (Rao et al., 2026); and text-support methods: CapeX (Rusanovsky et al., 2025), CapeNext (Zhu et al., 2026). For Tab. 1, the reported baseline results follow GenCape (Rao et al., 2026). CapeX, CapeNext, and EdgeCape, marked with †, are reproduced in this paper; details are provided in Appendix A.5. For the stricter-threshold comparison in Tab. 2, we focus on CapeFormer and methods that explicitly model inter-keypoint structure, i.e., GraphCape, CapeNext, CapeX, EdgeCape, and GenCape.

Implementation Details. HyperCape follows the training and data-processing protocol of prior graph-based CAPE methods (Hirschorn & Avidan, 2024; Rusanovsky et al., 2025) and uses SwinV2-T (Liu et al., 2022) as the visual backbone. The text encoder is a frozen gte-base-en-v1.5 (Li et al., 2023), and the Transformer uses a model dimension of $d = 256$ with three encoder and three decoder layers. The hypergraph contains $M = 8$ latent hyperedges with top-k assignment at $k = 2$,

Table 1: **Comparison with state-of-the-art methods**: PCK@0.2 under the 1-shot setting. Reported baseline results follow GenCape (Rao et al., 2026) (†: reproduced in this work). 'Graph' denotes an external graph prior, while '(Graph)' denotes a learned graph. The best, second-best, and third-best results are highlighted.

Method	Support	Split 1	Split 2	Split 3	Split 4	Split 5	Avg.
POMNet (Xu et al., 2022a)	Image	84.23	78.25	78.17	78.68	79.17	79.70
CapeFormer (Shi et al., 2023)	Image	89.48	86.69	85.31	84.79	84.97	86.25
ESCAPE (Nguyen et al., 2024)	Image	86.89	82.55	81.25	81.72	81.32	82.74
MetaPoint+ (Chen et al., 2024)	Image	90.43	85.59	84.52	84.34	85.96	86.17
SDPNet (Ren et al., 2024)	Image	91.54	86.72	85.49	85.77	87.26	87.36
X-Pose (Yang et al., 2024)	Image	89.07	85.05	85.26	85.52	85.79	86.14
SCAPE (Liang et al., 2024)	Image	91.67	86.87	87.29	85.01	86.92	87.55
PPM+CPT (Peng et al., 2024)	Image	91.03	88.06	84.48	86.73	87.40	87.54
GraphCape (Hirschorn & Avidan, 2024)	Image + Graph	91.19	87.81	85.68	85.87	85.61	87.23
CapeX† (Rusanovsky et al., 2025)	Text + Graph	92.60	88.59	85.87	87.28	89.01	88.67
CapeNext† (Zhu et al., 2026)	Text + Graph	92.06	88.23	85.89	86.59	88.21	88.20
EdgeCape† (Hirschorn & Avidan, 2026)	Image + Graph	91.22	86.69	86.15	85.02	85.92	87.00
GenCape (Rao et al., 2026)	Image (Graph)	92.05	88.69	86.89	85.88	87.02	88.09
HyperCape (Ours)	Text **(Hypergraph)**	93.07	88.68	86.42	86.74	90.16	89.01

Table 2: **Comparison with graph-based methods under stricter thresholds**: mPCK per split and average PCK@0.05 under the 1-shot setting. The best, second-best, and third-best results are highlighted.

Method	Graph Type	Support	mPCK						PCK@0.05
			Split 1	Split 2	Split 3	Split 4	Split 5	Avg.	Avg.
CapeFormer	None	Image	73.66	70.78	68.83	68.88	70.93	70.61	46.04
GraphCape	Annotated (Pairwise)	Image	75.97	72.85	69.90	70.81	71.91	72.29	47.75
CapeNext	Annotated (Pairwise)	Text	77.45	71.62	70.06	72.01	73.35	72.90	48.18
CapeX	Annotated (Pairwise)	Text	77.59	72.80	70.87	72.73	75.14	73.83	49.37
EdgeCape	Refined (Pairwise)	Image	77.18	71.79	70.99	70.76	72.76	72.70	49.39
GenCape	Learned (Pairwise)	Image	76.90	72.34	71.76	70.96	73.09	73.01	49.42
HyperCape (Ours)	Learned **(Higher-order)**	Text	79.44	73.72	71.99	72.85	76.32	74.87	51.47

membership threshold $\tau = 0.4$, geometry weight $\lambda_{\text{geo}} = 0.1$, and balancing weight $\lambda_{\text{bal}} = 0.01$. These values were selected on the MP-100 Split 1 validation categories and kept fixed across all five splits. The model is implemented in MMPose (Contributors, 2020) and trained with Adam for 200 epochs on a single NVIDIA RTX A6000 GPU. Full architecture, optimization, and additional implementation details are provided in Appendix A, with parameter-complexity analysis in Appendix C. Additional analyses of hypergraph configuration and utilization are provided in Appendix B.2. We additionally evaluate HyperCape across three random seeds in Appendix B.4 to assess training stability. Unless otherwise specified, all ablation studies are conducted on MP-100 Split 1.

4.2 Comparison with State-of-the-Art Methods

Comparison at PCK@0.2. As reported in Tab. 1, HyperCape achieves an average PCK of 89.01%, outperforming CapeX, the strongest graph-based baseline, as well as GenCape, which infers pairwise structure from the support image. Without relying on an annotated skeleton, HyperCape also outperforms methods that explicitly use one, demonstrating the effectiveness of learning higher-order relations directly from the input representations.

Comparison under stricter thresholds. Tab. 2 compares graph-based methods under stricter thresholds, reporting both mPCK and PCK@0.05, with CapeFormer included as a baseline without explicit structural modeling. While existing pairwise methods remain within a narrow mPCK range of 72.29% to 73.83%, regardless of whether their structure is annotated, refined, or learned, HyperCape reaches 74.87%. This indicates that higher-order structural modeling provides additional gains beyond variations in pairwise graph construction. Moreover, the margin over the strongest prior result increases from +0.34%p at PCK@0.2 to +2.05%p at PCK@0.05. This suggests that PCK@0.2 is largely saturated on MP-100 and that HyperCape provides a clearer advantage under stricter localization criteria. The result further supports the benefit of modeling shared higher-order relations among structurally related keypoints.

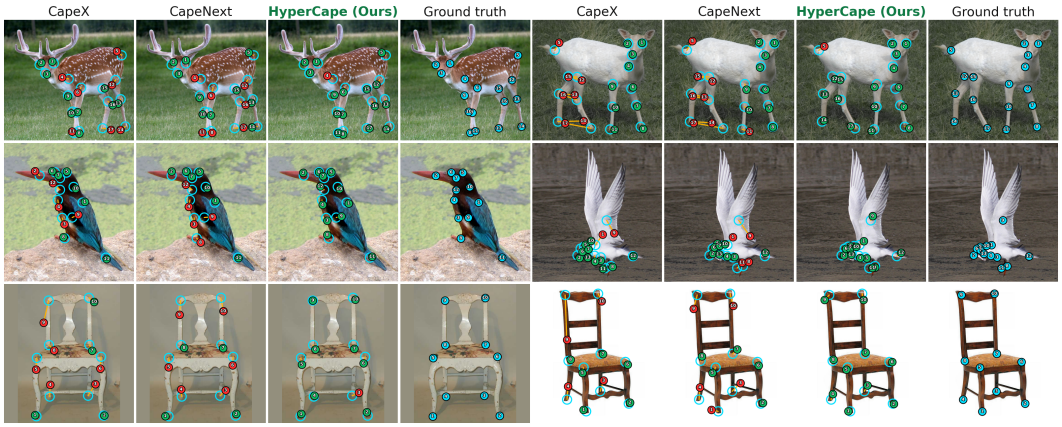


Figure 3: **Qualitative comparison on unseen MP-100 categories.** Green predictions fall within PCK@0.05 of the ground truth, red predictions do not, cyan markers denote ground-truth locations, and orange segments connect incorrect predictions to their corresponding ground truth.

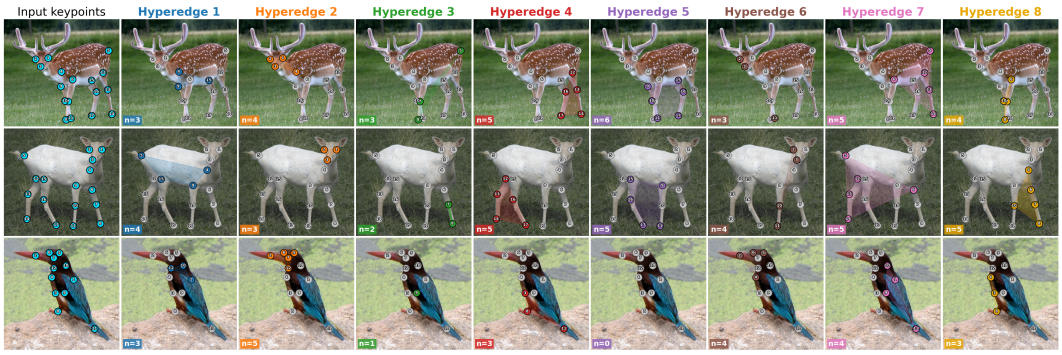


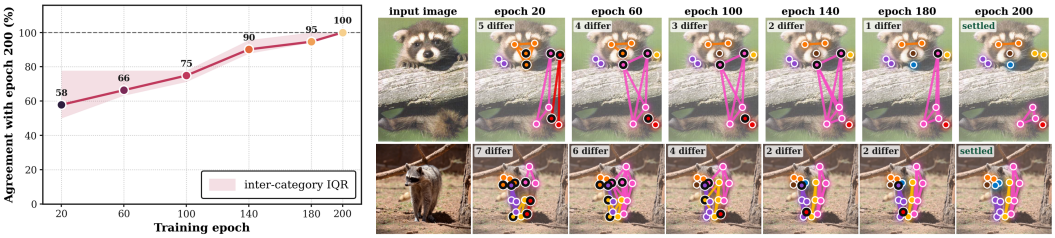
Figure 4: **Visualization of learned hyperedges.** Each column shows the keypoints assigned to one hyperedge across unseen samples. The value n denotes the number of assigned keypoints.

Qualitative Results. Fig. 3 compares HyperCape with text-supported CAPE methods, i.e., CapeX and CapeNext, on unseen MP-100 categories. HyperCape produces more accurate predictions than CapeX and CapeNext across diverse objects, indicating that higher-order structural reasoning improves localization beyond the pairwise baselines. Additional qualitative comparisons across diverse unseen categories are provided in Appendix E.

To further examine what HyperCape learns, Fig. 4 visualizes the keypoints assigned to each hyperedge across two deer instances and a kingfisher. The learned hyperedges exhibit distinct and consistent structural roles. For example, Hyperedge 2 consistently captures head-related keypoints: eyes, nose, and neck in deer, and beak, crown, forehead, eye, and throat in the kingfisher. Hyperedge 4 similarly groups lower-body or posterior regions, selecting the same five hip, knee, and back-paw keypoints in both deer instances, while focusing on the belly, leg, and tail in the bird. We also observe side-specific specialization: Hyperedge 5 predominantly selects right-side deer keypoints, whereas Hyperedge 7 focuses on left-side keypoints. Notably, Hyperedge 5 is unused for the kingfisher shown in left profile, where right-side keypoints are not visible. These examples suggest that the learned hyperedges capture reusable higher-order structural patterns across different instances, keypoint vocabularies, and object categories. Appendix B.6–B.7 further shows that these structural roles are reused across diverse categories while allowing selective instance-level variation.

4.3 Ablation and Analysis

Effect of HyperCape Components. We use a graph-free Base model in which the structural projection of the decoder feed-forward block is replaced by a standard two-layer FFN, removing the incidence matrix, gating, and structural message passing while retaining comparable parameter capacity. Tab. 3 evaluates the contributions of higher-order reasoning and geometry-aware refinement on MP-100 Split 1. The Base model achieves 77.29 mPCK and 52.37 PCK@0.05. Adding higher-



(a) Agreement with final assignment (b) Assignment changes over training

Figure 5: **Evolution of learned hyperedge assignments during training. (a)** Percentage of keypoints whose highest-weight hyperedge, i.e., the hyperedge with the largest H_{im}, matches that at epoch 200. The shaded region shows the interquartile range (IQR) across categories. **(b)** Assignment evolution for two raccoon queries. Each keypoint is colored by the hyperedge with the largest membership weight H_{im}, while black rings indicate keypoints whose highest-weight hyperedge differs from that at epoch 200.

Table 3: **Ablation of HyperCape components.**

	Higher-order reasoning	Geometry refinement	mPCK	PCK@0.05
Base			77.29	52.37
w/ H.O.	✓		78.53	54.58
w/ Geo.		✓	77.95	53.53
HyperCape	✓	✓	**79.44**	**55.45**

Table 4: **Ablation of structural modeling.**

Structure	mPCK	PCK@0.05
Base	77.29	52.37
Random H	76.98	51.77
Text k-means H	78.25	53.73
Learned pairwise	78.69	54.47
HyperCape	**79.44**	**55.45**

order reasoning improves these scores to 78.53 and 54.58, while geometry-aware refinement alone reaches 77.95 and 53.53. Combining both components yields the best performance of 79.44 mPCK and 55.45 PCK@0.05, showing that higher-order feature reasoning and geometric refinement provide complementary benefits. Further quantitative and qualitative analyses of the geometric refinement are provided in Appendix B.3.

Effect of Structural Modeling. We further examine whether the gain arises specifically from learned higher-order structure by comparing HyperCape with alternative structural modeling strategies under the same backbone and decoder framework. Random H uses a random hypergraph, whereas Text k-means H constructs a category-specific hypergraph from frozen keypoint text embeddings. Learned pairwise induces an instance-dependent K-NN graph from the same contextualized keypoint representations used by HyperCape, with $K = 7$ chosen to approximately match its effective neighborhood size. As shown in Tab. 4, Random H slightly underperforms the Base model, indicating that arbitrary higher-order interactions are not sufficient. Text k-means H benefits from semantic grouping, while Learned pairwise further improves performance through input-dependent structural modeling. HyperCape achieves the strongest result, demonstrating the benefit of learned higher-order modeling over these alternative structural strategies.

Evolution of Hyperedge Assignments. We next examine how the learned hyperedge assignments evolve during training. As shown in Fig. 5a, the highest-weight hyperedge of each keypoint progressively approaches its final assignment, with agreement increasing from 57.8% at epoch 20 to 95% at epoch 180. Fig. 5b illustrates the same trend for two raccoon queries, where initially varying assignments gradually settle into stable hyperedge groups. These results show that the learned higher-order structure progressively stabilizes during training rather than remaining arbitrary or highly variable.

5 Conclusion

We proposed HyperCape, a category-agnostic pose estimation framework that learns higher-order relations among semantic keypoints through a learned hypergraph. By grouping related keypoints into shared hyperedges, HyperCape enables higher-order feature reasoning and geometry-aware coordinate refinement without relying on a predefined pairwise skeleton. Experiments on MP-100 show that HyperCape outperforms existing graph-based CAPE methods, with larger gains under stricter localization thresholds. Qualitative analyses further show that the learned hyperedges form coherent structural groups across diverse unseen samples and categories. These results demonstrate the benefit of higher-order structural modeling over pairwise keypoint graphs for category-agnostic pose estimation.

AI USE STATEMENT

We used generative AI tools to assist with writing and language polishing, including improving clarity, grammar, and presentation of the manuscript. We also used generative AI tools for retrieval and discovery purposes, such as identifying potentially relevant related work and literature. All retrieved references and AI-assisted text were manually reviewed and verified by the authors before inclusion. We did not use generative AI tools to generate research ideas, conduct experiments, create synthetic datasets, or derive mathematical claims. The authors take full responsibility for the final content of this work, including all text, claims, analyses, and citations.

ETHICS STATEMENT

This work uses publicly available benchmark data and does not involve human-subject experiments, personally identifiable information, or the collection of new sensitive data. We do not release new datasets or annotations containing private information. While category-agnostic pose estimation may be applied in a variety of downstream settings, our work focuses on methodological development and benchmark evaluation.

REPRODUCIBILITY STATEMENT

To ensure reproducibility, we provide the dataset and evaluation protocol in Sec. 4.1, and comprehensive implementation details, including architecture, optimization, augmentation, and baseline reproduction, in Appendix A. The corrected textual keypoint descriptions are listed in Appendix D, with additional ablations and computational analyses provided in Appendices B and C. Upon acceptance, we will release the full source code and configuration files required to reproduce our results.

REFERENCES

Junjie Chen, Jiebin Yan, et al. Meta-point learning and refining for category-agnostic pose estimation. In *Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2024.

Eli Chien, Chao Pan, et al. You are allset: A multiset function framework for hypergraph neural networks. In *International Conference on Learning Representations (ICLR)*, 2022.

MMPose Contributors. Openmmlab pose estimation toolbox and benchmark, 2020. URL `https://github.com/open-mmlab/mmpose`.

Yifan Feng, Haoxuan You, et al. Hypergraph neural networks. In *AAAI conference on artificial intelligence (AAAI)*, 2019.

Joshua Fixelle. Hypergraph vision transformers: Images are more than nodes, more than edges. In *Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2025.

Yan Han, Peihao Wang, et al. Vision HGNN: An image is more than a graph of nodes. In *International Conference on Computer Vision (ICCV)*. IEEE, 2023.

Or Hirschorn and Shai Avidan. A graph-based approach for category-agnostic pose estimation. In *European Conference on Computer Vision (ECCV)*. Springer, 2024.

Or Hirschorn and Shai Avidan. Edgecape: Edge weight prediction for category-agnostic pose estimation. In *International Conference on Learning Representations (ICLR)*, 2026.

Junho Kim, Hyungjin Chung, et al. CapeLLm: Support-free category-agnostic pose estimation with multimodal large language models. In *International Conference on Computer Vision (ICCV)*. IEEE, 2025.

Mengqi Lei, Yihong Wu, et al. Softhgnn: Soft hypergraph neural networks for general visual recognition. *International Journal of Computer Vision (IJCV)*, 134(5):244, 2026.

Caoshuo Li, Tanzhe Li, et al. Dvhgnn: Multi-scale dilated vision hgnn for efficient vision recognition. In *Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2025.

Zehan Li, Xin Zhang, et al. Towards general text embeddings with multi-stage contrastive learning. *arXiv preprint arXiv:2308.03281*, 2023.

Yujia Liang, Zixuan Ye, et al. Scape: A simple and strong category-agnostic pose estimator. In *European Conference on Computer Vision (ECCV)*. Springer, 2024.

Shengyuan Liu, Pei Lv, et al. Semi-dynamic hypergraph neural network for 3d pose estimation. In *International Joint Conference on Artificial Intelligence (IJCAI)*, 2020a.

Xingyu Liu, Rico Jonschkowski, et al. Keypose: Multi-view 3d labeling and keypoint estimation for transparent objects. In *Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2020b.

Ze Liu, Han Hu, et al. Swin transformer v2: Scaling up capacity and resolution. In *Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2022.

Alexander Mathis, Pranav Mamidanna, et al. Deeplabcut: markerless pose estimation of user-defined body parts with deep learning. *Nature neuroscience*, 21(9):1281–1289, 2018.

Khoi Duc Nguyen, Chen Li, et al. Escape: Encoding super-keypoints for category-agnostic pose estimation. In *Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2024.

Duo Peng, Zhengbo Zhang, et al. Harnessing text-to-image diffusion models for category-agnostic pose estimation. In *European Conference on Computer Vision (ECCV)*. Springer, 2024.

Jiyong Rao, Yu Wang, et al. Gencape: Structure-inductive generative modeling for category-agnostic pose estimation. In *International Conference on Learning Representations (ICLR)*, 2026.

Pengfei Ren, Yuanyuan Gao, et al. Dynamic support information mining for category-agnostic pose estimation. In *Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2024.

Matan Rusanovsky, Or Hirschorn, et al. Capex: Category-agnostic pose estimation from textual point explanation. In *International Conference on Learning Representations (ICLR)*, 2025.

Vıctor Garcia Satorras, Emiel Hoogeboom, et al. E(n) equivariant graph neural networks. In *International Conference on Machine Learning (ICML)*. PMLR, 2021.

Noam Shazeer, Azalia Mirhoseini, et al. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. In *International Conference on Learning Representations (ICLR)*, 2017.

Min Shi, Zihao Huang, et al. Matching is not enough: A two-stage framework for category-agnostic pose estimation. In *Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2023.

Liangchen Song, Gang Yu, et al. Human pose estimation and its application to action recognition: A survey. *Journal of Visual Communication and Image Representation*, 76:103055, 2021.

Xibin Song, Peng Wang, et al. Apollocar3d: A large 3d car instance understanding benchmark for autonomous driving. In *Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2019.

Lumin Xu, Sheng Jin, et al. Pose for everything: Towards category-agnostic pose estimation. In *European Conference on Computer Vision (ECCV)*. Springer, 2022a.

Xixia Xu, Qi Zou, et al. Adaptive hypergraph neural network for multi-person pose estimation. In *AAAI Conference on Artificial Intelligence (AAAI)*, 2022b.

Naganand Yadati, Madhav Nimishakavi, et al. Hypergcn: A new method for training graph convolutional networks on hypergraphs. *Advances in Neural Information Processing Systems (NeurIPS)*, 2019.

Jie Yang, Ailing Zeng, et al. X-pose: Detecting any keypoints. In *European Conference on Computer Vision (ECCV)*. Springer, 2024.

Siyuan Zheng and Weiqun Cao. Ehgformer: An efficient hypergraph-injected transformer for 3d human pose estimation. *Image and Vision Computing*, 154:105425, 2025.

Yu Zhu, Dan Zeng, et al. Capenext: Rethinking and refining dynamic support information for category-agnostic pose estimation. In *AAAI Conference on Artificial Intelligence (AAAI)*, 2026.

APPENDIX

A IMPLEMENTATION DETAILS

This section provides additional architectural, training, and implementation details of HyperCape.

A.1 ARCHITECTURE AND TRAINING CONFIGURATION

Given a query image and a set of textual keypoint descriptions, HyperCape extracts visual and textual features using SwinV2-T (Liu et al., 2022) and a frozen gte-base-en-v1.5 (Li et al., 2023), respectively. The two feature streams are projected to a common embedding space and jointly contextualized by a Transformer encoder. Following CapeX (Rusanovsky et al., 2025), the contextualized representations are used to construct keypoint similarity heatmaps and initial localization proposals, which are subsequently refined by a three-layer Transformer decoder. HyperCape modifies this pipeline by inducing a latent hypergraph from the contextualized keypoint representations and using it for both higher-order feature reasoning and geometry-aware coordinate refinement. The remaining components follow CapeX. Tab. 5 summarizes the architecture and training configuration used in our experiments.

Table 5: **Architecture and training configuration of HyperCape.**

Configuration	Value
Input resolution	256×256
Visual backbone	SwinV2-T, ImageNet pretrained and fine-tuned
Text backbone	gte-base-en-v1.5, frozen
Model dimension d	256
FFN dimension d_{ff}	768
Encoder / decoder layers	3 / 3
Dropout	0.1
Number of hyperedges M	8
Top-k routing	2
Membership threshold τ	0.4
Geometry weight λ_{geo}	0.1
Balancing weight λ_{bal}	0.01
Optimizer	Adam
Initial learning rate	1×10^{-5}
Weight decay	0
Batch size	16
Training epochs	200
Learning-rate decay	$\times 0.1$ at epochs 160 and 180
Gradient clipping	None
Main-run random seed	1
Hardware	NVIDIA RTX A6000

The SwinV2-T backbone is initialized from ImageNet-pretrained weights and optimized jointly with the trainable components of HyperCape. We use the standard SwinV2-T configuration with embedding dimension 96, stage depths $[2, 2, 6, 2]$, attention heads $[3, 6, 12, 24]$, window size 16, and drop-path rate 0.2. The gte-base-en-v1.5 text encoder remains frozen throughout training.

During training, scale augmentation is sampled from a Gaussian perturbation with standard deviation 0.15 and clipped to $[0.85, 1.15]$. Rotation is sampled analogously and clipped to $[-15°, 15°]$, with rotation applied with probability 0.6. No horizontal flipping is used because it would alter the semantics of laterality-sensitive descriptions such as “left eye” and “right eye”. Images are normalized using ImageNet statistics with mean $(0.485, 0.456, 0.406)$ and standard deviation

$(0.229, 0.224, 0.225)$. Target heatmaps are generated with Gaussian standard deviation $\sigma = 1$. Validation and test use the same preprocessing pipeline without random scale and rotation.

A.2 Hypergraph Induction and Decoder Integration

Hypergraph induction is performed once from the contextualized encoder representations $\mathbf{V}$. The hyperedge assignment parameters are shared across samples, while the resulting memberships are computed independently for each sample from its contextualized keypoint representations. For the default configuration, we use $M = 8$, top-k assignment with $k = 2$, and membership threshold $\tau = 0.4$. The noisy gating term is enabled during training and disabled during inference.

The incidence matrix $\mathbf{H}$ is constructed once before decoding and shared across all three decoder layers. Thus, the structural memberships remain fixed during iterative refinement, while hyperedge representations and geometric interactions are recomputed from the current keypoint representations and coordinates at each layer. The parameters of the higher-order reasoning and geometry-aware refinement modules are independently learned for each decoder layer.

Within each decoder layer, the higher-order reasoning module replaces the graph-based structural projection in the feed-forward block of CapeX. Self- and cross-attention follow CapeX, while the resulting coordinate prediction is further adjusted by the proposed geometry-aware refinement.

A.3 Sparse Assignment and Validity Handling

MP-100 categories contain different numbers of semantic keypoints, and keypoint sets are padded to a common maximum cardinality of 100 within a mini-batch. For sample b, we denote the number of valid keypoints by N_b. Padded keypoints are masked before hypergraph induction and do not participate in hyperedge assignment, higher-order reasoning, geometric refinement, or the balancing objective. For each valid keypoint, we retain the top-k sigmoid membership scores that exceed the threshold τ:

$$Z_{im} = \mathbb{I}\left[m \in \mathrm{TopK}(\boldsymbol{\pi}_i, k)\right] \cdot \mathbb{I}\left[\pi_{im} \geq \tau\right]. \tag{13}$$

The weighted incidence matrix is constructed as $H_{im} = \pi_{im} Z_{im}$. The binary mask determines the support of higher-order propagation, while the continuous memberships H_{im} weight both node-to-hyperedge aggregation and hyperedge-to-node propagation.

For mixed-category mini-batches, hyperedge utilization is accumulated only over valid keypoints:

$$c_m = \sum_{b \in \mathcal{B}} \sum_{i=1}^{N_b} H_{im}^{(b)}. \tag{14}$$

The resulting utilization vector $\mathbf{c} = [c_1, \ldots, c_M]^\top$ is used to compute the balancing loss in Sec. 3.5. We use $\delta_{\mathrm{agg}} = 10^{-6}$ for higher-order aggregation and $\delta_{\mathrm{bal}} = 10^{-10}$ for the balancing objective.

A.4 Geometry-Aware Refinement Details

The geometry-aware refinement module is applied after the feature-based coordinate prediction at each decoder layer. The incidence matrix $\mathbf{H}$ remains fixed across decoder layers, while the hyperedge geometry and pairwise interactions are recomputed from the current keypoint representations and coordinates. We use $\lambda_{\mathrm{geo}} = 0.1$ and implement $f_{\mathrm{geo}}^{(\ell)}$ as a two-layer MLP with hidden dimension 64 and independent parameters for each decoder layer.

Coordinate Stability. We use $\delta_{\mathrm{geo}} = 10^{-6}$ in the geometric normalization terms. Since iterative refinement applies the inverse sigmoid $\sigma^{-1}(\cdot)$, we constrain normalized coordinates to remain within $[\epsilon_{\mathrm{geo}}, 1 - \epsilon_{\mathrm{geo}}]$ using $\epsilon_{\mathrm{geo}} = 10^{-3}$. A straight-through clamp is applied after each geometry-refinement step so that the forward coordinates remain valid while gradients continue to propagate through boundary predictions.

A.5 Baseline Reproduction

For the PCK@0.2 comparison in Tab. 1, we use the benchmark results reported in GenCape (Rao et al., 2026) when available. Because the original textual keypoint descriptions include several

inconsistent or duplicated entries, we retrain CapeX and CapeNext using the corrected descriptions in Appendix D. Since EdgeCape is not reported in the GenCape benchmark, we reproduce it with SwinV2-T using its official implementation. These reproduced results are marked with † in Tab. 1.

For the stricter-threshold comparison in Tab. 2, all methods use SwinV2-T as the visual backbone. CapeFormer, GraphCape, CapeX, CapeNext, and EdgeCape are reproduced using their official implementations. Since no official implementation of GenCape is publicly available, we reimplement it following the architecture and training procedure described in the original paper.

CapeX and HyperCape use frozen gte-base-en-v1.5 text features, whereas CapeNext retains CLIP ViT-B/32 to preserve its original CLIP-based multimodal architecture. To control for this difference, we additionally compare CapeX, CapeNext, and HyperCape using the same CLIP ViT-B/32 text encoder in Appendix B.1.

A.6 Structural Modeling Ablations

Text k-means Hypergraph. For the Text k-means H variant, we construct a category-specific fixed hypergraph directly from the keypoint descriptions. We encode each description using the frozen gte-base-en-v1.5 text encoder and ℓ_2-normalize the resulting embeddings. For each category, we perform spherical k-means with $M = 8$ centroids and assign each keypoint to its two nearest centroids. The corresponding membership weights are obtained by mapping cosine similarity to $[0, 1]$ as $(\cos(\cdot, \cdot) + 1)/2$. The resulting incidence matrix is fixed for all instances and epochs of the category and is also used unchanged at test time. This variant contains no learnable gating parameters and does not use the balancing loss. For categories with fewer than M keypoints, only the available clusters are populated and the remaining hyperedges are left empty.

Learned Pairwise Graph. For the Learned pairwise variant in Tab. 4, we replace the learned hypergraph with an instance-dependent pairwise graph while preserving both structural feature reasoning and geometry-aware refinement. The graph is induced from the same contextualized keypoint representations $\mathbf{V}$ used for hypergraph induction. We ℓ_2-normalize the keypoint features, compute pairwise cosine similarities, exclude self-connections and padded keypoints, and retain the top-K neighbors of each keypoint. The resulting graph is symmetrized by taking the union of directed neighborhoods and row-normalized before message passing. We use $K = 7$, chosen to approximately match the effective neighborhood size of HyperCape. With this setting, the induced pairwise graph has an average of 7.64 neighbors per keypoint after symmetrization, compared with 6.97 unique co-members reachable through the learned hyperedges of HyperCape. The graph is constructed once from the encoder output for each instance and shared across decoder layers.

For feature reasoning, the hypergraph propagation in Sec. 3.3 is replaced by the same GCN operator used by CapeX, with the induced adjacency matrix serving as the pairwise structure. For geometry-aware refinement, we construct a pairwise counterpart of Sec. 3.4 using the graph neighborhood $\mathcal{N}(i)$ in place of hyperedge membership. For each keypoint i, the neighborhood representation is computed as an adjacency-weighted average of neighboring features, and the geometric center and spread are computed over $\mathcal{N}(i)$. The relative-distance feature γ_{ij} and interaction weight ω_{ij} follow the same formulation as in HyperCape. The resulting coordinate correction is aggregated using the pairwise adjacency weights A_{ij} instead of the hypergraph memberships H_{jm}. All remaining refinement settings, including zero initialization of the output layer and the straight-through coordinate clamp, are kept identical.

B Additional Ablation and Analysis

B.1 Controlled Comparisons across Feature Encoders

To examine whether the improvement of HyperCape is attributable to the choice of visual or textual feature encoder, we conduct two controlled comparisons on MP-100 Split 1. The first controls the text encoder across CapeX, CapeNext, and HyperCape, while the second varies the visual backbone while keeping the textual representation fixed.

Text Encoder Control. CapeX and HyperCape use gte-base-en-v1.5 in the main experiments, whereas CapeNext retains CLIP ViT-B/32 to preserve its original CLIP-based multimodal architecture. To control for this difference, we additionally use CLIP ViT-B/32 for CapeX and HyperCape, matching the text encoder used by CapeNext, while keeping the visual backbone fixed to SwinV2-T. As shown in Tab. 6, HyperCape consistently outperforms CapeX and CapeNext across all PCK thresholds. The gain over CapeX is particularly pronounced under stricter localization criteria, increasing from 0.59%p at PCK@0.2 to 2.66%p at PCK@0.05. This result indicates that the improvement of HyperCape is not explained by its choice of text encoder.

Table 6: **Text encoder-controlled comparison on MP-100 Split 1.** CapeX, CapeNext, and HyperCape use the same CLIP ViT-B/32 text encoder and SwinV2-T visual backbone.

Method	PCK@0.05	PCK@0.10	PCK@0.15	PCK@0.20	mPCK
CapeNext	52.63	77.75	87.36	92.06	77.45
CapeX	53.33	77.85	87.19	91.89	77.56
HyperCape	**55.99**	**79.57**	**88.30**	**92.48**	**79.09**

Visual Backbone Control. We next examine whether the advantage of HyperCape persists across different visual backbones. For this comparison, we use CapeX as the direct structural counterpart of HyperCape, allowing the text encoder to remain fixed as gte-base-en-v1.5 while changing only the visual backbone. As shown in Tab. 7, HyperCape improves mPCK over CapeX with both DINOv2 ViT-S/14 and SwinV2-S. The improvement is again most evident under the strict PCK@0.05 criterion, with gains of 3.03%p and 2.69%p for DINOv2 ViT-S/14 and SwinV2-S, respectively. Together with the text-encoder control above, these results indicate that the benefit of HyperCape is not tied to a particular visual or textual feature encoder.

Table 7: **Visual backbone-controlled comparison on MP-100 Split 1.** CapeX and HyperCape use the same frozen gte-base-en-v1.5 text encoder while varying the visual backbone.

Backbone	Method	PCK@0.05	PCK@0.10	PCK@0.15	PCK@0.20	mPCK
DINOv2 ViT-S/14	CapeX	54.75	80.68	89.21	93.26	79.48
	HyperCape	**57.78**	**82.88**	**90.71**	**93.98**	**81.34**
SwinV2-S	CapeX	67.36	86.85	**93.03**	**95.60**	85.71
	HyperCape	**70.06**	**87.22**	92.74	95.49	**86.38**

B.2 Sensitivity to Hypergraph Configuration

We examine the sensitivity of HyperCape to the number of latent hyperedges M and the top-k routing parameter on MP-100 Split 1. As shown in Tab. 8, performance generally improves when moving from a single hyperedge to a moderate number of latent hyperedges, while excessively increasing M does not consistently provide further gains. The default configuration ($M = 8, k = 2$) was selected using the held-out validation categories of MP-100 Split 1, independently of the test categories reported here. Tab. 8 reports test mPCK for the same grid as a sensitivity analysis, and the selected configuration also attains the best test mPCK. Notably, $M = 8$ maintains strong performance across $k \in \{1, 2, 3, 4\}$, suggesting that the learned higher-order representation is not highly sensitive to small variations in routing sparsity around the selected configuration. These results suggest that a moderate number of latent hyperedges is sufficient, while increasing M further does not provide consistent benefits.

Table 8: Sensitivity to the number of latent hyperedges M and top-k routing on MP-100 Split 1. We report mPCK averaged over PCK thresholds $\{0.05, 0.10, 0.15, 0.20\}$. The best result is shown in bold. Dashes indicate configurations that were not evaluated.

M	$k=1$	$k=2$	$k=3$	$k=4$
1	76.74	–	–	–
4	78.83	77.86	–	–
6	77.45	78.57	78.84	–
8	79.11	**79.44**	79.02	78.97
16	77.14	78.34	78.05	77.69
32	77.36	77.63	77.93	77.89

Hyperedge Utilization. We further examine hyperedge utilization while fixing $k = 2$ and varying the number of latent hyperedges. As shown in Tab. 9, for moderate M, nearly all hyperedges are broadly reused across the unseen categories. As M increases, however, hyperedge usage becomes less uniformly distributed across categories, with reduced category coverage and lower utilization entropy at $M = 32$. This trend is consistent with the lack of further performance gains for larger M in Tab. 8.

Table 9: **Hyperedge utilization across unseen categories.** Statistics are computed on the 20 test categories of MP-100 Split 1 with $k = 2$. A hyperedge is considered active when it has at least one nonzero membership, $H_{im} > 0$. "Categories / Edge" denotes the average number of categories using each active hyperedge, and "Private" denotes hyperedges used by at most three categories. "Entropy" is the Shannon entropy of the normalized hyperedge utilization distribution, divided by $\log M$, such that 1 indicates perfectly uniform utilization.

M	Active	Categories / Edge	Private (≤ 3)	Entropy
4	4/4	20.0	0/4	0.996
6	6/6	20.0	0/6	0.995
8	8/8	19.9	0/8	0.986
16	15/16	17.8	0/16	0.930
32	30/32	12.2	3/32	0.890

B.3 Analysis of Geometry-Aware Refinement

We further analyze the geometric corrections on MP-100 Split 1 over 845,464 keypoint predictions from all 20 test categories. In Tab. 10, Δe denotes the mean reduction in localization error after geometric refinement, "Improved" the fraction of keypoints with reduced error, $|\Delta p|$ the mean correction magnitude, and "Aligned" the fraction of corrections with positive cosine similarity to the direction toward the ground-truth keypoint. Errors and correction magnitudes are measured in normalized 256×256 input-frame coordinates. At the first layer, refinement reduces the localization error by 0.42%p on average, improves 66.1% of keypoints, and yields a median cosine similarity of 0.625, with 73.3% of corrections aligned toward the ground truth. The correction magnitude decreases across layers, while the directional alignment remains positive, indicating stronger geometric adjustments in earlier stages and increasingly conservative updates as the predictions stabilize.

Table 10: **Analysis of geometry-aware refinement.**

Layer	Δe (%p)	Improved	$\lvert\Delta p\rvert$ (%)	Median $\cos\theta$	Aligned
1	+0.42	66.1%	1.38	0.625	73.3%
2	+0.20	62.4%	0.84	0.487	67.7%
3	+0.02	56.2%	0.17	0.212	57.6%

Fig. 6 further visualizes the geometric corrections across decoder layers. The refinement generally moves keypoint predictions toward their ground-truth locations, while the correction magnitude becomes smaller in later layers as the predictions stabilize.

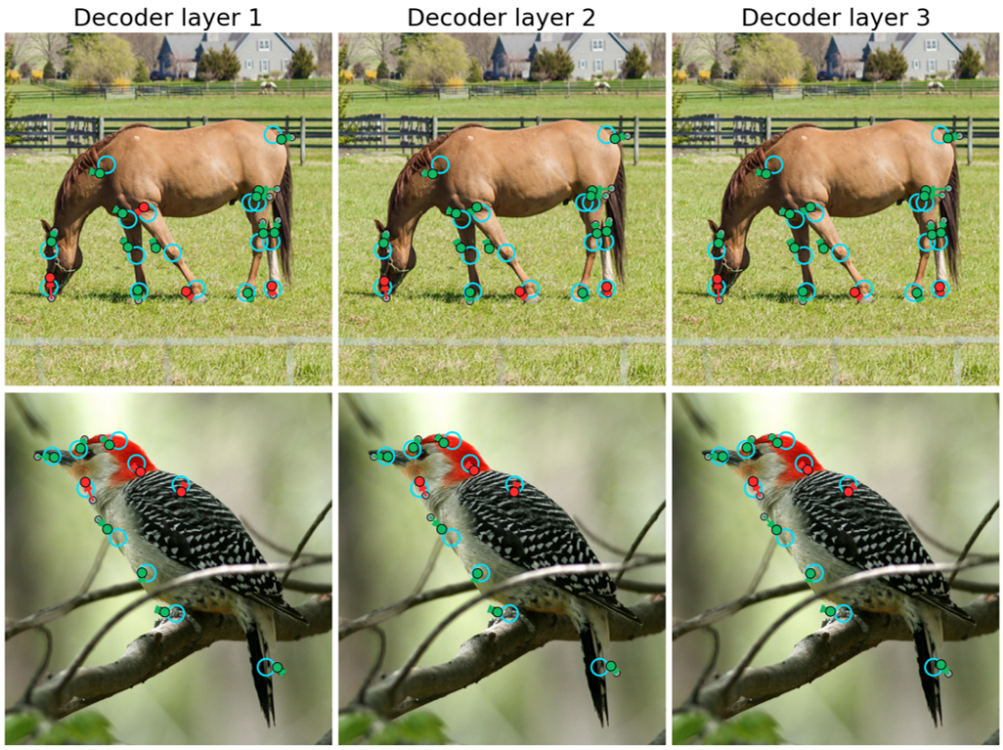


Figure 6: Visualization of geometry-aware refinement across decoder layers. Gray markers denote coordinates before geometric refinement, while green and red markers indicate corrections that move closer to and farther from the ground truth, respectively. Cyan circles denote ground-truth locations. Correction vectors are enlarged for visualization.

B.4 Robustness Across Random Seeds

To evaluate training stability, we repeat HyperCape with three random seeds for each MP-100 split while keeping all other settings fixed. As shown in Tab. 11, HyperCape exhibits low variance across all splits and PCK thresholds, indicating stable performance with respect to random initialization. The main-run seed (seed 1) is one of the three seeds reported here.

Table 11: **Robustness of HyperCape across random seeds.** We train HyperCape with three random seeds for each MP-100 split. Each entry reports the mean and standard deviation across the three runs. The last row reports the average of the split-wise means.

Split	PCK@0.05	PCK@0.10	PCK@0.15	PCK@0.20	mPCK
1	55.13±0.52	79.46±0.71	88.57±0.49	92.78±0.26	78.99±0.46
2	49.96±0.62	73.59±0.07	83.23±0.19	88.58±0.11	73.84±0.13
3	49.44±0.30	71.50±0.19	81.35±0.19	86.31±0.10	72.15±0.14
4	51.18±0.53	72.79±0.41	81.70±0.46	86.61±0.33	73.07±0.30
5	53.84±0.79	76.96±0.44	85.98±0.21	90.15±0.06	76.73±0.36
Avg.	51.91	74.86	84.17	88.89	74.96

B.5 Performance Breakdown across Test Categories

We further analyze how the performance gain of HyperCape over CapeX is distributed across different test categories under the standard MP-100 protocol. Since MP-100 defines category-disjoint train/test splits, we aggregate the test queries from all five predefined splits into nine broad object types. This analysis examines whether the overall improvement of HyperCape is concentrated in a small subset of test categories or broadly observed across different object types.

MP-100 labels animal-body categories by biological families, which we merge into a single *Animal Body* group for readability, while the remaining categories are grouped primarily by functional type. The nine groups cover all 100 categories and 300,000 evaluation queries.

As shown in Tab. 12, HyperCape outperforms CapeX in eight of the nine broad object types. The largest gains are observed for furniture and vehicle, with mPCK improvements of 4.46%p and 3.03%p, respectively. Improvements are also observed across animal bodies, birds, people, and clothing. Overall, HyperCape improves the query-weighted mPCK by 1.04%p over CapeX.

Table 12: **Performance gain over CapeX across broad MP-100 test-category types.** Δ denotes the mPCK difference between HyperCape and CapeX, where mPCK averages PCK at thresholds $\{0.05, 0.10, 0.15, 0.20\}$.

Type	# Categories	# Queries	Δ mPCK
Furniture	5	15,000	**+4.457**
Vehicle	3	9,000	**+3.026**
Hand	1	3,000	+2.244
Bird	8	24,000	+1.521
Animal Body	35	105,000	+1.283
Person	3	9,000	+1.165
Clothes	13	39,000	+0.530
Animal Face	30	90,000	+0.123
Insect	2	6,000	-0.447
Overall	**100**	**300,000**	**+1.039**

B.6 Hyperedge Assignments across Diverse Categories

Fig. 7 provides additional visualizations of the learned hyperedges across diverse unseen categories, including vehicles, faces, clothing, animal bodies, people, and insects. A particularly consistent pattern emerges for Hyperedges 5 and 7, which specialize in right- and left-side keypoints, respectively. Across all seven categories, Hyperedge 5 selects only right-side keypoints, while Hyperedge 7 selects left-side keypoints with only a few additional midline landmarks. This specialization persists despite substantial differences in keypoint vocabulary, ranging from headlights and sleeves to facial landmarks and limb joints. Other hyperedges also exhibit recurring structural roles: Hyperedge 2 tends to capture head or upper-end regions, whereas Hyperedge 4 focuses on lower-body or lower-part structures. Together, these examples show that the learned hyperedges acquire distinct structural roles that are reused across markedly different object categories.

B.7 Instance-Level Variation in Hyperedge Assignments

Although the learned hyperedges exhibit consistent structural roles across categories, their assignments are not entirely fixed within a category. To examine this behavior, we compare the dominant hyperedge assignment, $\arg\max_m H_{im}$, for keypoints visible in pairs of query images from the same category. Fig. 8 provides representative examples of these instance-level assignment changes. Across 3,800 within-category instance pairs from the 20 MP-100 Split 1 test categories, 10.7% of the keypoints visible in both images change their dominant hyperedge assignment. At the same time, 66.4% of keypoint slots retain the same dominant hyperedge across all evaluated instances of

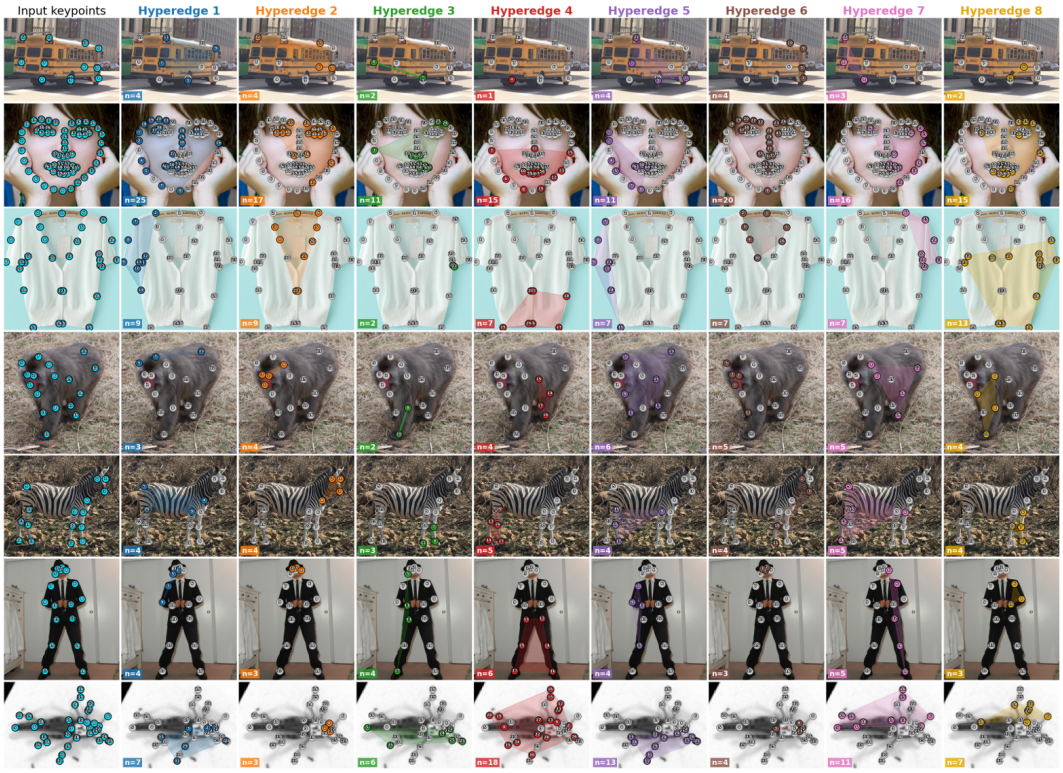


Figure 7: **Learned hyperedges across diverse unseen categories.** Each row corresponds to a different object category, and each column visualizes the keypoints assigned to one learned hyperedge. The value n denotes the number of assigned keypoints.



Figure 8: **Instance-level variation in dominant hyperedge assignments.** Four instances of `swivelchair`, an unseen MP-100 category, are shown. We visualize the dominant hyperedge assignment, $\arg\max_m H_{im}$, for the same keypoint set across the instances. Colored markers indicate keypoints whose dominant assignment varies across the instances, while gray markers denote assignments that remain unchanged.

their category. These results suggest that the learned assignments are largely stable within a category while allowing selective instance-level variation.

C Parameter Complexity

As shown in Tab. 13, CapeX and HyperCape share the same frozen gte-base-en-v1.5 text encoder, which accounts for most of the total parameter count. CapeX contains 37.36M trainable parameters, whereas the main HyperCape configuration with $M = 8$ contains 37.91M, corresponding to only a 1.47% increase. The hyperedge assignment module itself contains only 4,096 parameters. Increasing the number of hyperedges from $M = 4$ to $M = 32$ changes the trainable parameter count by less than 0.04% relative to the main configuration, while varying the top-k value does not affect the parameter count.

Table 13: **Parameter complexity of HyperCape.** The main configuration ($M = 8$) increases the number of trainable parameters by only 1.47% over CapeX.

Model	Trainable Params.	vs. CapeX	Total Params.
CapeX	37,361,048	–	174,137,240
HyperCape ($M = 4$)	37,906,651	$+1.46\%$	174,682,843
HyperCape ($M = 6$)	37,907,675	$+1.46\%$	174,683,867
HyperCape ($M = 8$)	**37,908,699**	$\mathbf{+1.47\%}$	**174,684,891**
HyperCape ($M = 16$)	37,912,795	$+1.48\%$	174,688,987
HyperCape ($M = 32$)	37,920,987	$+1.50\%$	174,697,179

D Corrected Textual Keypoint Descriptions

Text-driven CAPE uses textual descriptions to specify target keypoints. We identified inconsistencies in the descriptions released with CapeX (Rusanovsky et al., 2025) and corrected them at the category level prior to all training and evaluation. The corrections only resolve duplicated descriptions and inconsistent naming conventions, and the same corrected textual descriptions are used for HyperCape, CapeX, and CapeNext.

Duplicate Descriptions. The released descriptions are not always one-to-one with keypoints. In total, 157 of 1565 keypoints (10.0%), spanning 12 of the 100 categories, share their description with at least one other keypoint of the same category. For example, multiple keypoints on a vest dress may share the description *upper right body of the dress*, and several points along a locust leg may all be described as *front left leg*. Such duplicates make distinct keypoints indistinguishable from their textual input alone. We therefore assign distinct descriptions to the affected keypoints using simple semantic terms that reflect their relative anatomical or structural roles. Representative examples of these corrections are provided below.

Table 14: **Representative examples of corrected keypoint descriptions.**

Category	Original	Corrected
Vest dress	upper right body of the dress	right armhole / upper right side below the armhole
Locust	front left leg	base / upper joint / lower joint / tip
Face	middle side of the right eyebrow	outer middle / middle / inner middle of the right eyebrow

Inconsistent Left/Right Convention. We also found that the released descriptions use different left/right conventions across categories. Animal faces, clothes, and human-face categories follow a viewer-centric convention, whereas animal bodies, birds, vehicles, and the human body follow an egocentric convention. We therefore unify the affected category-level descriptions to the egocentric convention while leaving all keypoint coordinates unchanged. This correction only changes the textual naming convention. The correction affects 45 of the 100 categories and 508 keypoints containing left/right terminology.

E ADDITIONAL QUALITATIVE RESULTS

Fig. 9 provides additional qualitative comparisons with CapeX and CapeNext, the two prior CAPE methods that also use textual keypoint descriptions. Across these categories, HyperCape produces predictions that more closely match the ground-truth keypoint configuration. These examples complement the qualitative results in the main paper and further illustrate the effectiveness of HyperCape across diverse object types.



Figure 9: Additional qualitative results.