

Article

Edge-Efficient Compositional Recognition via Disentangled Prompt Tuning of Frozen Vision-Language Models

Xiaocheng Lu ¹, Chuan He ², Ziming Liu ², Fushuo Huo ², Sikai Bai ¹, Tao Han ¹
 and Jie Zhang ^{1,*}

¹ Department of Computer Science and Engineering, The Hong Kong University of Science and Technology, Hong Kong SAR, China

² Department of Computing, The Hong Kong Polytechnic University, Hong Kong SAR, China

* Correspondence: csejzhang@ust.hk

How To Cite: Lu, X.; He, C.; Liu, Z.; et al. Edge-Efficient Compositional Recognition via Disentangled Prompt Tuning of Frozen Vision-Language Models. *Edge Intelligence and Systems* **2026**, *1*(1), 3.

Received: 11 May 2026

Revised: 29 May 2026

Accepted: 18 June 2026

Published: 17 July 2026

Abstract: Deploying vision-language models (VLMs) for fine-grained compositional recognition on edge devices presents a fundamental tension: full fine-tuning is prohibitive for resource-constrained nodes, while frozen backbones with generic prompts fail on the long-tailed state-object compositions encountered in real-world edge scenarios. Soft prompt tuning offers a deployable middle ground because it leaves the backbone frozen and updates only a compact set of prompt parameters per task. However, existing prompt tuning methods for compositional zero-shot learning (CZSL) jointly optimize state and object embeddings, causing prompts to drift toward locally suboptimal regions under uneven entanglement distributions. We propose Progressively Disentangled and Recurrent Prompt Tuning (PDRPT), an edge-efficient framework that decouples object and state updates before joint refinement, suppresses traction force from highly-entangled prompts, and preserves alignment with the natural language space of CLIP. Experiments show consistent improvements over prior state-of-the-art methods while maintaining compact task adapters, providing a deployable path for VLM-based compositional perception on consumer-grade edge hardware. In this revision, we explicitly separate the experimental settings: accuracy results are obtained with CLIP ViT-L/14, while measured edge profiling is conducted with a deployable CLIP ViT-B/32 configuration on an Apple M1 (Apple Inc., Cupertino, CA, USA); ViT-L/14 edge numbers are reported as analytical FLOPs-scaled estimates rather than direct on-device measurements.

Keywords: edge intelligence; vision-language models; prompt tuning; compositional zero-shot learning; parameter-efficient adaptation

1. Introduction

Edge intelligence—deploying machine learning models on resource-constrained nodes such as smart cameras, agricultural drones, household robots, and wearable inspection devices [1,2]—increasingly demands the ability to recognize compositional visual concepts. A quality-control camera at a factory needs to flag a “cracked sealing ring” it has never been trained on; a household assistant must recognize a “wet tile floor” as a slip hazard even though training data only contained “dry tile floor”; an orchard drone needs to identify “unripe diseased apples” from sparse labelled examples. Such state-object compositions follow a long-tailed distribution that no fixed training set can enumerate, and the generalization gap is precisely what compositional zero-shot learning (CZSL) aims to close: recognizing unseen state-object pairs (e.g., ripe apple) from a vocabulary of seen primitive concepts.



Copyright: © 2026 by the authors. This is an open access article under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

Publisher’s Note: Scilight stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Vision-language models (VLMs) such as CLIP [3] have emerged as the strongest backbone for CZSL because they natively encode the language space in which compositions can be described. Yet adapting a 400 M-parameter VLM to a downstream CZSL task on the edge introduces a fundamental tension. Full fine-tuning of the backbone is intractable on edge nodes that may have only a few GB of memory and a single low-power neural accelerator; meanwhile, freezing the backbone and using generic hand-crafted prompts (“a photo of [state] [object]”) leaves substantial accuracy on the table, particularly under the long-tailed compositional distribution typical of edge scenarios. Soft prompt tuning—where only a small set of continuous prompt embeddings is learnt while the backbone weights are frozen—offers a deployable middle ground: the backbone can be quantized once and cached on-device, and only $|S \cup O| \times d$ prompt parameters need to be transferred or updated to adapt the system to a new task domain. This makes prompt tuning a natural fit for the storage, bandwidth, and energy constraints of edge AI deployment.

The prompt tuning approach is appealing in principle, but its practical realization on long-tailed CZSL distributions is undermined by a phenomenon we call prompt entanglement. State and object embeddings are not statistically independent: “ripe” co-occurs with fruits but never with shoes; “wet” is plausible for floors and clothes but ill-defined for solid metal objects. When state and object prompts are jointly optimized—as in CSP [4] and DFSP [5]—updating a heavily-co-occurring object prompt drags its associated state prompts in the same direction, biasing the latter away from primitives that they should also serve. This traction force is unevenly distributed across primitives because the seen-composition graph itself is long-tailed (Figure 1), and it traps the joint optimization in suboptimal regions (Figure 2b). Existing CZSL methods that rely on dual classifiers [6,7], graph reasoning [8], or external knowledge constraints [9] either operate in the visual space alone or treat the optimization process as a black box, leaving the entanglement problem untouched.

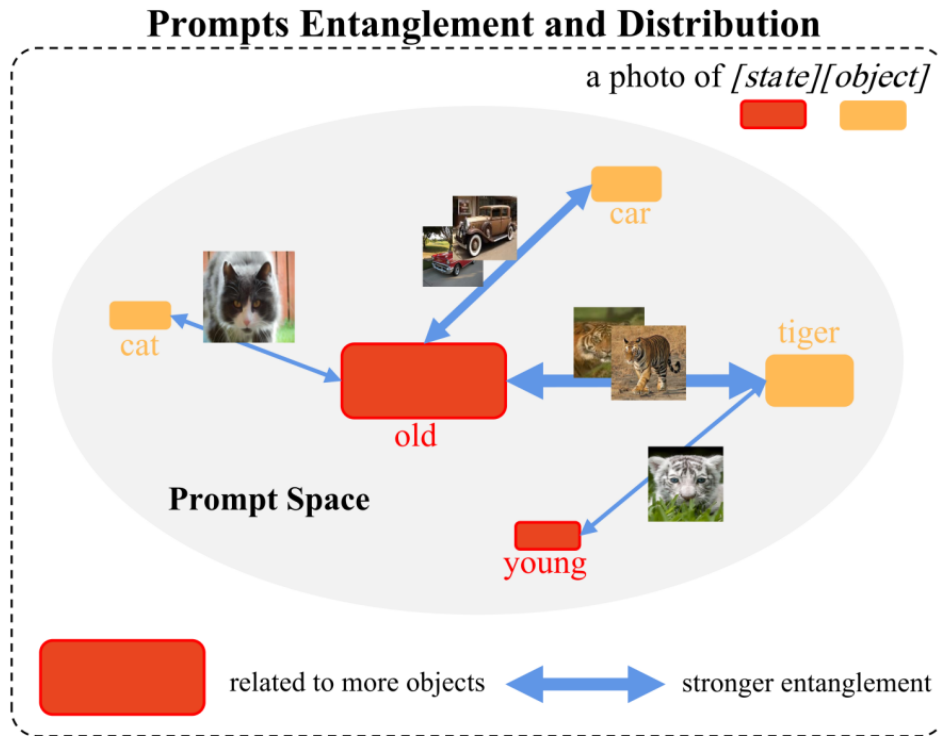


Figure 1. Prompt entanglement under long-tailed compositional distributions. Related state and object tokens cause entanglement (blue line) during optimization, and the entanglements are uneven—a state prompt with many associated object prompts (or vice versa) is shown larger. Thicker lines indicate stronger entanglement, exerting traction force that biases other prompts during gradient updates and ultimately limits the parameter-efficient adaptation that edge deployment requires.

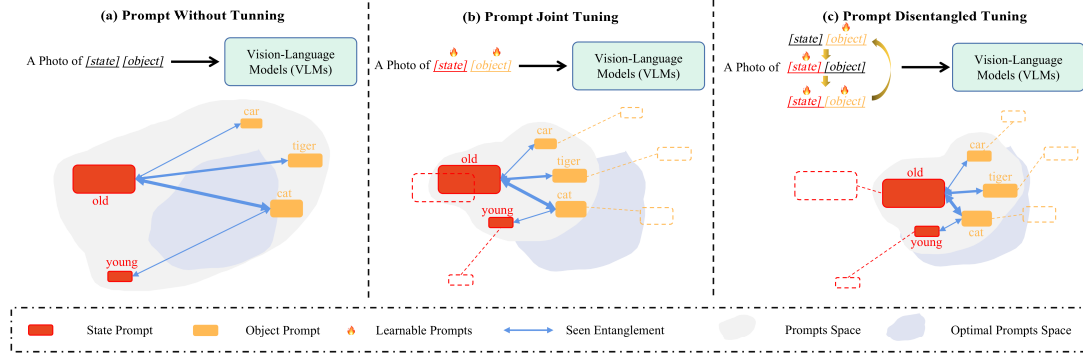


Figure 2. Optimization trajectories of three prompt tuning strategies: **(a)** the initial prompt-space configuration; Joint tuning **(b)** is pulled toward the high-entanglement region by traction force, settling at a local optimum. The proposed PDRPT tuning **(c)** decouples updates so that strongly-entangled prompts can move freely, leading the prompt space to a more globally favorable solution. Red boxes denote state prompts, orange boxes denote object prompts, and the corresponding red and orange dashed boxes indicate their prompt-space regions.

We propose Progressively Disentangled and Recurrent Prompt Tuning (PDRPT), an edge-efficient prompt tuning framework that explicitly addresses entanglement to make VLM-based compositional recognition deployable on consumer-grade edge accelerators. PDRPT treats states and objects as separately learnable tokens within the prompt template “a photo of [state] [object]” and tunes them through a three-stage status machine: object only, state only, then state-and-object jointly. Each stage runs for K epochs, and the cycle is repeated periodically (hence recurrent); during each single-modal stage the gradient flow to the frozen modality is severed, which lets the active prompts move further from their entangled initialization before the joint stage refines them. To compensate for the long-tailed entanglement distribution, we introduce an entanglement-aware reweighting term that suppresses gradient flow to highly-connected prompts and amplifies updates to weakly-connected ones, and a text-prompt regularizer that pulls learnt embeddings toward the natural-language space of Contrastive Language–Image Pre-training (CLIP) so that the prompts retain compositional generalization rather than overfitting to seen pairs.

The contributions of this work are threefold:

- We frame compositional zero-shot recognition as an edge-efficient adaptation problem for frozen vision-language models and analyze prompt entanglement as the central obstacle to parameter-efficient prompt tuning under long-tailed compositional distributions.
- We propose PDRPT, a progressively disentangled and recurrent tuning algorithm with entanglement-aware reweighting and text-prompt regularization. PDRPT updates only a small fraction of CLIP-L/14 parameters, leaves the backbone frozen and quantization-friendly, and produces compact task adapters ranging from tens of kilobytes to 1.94 MB at 16-bit floating point (FP16) across our four benchmarks, which is suitable for low-bandwidth edge deployment.
- Beyond accuracy gains of 0.6–5.4% AUC over the prior state of the art on four standard CZSL benchmarks (UT-Zappos, MIT-States, AO-Clevr, C-GQA), we evaluate end-to-end inference latency, peak memory, FLOPs, and adaptation bandwidth on Apple M1 using the MPS GPU backend at FP16 precision and interpret these measurements against published edge-AI benchmark trends to support cross-platform applicability.

To avoid over-claiming edge deployment, we now describe Apple M1 as a consumer edge proxy and distinguish it from deeply embedded accelerators such as Jetson-class modules, Raspberry Pi-class boards, Edge TPU devices, and microcontroller NPUs. Quantization is discussed as a compatibility advantage of frozen-backbone prompt tuning, not as an experimentally validated 4-bit/8-bit integer (INT4/INT8) result in this manuscript.

Related work can be grouped into three families.

CZSL methods fall into three families: (i) dual-classifier visual approaches that recognize state and object independently [6–11] and often integrate graphs or external knowledge to capture inter-primitive structure; (ii) joint-embedding methods that treat each composition as an entity and learn a common embedding space [7,12–19]; and (iii) recent VLM-based methods such as CSP [4] and DFSP [5] that fine-tune soft prompts on a frozen CLIP backbone. The latter family is the closest predecessor of our work; PDRPT differs in explicitly disentangling the state-object optimization to overcome traction force, which prior VLM-based methods do not address. Prompt learning originated in NLP [20–26], was transferred to vision tasks by CoOp [25] via continuous context tokens, and has since become a canonical parameter-efficient adaptation technique for foundation models. Compared with

adjacent parameter-efficient strategies such as LoRA [27], adapters [28], and prefix tuning [29], soft prompts uniquely keep all model weights frozen, which is critical for deployment scenarios where the backbone has been pre-quantized and shipped to edge nodes and where re-flashing weight files is prohibitively expensive. Edge AI deployment of VLMs has been studied from several angles [1,2,30]: model compression and distillation toward smaller CLIP variants such as MobileCLIP [31] and TinyCLIP [32], post-training quantization, and federated prompt learning across edge clients [33]. PDRPT is complementary to all of these directions in that it leaves the backbone fully frozen and contributes a more accurate prompt adapter that can sit on top of any quantized CLIP variant.

2. Materials and Methods

PDRPT is built around a deployment scenario in which a frozen, pre-trained vision-language backbone is provisioned once on the edge node and only a compact set of prompt parameters is tuned per task. We first formalize this edge deployment setting and the assumptions it makes about computation, storage, and bandwidth (Section 2.1). We then introduce the prompt formulation on top of CLIP (Section 2.3) and describe the progressive disentangled tuning, entanglement-aware reweighting, and text regularization that constitute PDRPT (Section 2.4). The compositional structure of the algorithm is summarized in Figure 3.

2.1. Edge Deployment Setting

We assume a two-tier deployment topology that is typical for edge AI applications. A cloud or workstation tier is used to perform CLIP-style large-scale pre-training and any task-specific prompt tuning on the seen-composition training set; the resulting artifact consists of two pieces: (i) the frozen CLIP backbone weights $\{E_v, E_t\}$, optionally quantized to FP16 or INT8 for on-device storage, and (ii) the per-task prompt embeddings $\{\theta_s, \theta_o\}$ that PDRPT optimizes. An *edge tier*—a smart camera, drone, robot, or wearable device with limited memory and a low-power neural accelerator—ingests query images, performs forward passes through the cached backbone and the locally-stored prompts, and returns the top-1 composition. The compute and storage budgets at the edge tier impose three operational requirements that motivate our algorithmic choices: (a) the backbone must be *frozen* so that it can be loaded once and reused across tasks, allowing aggressive quantization without retraining; (b) the per-task adapter must be *parameter-light* so that updating to a new task requires only a small download; and (c) the inference pipeline must be *batch-1 friendly*, since most edge workloads process a single image at a time rather than large batches. We design PDRPT to satisfy (a) by construction, (b) by tuning only the state and object embedding tables, and (c) by separating the text-encoder forward pass (which can be pre-computed once per task and cached) from the image-encoder forward pass (which is the only per-frame work).

Concretely, the RTX 3090 workstation is used for prompt optimization and accuracy evaluation, whereas the Apple M1 device is used only for inference profiling of the frozen-backbone deployment pipeline. The two measurements answer different questions: whether PDRPT improves CZSL recognition accuracy, and whether the resulting frozen-backbone-plus-prompt inference path remains feasible on a consumer edge-class device.

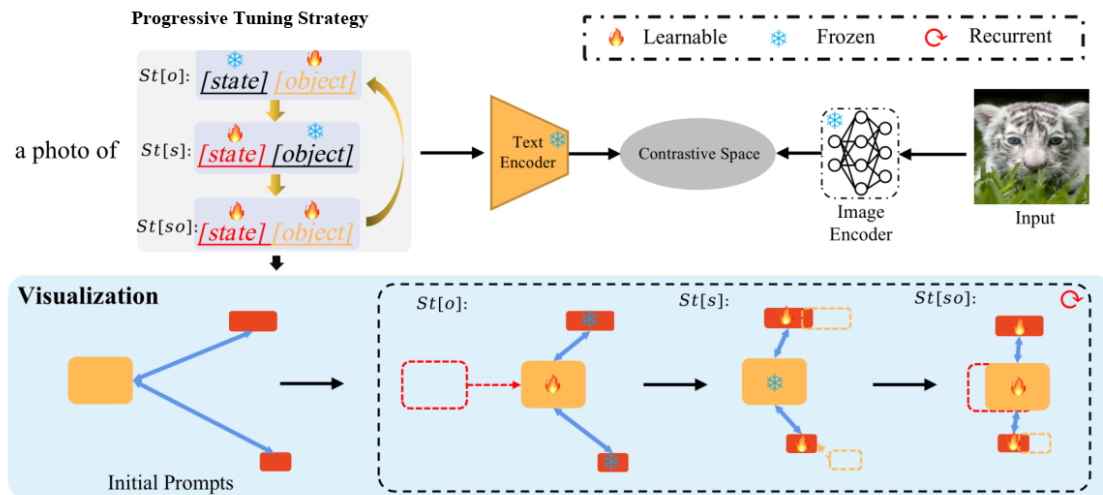


Figure 3. Overall framework of PDRPT. Compositional prompts are converted into learnable state and object tokens and tuned with a progressive disentangled and recurrent strategy. PDRPT first updates the object vocabulary,

then the state vocabulary, and finally tunes both jointly. This schedule suppresses entanglement-driven traction force and allows prompts with different entanglement degrees to be updated more effectively. Red elements denote state-related prompts and updates, orange elements denote object-related prompts and updates, and the red and orange dashed boxes delimit the corresponding processing stages.

2.2. Problem Formulation

The goal of CZSL is to recognize compositions formed from two primitive concept sets: an object set $O = \{o_1, o_2, \dots, o_m\}$ and a state/attribute set $S = \{s_1, s_2, \dots, s_n\}$. The full composition space is $C = S \times O$ with size $n \times m$. We define two disjoint subsets C_{se} and C_{un} , where both are subsets of C and $C_{se} \cap C_{un} = \emptyset$. C_{se} and C_{un} represent the seen and unseen composition sets, respectively. C_{se} is used for training, while $C_{se} \cup C_{un}$ is used for testing under the generalized zero-shot setting.

Assuming a training set $\mathcal{T} = \{(x_i, c_i) | x \in \mathcal{X}, c \in \mathcal{C}^{se}\}$ where $\mathcal{X}$ is the input image space and c belongs to the seen composition label set, CZSL aims to train a model $\mathcal{M}: \mathcal{X} \rightarrow \mathcal{C}^t$ to predict compositions in the test samples space $\mathcal{C}^t$. If $\mathcal{C}^t \cap \mathcal{C}^{se} \equiv \emptyset$, the model only needs to predict unseen compositions. Following the setting of Generalized ZSL [34], testing samples contain seen and unseen compositions, i.e., $\mathcal{C}^{se} \cup \mathcal{C}^{un}$ in this paper.

2.3. Prompt Initialization

Since combining multiple senses helps us better understand and analyze new information, human learning is essentially multi-modal, so vision language models (VLMs) are more consistent with human cognition of the world. CLIP [3] is a pioneering work of VLMs on visual tasks that promotes the rapid development of computer vision in many areas. Notably, CLIP has been pre-trained with approximately 400 million image-text pairs sourced from the internet.

Our model still follows the architecture of CLIP, whose entire model consists of an image encoder and a text encoder. Specifically, the image encoder is either a vision transformer or a convolutional neural network, and the text encoder is composed of multiple transformer layers. The two separate encoders are frozen during the training phase, and they are denoted as E_v and E_t to represent a vision encoder and a text encoder, respectively.

In CLIP, text prompts typically take the form “a photo of [class]”, where [class] denotes the target class name. For each task, the prompt is tokenized and mapped into the prompt embedding vocabulary P . The text encoder E_t then extracts textual representations f_t from these prompt embeddings, while the image encoder E_v extracts image features f_v . This process is formulated as follows:

$$f_v = \text{Norm}(E_v(x)), \quad f_t = \text{Norm}(E_t(p)), \quad (1)$$

where x represents the input query image in $\mathcal{T}$ and p denotes the prompt in P . To limit f_t and f_v to a standard range, they need to be normalized, and $\text{Norm}(\cdot)$ is denoted as:

$$\text{Norm}(f) = \frac{f}{\|f\|_2} \quad (2)$$

We then compute the cosine similarity between the query image representation and each text representation to obtain the final prediction:

$$\text{Sim}(y = (s, o) | x; \theta) = \frac{\exp(f_v \cdot f_t(s, o))}{\sum_{(\bar{s}, \bar{o}) \in \mathcal{C}^{se}} \exp(f_v \cdot f_t(\bar{s}, \bar{o}))} \quad (3)$$

where $f_t(s, o)$ denotes the text feature of input prompt $p_{s,o}$.

Since entities in CZSL are compositions of state and object primitives, the prompt can be written as “a photo of [state] [object]” under the CLIP framework. Here, [state] [object] plays the role of [class], and the learnable vocabulary has dimension $|S \cup O|$. The two vocabularies [state] and [object] are learnable during training on seen image-text pairs. The prompts are passed through the tokenizer and embedding functions as follows:

$$p_{s,o} = \{x_1, x_2, \dots, x_{pre}, x_s, x_o\}, \quad (4)$$

where $p_{s,o}$ has prefix part $\{x_1, x_2, \dots, x_{pre}\}$ and vocabulary part $\{x_s, x_o\}$. To represent learnability in $p_{s,o}$, it could be reformulated as:

$$\xi(p_{s,o}) = \{x_1, x_2, \dots, x_{pre}, \theta_s, \theta_o\}, \quad (5)$$

where $\{\theta_s, \theta_o\}$ is the learnable parameters in all prompts. So we only need to tune $|\mathcal{S} \cup \mathcal{O}| \times d$ parameters, where d is the dimension of embeddings. As a result, f_t needs to be converted to $\text{Norm}(E_t(\xi(p_{s,o})))$. Finally, we can minimize the cross entropy loss for classification:

$$\mathcal{L} = -\frac{1}{|\mathcal{C}^{se}|} \sum_{(x,y) \in \mathcal{T}} \log \text{Sim}(y = (s, o) \mid x; \theta) \quad (6)$$

2.4. Disentangled and Recurrent Prompt Tuning

An intrinsic relationship exists between [state] and [object], causing entanglement between the two vocabularies. For a simple example, [object] may contain tiger and apple, while [state] may contain old and ripe: old is more naturally combined with tiger, whereas ripe is more naturally combined with apple. Hence, when the tiger embedding is optimized, the old embedding may also be updated indirectly, regardless of the ripe state. These entanglements act as traction forces during parameter updates. As illustrated in Figure 2, such a traction force causes associated states and objects to accommodate each other during optimization, trapping the prompts in a suboptimal region. Our disentangled prompt-tuning framework mitigates this effect by controlling which prompt groups are updated at each stage.

Progressive tuning strategy. To learn more independent feature spaces for states and objects, PDRPT restructures tuning into three phases. When state parameters are frozen, object embeddings can move toward their own optima with less interference from associated state prompts, and vice versa. Specifically, the two learnable vocabularies [state] and [object] can be selectively updated or frozen. The tuning process is therefore represented by three statuses: object-only updating, state-only updating, and joint state-object updating. We denote these statuses by St as follows:

$$St = \begin{cases} o: [\text{state}] [\text{object}], \\ s: [\underline{\text{state}}] [\text{object}], \\ so: [\underline{\text{state}}][\underline{\text{object}}]. \end{cases} \quad (7)$$

where the three statuses could be abbreviated as $St[o]$, $St[s]$ and $St[so]$. These stages constitute a status machine, which controls the updating of the status of model parameters.

We next describe the status-machine operation. Given an initial state T_{St} and a round length K , the model runs K epochs in the current state and then transitions to the next state. The transition should follow a fixed order rather than being randomly selected, so that the model can progressively learn object, state, and joint compositional information. Since human perception often identifies the object before judging its state, we use “ $St[o] \rightarrow St[s] \rightarrow St[so]$ ” as the default transition sequence. In PDRPT, each status runs for K epochs before switching to the next one. This process disentangles state and object embeddings by freezing one group while updating the other, allowing each group to learn its feature space more independently. One full execution of the three statuses is defined as a round, and the rounds are repeated periodically until training ends.

Prompt Regularization. Prompt learning without regularization can overfit to a specific downstream task. The learned prompts may drift away from the original natural language prompt space and lose generalizability. To limit this drift, we input the original textual prompts “a photo of [attribute] [object]” to CLIP’s text encoder and use the resulting fixed textual embeddings as guidance. During prompt learning, we pull the optimized textual embeddings toward the fixed textual embeddings. In $\mathcal{L}_{\text{text}}$, we minimize the average distance between each learned textual embedding and its fixed counterpart. We use cosine similarity distance or L1 distance as the metric.

$$\mathcal{L}_{\text{text}} = -\frac{1}{|\mathcal{C}^{se}|} \sum_{(\bar{s}, \bar{o}) \in \mathcal{C}^{se}} \text{Sim}(f_t(St), f_t(\bar{s}, \bar{o})) \quad (8)$$

Explanation. PDRPT avoids local optima caused by entanglement by preventing all prompt groups from being dragged simultaneously. In the first step $St[o]$, only object tokens are updated while state tokens are fixed, allowing object prompts to move without immediately pulling state prompts. In $St[s]$, state prompts are then updated under fixed object prompts. Finally, both prompt groups are jointly refined in $St[so]$. This staged optimization reduces the traction force from highly entangled prompts and gives weakly entangled prompts more opportunity to reach a useful parameter region.

Why update object prompts before state prompts? First, object words are usually better grounded in CLIP’s pre-training distribution than state words, so object prompts provide a more stable anchor for compositional prompt learning. Second, recognizing the state of an entity often depends on first identifying the object. The experiment in the ablation study in Section 3.3 confirms that using $St[o]$ as the initial status leads to better prompt-space optimization.

Entanglement Rebalance. States and objects in different datasets have different degrees of entanglement. The more long-tailed this entanglement distribution is, the more likely joint optimization is to fall into a local optimum under strong traction force. Prompts associated with many other prompts exert stronger traction and also experience greater resistance during parameter updates. We therefore design rebalancing weights to suppress updates of highly entangled prompts and promote updates of weakly entangled prompts. We introduce two CZSL quantities, Ent^s and Ent^o , denoted as follows:

$$\begin{aligned} Ent^s &= \{E_{s_1}, E_{s_2}, \dots, E_{s_n} \mid s \in \mathcal{S}\}, \\ Ent^o &= \{E_{o_1}, E_{o_2}, \dots, E_{o_m} \mid o \in \mathcal{O}\}. \end{aligned} \quad (9)$$

where E_{s_1} represents that s_1 has E_{s_1} related object prompts and so on. For each prompt, its weight is expressed in the following formula:

$$w_i^s = \frac{(1/E_{s_i})^\lambda}{\sum_{j=1}^{|\mathcal{S}|} (1/E_{s_j})^\lambda}, \quad w_i^o = \frac{(1/E_{o_i})^\lambda}{\sum_{j=1}^{|\mathcal{O}|} (1/E_{o_j})^\lambda} \quad (10)$$

where λ is the scale parameter for weight distribution. As the loss calculation is for the composed entity, the learning of the prompts in the three training stages is different because the weight should be calculated separately. The formula is as follows:

$$w = C + \alpha \times \begin{cases} \text{Norm}(\mathbf{1}(w^s) \times w^o), & \text{if } St = St[o], \\ \text{Norm}(w^s \times \mathbf{1}(w^o)), & \text{if } St = St[s], \\ \text{Norm}(w^s \times w^o), & \text{if } St = St[so]. \end{cases} \quad (11)$$

where $\mathbf{1}(\cdot)$ denotes that the set all the values of the input weight matrix to 1, C is a constant value that ensures that the weight value is greater than C and α is a scale parameter for the weight of entanglement rebalancing.

To remove ambiguity, $\mathbf{1}(\cdot)$ in Equation (11) is an all-ones broadcast operator, not an indicator function. In the object-only stage $St[o]$, the state dimension is broadcast with ones and the composition weight is computed from the object vector only; in the state-only stage $St[s]$, the object dimension is broadcast with ones and the composition weight is computed from the state vector only; in $St[so]$, both primitive weights are used. The multiplication between broadcast vectors is an element-wise Hadamard product. The constant C is a positive floor preventing zero-valued sample weights, while alpha rescales the normalized entanglement weights; both are selected on validation data together with the round length K .

Finally, the composition classification cross-entropy loss could be formulated as follows:

$$\mathcal{L} = -\frac{w}{|\mathcal{C}^{se}|} \sum_{(x,y) \in \mathcal{T}} \log \text{Sim}(y = (s, o) \mid x; \theta) \quad (12)$$

For inference, there are many unseen compositions in CZSL, so we recompose the fine-tuned state and object vocabularies into test prompts. After learning the state and object prompt parameters during training, the test prompts are rearrangements of these learned primitives. The most likely state-object composition is computed as:

$$\hat{y} = \arg \max_{y \in \mathcal{C}^{se} \cup \mathcal{C}^{un}} \text{Sim}(y = (s, o) \mid x; \theta). \quad (13)$$

3. Results

We organize the empirical evaluation in two parts. We first report accuracy on the four standard CZSL benchmarks, comparing PDRPT against representative state-of-the-art methods (Sections 3.1–3.3). We then report edge deployment metrics measured on Apple M1 hardware together with analytical FLOPs and bandwidth analyses (Section 3.4), demonstrating that the accuracy gains do not come at the cost of edge feasibility.

3.1. Experimental Setup

Datasets. Our experiments are conducted on four challenging benchmark datasets: C-GQA [35], MIT-States [36], UT-Zappos [37], and AO-Clevr [12]. Specifically, AO-Clevr contains 180 K synthetic images with 8 states and 3 objects. The UT-Zappos shoes dataset consists of 29,126 images composed of 16 states and 12 objects. C-GQA, the largest paired dataset for CZSL in our evaluation, contains 453 states and 870 objects, totaling 39,298 images. As for compositions, AO-Clevr includes 16 seen compositions, 4 unseen compositions for validation, and 4 unseen compositions for testing, following the split settings in [12]. UT-Zappos consists of 83 seen and 15/18 (validation/test) unseen compositions. MIT-States contains 1262 seen compositions and 300 unseen compositions for

validation and 400 unseen compositions for testing. For C-GQA, the split contains 5592 compositions for training, 1040 compositions for validation, and 923 compositions for testing. More details are reported in Appendix A.

Metrics. In our experiments, we calculate prediction accuracy over compositions. Following previous work [38], we compare four metrics with state-of-the-art methods: Seen, Unseen, HM, and AUC. Seen (S) and Unseen (U) denote the accuracy on seen and unseen compositions, respectively. HM is the harmonic mean of S and U. Since zero-shot models have an inherent bias toward seen compositions, we draw a seen-unseen accuracy curve at different calibration values from $-\infty$ to $+\infty$ and compute the Area Under the Curve (AUC).

Implementation Details. We implement PDRPT with PyTorch 1.12.1 [39] and optimize it with the Adam optimizer over the aforementioned four benchmark datasets. The image encoder and text encoder are both based on the pre-trained CLIP ViT-L/14 model, and the entire model is trained and evaluated on one NVIDIA RTX 3090 GPU (NVIDIA Corporation, Santa Clara, CA, USA). More parameter details are reported in Appendix B.

All accuracy tables in Sections 3.2 and 3.3 are obtained with CLIP ViT-L/14 on the RTX 3090. The Apple M1 experiments in Section 3.4 are a separate inference-only profiling study: they measure CLIP ViT-B/32 directly under the MPS FP16 backend and extrapolate ViT-L/14 latency from image-encoder FLOPs. Thus, the manuscript does not mix training hardware with deployment hardware.

3.2. Main Results

To verify the effectiveness of PDRPT, we compare our method with extensive prior compositional zero-shot learning algorithms, including vision-based methods AoP [40], LE+ [35], TMN [13], SymNet [6], CompCos [38], IVR [41], and SCEN [10], as well as vision-language models CLIP [3], CoOp [25], and CSP [4]. We analyze the four standard metrics Seen, Unseen, HM, and AUC. The experimental results are shown in Table 1. The best results are shown in bold and the second-best results are underlined.

Table 1. Results of PDRPT on UT-Zappos, MIT-States, AO-Clevr and C-GQA. Seen and Unseen are the predicted accuracy evaluated on seen and unseen compositions. HM is the harmonic mean of Unseen and Seen and AUC is the area under the curve. The best results are in bold and the second results are underlined.

Method	UT-Zappos				MIT-States				AO-Clevr				C-GQA			
	Seen	Unseen	HM	AUC	Seen	Unseen	HM	AUC	Seen	Unseen	HM	AUC	Seen	Unseen	HM	AUC
AoP	59.8	54.2	40.8	25.9	14.3	17.4	9.9	1.6	95.5	85.5	64.8	65.8	17.0	5.6	5.9	0.7
LE+	53.0	61.9	41.0	25.7	15.0	20.1	10.7	2.0	95.7	99.2	92.3	93.5	18.1	5.6	6.1	0.8
TMN	58.7	60.0	45.0	29.3	20.2	20.1	13.0	2.9	96.1	93.9	45.0	29.3	23.1	6.5	7.5	1.1
SymNet	49.8	57.4	40.4	23.4	24.2	25.2	16.1	3.0	87.1	97.8	71.8	74.2	<u>30.9</u>	13.3	13.5	3.1
CompCos	59.8	62.8	43.1	28.7	25.3	24.6	16.4	4.5	96.3	99.1	94.5	94.2	30.7	12.2	12.8	2.9
IVR	56.9	65.5	46.2	30.6	-	-	-	-	<u>97.1</u>	99.3	95.1	<u>95.6</u>	27.3	10.0	10.9	2.2
SCEN	63.5	63.1	<u>47.8</u>	32.0	29.9	25.2	18.4	5.3	-	-	-	-	28.9	25.4	17.5	5.5
CLIP	15.8	49.1	15.6	5.0	30.2	46.0	26.1	11.0	61.4	99.7	60.3	51.8	7.5	25.0	8.6	1.4
CoOp	52.1	49.3	34.6	18.8	36.7	49.2	31.6	15.1	95.8	<u>99.8</u>	94.6	95.2	20.5	26.8	17.1	4.4
CSP	<u>64.2</u>	<u>66.2</u>	46.6	<u>33.0</u>	<u>46.7</u>	49.5	<u>35.9</u>	<u>19.3</u>	96.6	99.5	<u>95.7</u>	95.3	27.4	<u>27.1</u>	<u>19.9</u>	<u>5.9</u>
PDRPT	65.1	72.29	50.35	38.26	48.07	<u>49.46</u>	36.94	19.76	99.2	99.9	99.4	99.1	32.55	29.11	22.18	7.63

In Table 1, PDRPT outperforms almost all reported baselines. Specifically, PDRPT improves AUC by +5.4%, +3.8%, and +0.6% on UT-Zappos, AO-Clevr, and C-GQA, respectively, and improves HM by +6.4% on UT-Zappos. PDRPT also improves CSP by +0.9% on UT-Zappos and +2.6% on AO-Clevr for Seen accuracy. The only decreases are small drops on MIT-States Unseen and C-GQA Seen, while Unseen accuracy improves by +0.9% on UT-Zappos and +1.6% on C-GQA. Overall, PDRPT achieves stronger seen-unseen trade-offs than prior methods, supporting the effectiveness of reducing entanglement-driven traction force during prompt optimization.

The C-GQA result also exposes a scaling limitation: when the primitive vocabulary becomes very large, the fixed three-stage status machine has less room to disentangle overlapping dependencies, and the gain shifts toward unseen generalization rather than uniformly improving seen accuracy. We discuss this limitation further in Section 4.

Qualitative results. We visualize top-1 image-to-text retrieval predictions for seen and unseen compositions in Figure 4. The samples are randomly selected from the benchmark datasets. Correct primitive predictions are highlighted in green, and incorrect predictions are highlighted in blue. In successful cases, the top-1 prediction usually recovers both state and object correctly. In failure cases, challenging compositions such as “parked SUV” are more likely to confuse the state prediction. Overall, objects are more predictable than states, which is consistent with the compositional recognition setting.



Figure 4. Qualitative results. We evaluate top-1 predictions for representative cases from these datasets. The six columns on the left show successful predictions, and the remaining columns show failure cases. Green denotes correct predictions and blue denotes incorrect predictions. All images are randomly selected, and additional results are provided in Appendix C.

In addition, we report the unseen-seen accuracy curves on UT-Zappos and C-GQA in Figure 5. As the calibration value increases, unseen accuracy rises while seen accuracy declines. AUC summarizes the balance between seen and unseen accuracy across operating points. Compared with representative algorithms, PDRPT maintains a better balance between seen and unseen pairs on both datasets. On UT-Zappos, the PDRPT curve lies above the competing methods over most operating points; on C-GQA, it also outperforms the previous state-of-the-art CSP. These results demonstrate the robustness of the proposed method.

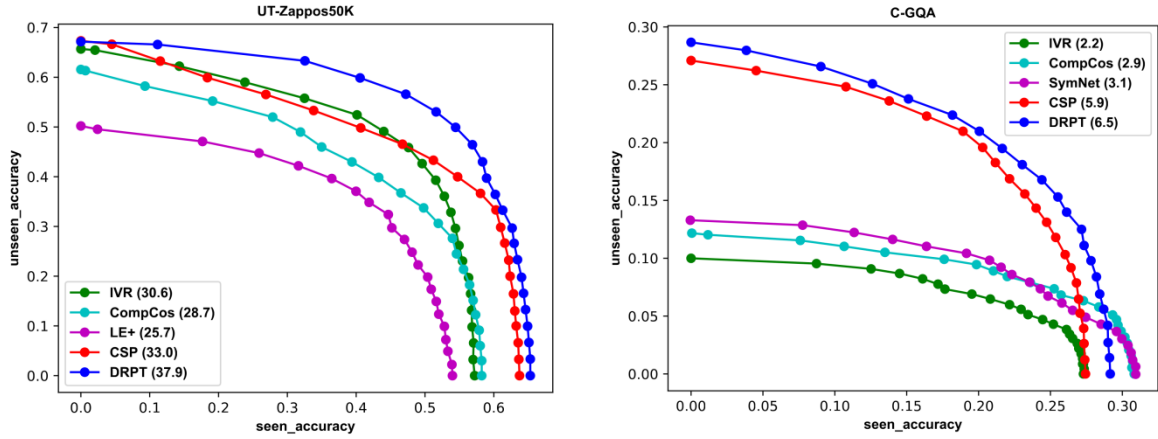


Figure 5. Unseen-seen accuracy curve on two challenging datasets, UT-Zappos and C-GQA, where PDRPT is in blue. We compare PDRPT with other representative algorithms, and AUC surpasses these models by large margins, verifying the superiority of our method.

Hyperparameter sensitivity analysis. We study the sensitivity of PDRPT to the round length K and the rebalancing coefficient w on UT-Zappos. Figure 6 shows that the best round length is $K = 3$ and the best AUC is obtained when $w = 1.5$. When K is smaller than 3, each state is updated for too few epochs, reducing the effectiveness of progressive tuning. A very small w is also ineffective because the entanglement-rebalancing term becomes too weak.

3.3. Ablation Study

To empirically show the effectiveness of PDRPT, we conduct extensive experiments and report the results for the ablation study.

Ablation on different components of PDRPT. We first evaluate the effectiveness of the entanglement reweighting term w and the progressive disentangled recurrent tuning strategy (D&R-PT) on UT-Zappos and C-GQA. The results are shown in Table 2. Both D&R-PT and w provide positive gains: D&R-PT improves AUC by +2.4% on UT-Zappos, while w improves AUC by +0.7%; on C-GQA, the two components improve Seen accuracy by +0.9% and +0.7%, respectively. Combining D&R-PT and w gives the best overall result, verifying that both components contribute to PDRPT.

Table 2. Ablation study for different components in PDRPT on UT-Zappos and C-GQA. The best results are in bold. Underlined values indicate the second-best results.

D&R-PT	w	UT-Zappos				C-GQA			
		S	U	HM	AUC	S	U	HM	AUC
No	No	64.2	66.2	46.4	33.0	27.4	<u>27.1</u>	19.9	5.9
No	Yes	64.6	65.2	48.3	33.7	28.1	26.8	<u>20.0</u>	<u>6.0</u>
Yes	No	<u>64.7</u>	67.4	<u>48.9</u>	<u>35.4</u>	<u>28.3</u>	26.9	19.7	5.9
Yes	Yes	65.1	67.1	53.0	38.4	29.2	28.7	20.5	6.5

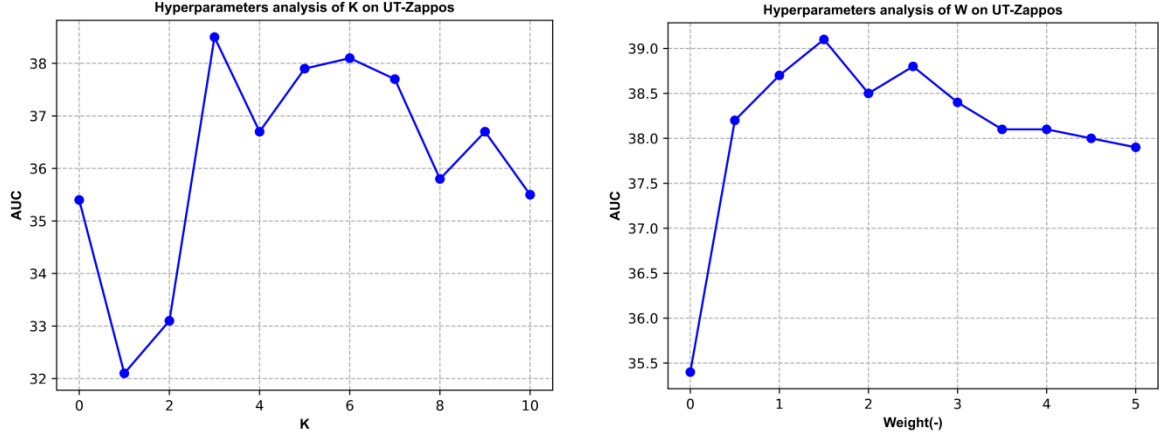


Figure 6. Unseen-seen accuracy curve on UT-Zappos [37] for hyperparameters analysis of K and w .

Ablation on different status transition sequences. St has three statuses in total, and there are six possible transition sequences in one round of status transition, as shown in Table 3. In these cases, “a $\rightarrow$ o $\rightarrow$ ao” follows the notation used in the original implementation, where a denotes the state/attribute status s and ao denotes the joint state-object status so ; equivalently, this sequence corresponds to $St[s] \rightarrow St[o] \rightarrow St[so]$. As shown in Table 3, “o $\rightarrow$ a $\rightarrow$ ao” obtains the best results among these sequences, and “o $\rightarrow$ ao $\rightarrow$ a” obtains the second-best results. When the joint status $St[so]$ appears too early, the overall effect is worse because state and object prompts have not yet been learned independently. Overall, at least one progressive transition sequence improves over joint tuning. Random Sequence denotes random status transitions and is calculated with 10 random seeds. Dynamic sequence transformation can be further studied in future work, including dynamic K values and automatic status switching.

Table 3. Ablation study for different training status transition sequences in PDRPT on UT-Zappos and C-GQA. The best results are in bold. Underlined values indicate the second-best results.

Status Sequence	UT-Zappos				C-GQA			
	S	U	HM	AUC	S	U	HM	AUC
a $\rightarrow$ o $\rightarrow$ ao	64.7	68.4	49.0	35.0	28.5	25.6	19.6	5.8
a $\rightarrow$ ao $\rightarrow$ o	63.4	70.0	46.4	33.5	26.8	<u>28.1</u>	20.0	6.1
o $\rightarrow$ a $\rightarrow$ ao	64.5	<u>69.4</u>	52.3	38.5	<u>29.2</u>	28.7	20.5	6.5
o $\rightarrow$ ao $\rightarrow$ a	<u>65.1</u>	67.3	<u>50.4</u>	<u>36.7</u>	28.4	27.7	<u>20.3</u>	<u>6.2</u>
ao $\rightarrow$ a $\rightarrow$ o	64.3	<u>69.4</u>	50.0	36.2	28.2	26.2	19.7	5.9
ao $\rightarrow$ o $\rightarrow$ a	64.2	67.9	46.5	33.7	28.3	27.5	19.8	6.0
Random Sequence	65.9	65.8	49.5	35.4	29.5	26.4	19.9	<u>6.2</u>

3.4. Edge Deployment Evaluation

Beyond accuracy, the practical value of PDRPT for edge intelligence rests on three deployment-relevant properties: (i) the inference latency of the resulting frozen-backbone-plus-prompt pipeline on representative edge hardware, (ii) the peak resident memory footprint, and (iii) the bandwidth required to transfer per-task prompt parameters from the cloud to the edge. We evaluate all three on Apple M1 (8-core CPU comprising 4 performance and 4 efficiency cores; 8-core integrated GPU with ~ 2.6 TFLOPS 32-bit floating point (FP32) throughput; 16-core Apple Neural Engine rated at 11 TOPS; 8 GB unified memory) using PyTorch 2.5 with the MPS GPU back-end at FP16 precision. Our empirical measurements therefore reflect the Apple M1 GPU/MPS execution path rather

than direct Neural Engine execution. Apple M1 is positioned as a representative consumer-grade edge AI platform whose overall accelerator budget is comparable to 2021-era flagship mobile SoCs. Empirical measurements are conducted on a CLIP ViT-B/32 backbone, the largest variant whose FP16 weights and PyTorch activations comfortably co-exist with typical OS and application memory in 8 GB unified RAM. For the larger ViT-L/14 backbone used in the accuracy evaluation, we report parameter counts and forward FLOPs analytically and extrapolate latency from the FLOPs ratio. We interpret these numbers together with public edge-AI benchmark trends such as MLPerf Mobile [42] to support the cross-platform reading of our results without claiming direct Neural Engine or Jetson measurements for every CLIP variant.

We use Apple M1 as a reproducible consumer edge-class proxy rather than as a substitute for all edge accelerators. The current measurements therefore support feasibility for laptop/desktop-class integrated edge devices, while deployment on Jetson Nano/Orin Nano, Raspberry Pi-class CPUs, Edge TPU, or microcontroller NPUs requires device-specific kernels and is left as future work.

3.4.1. Inference Latency and Peak Memory

Table 4 reports per-image end-to-end latency and peak resident memory of the PDRPT inference pipeline. We measure on a deployable ViT-B/32 backbone running on Apple M1 (MPS back-end, FP16)—the production-style configuration for consumer 8 GB devices—and report parameter counts and FLOPs analytically for the larger ViT-L/14 backbone used in the accuracy evaluation of Section 3.2. Because PDRPT freezes the backbone, the text-encoder forward pass over all candidate composition prompts can be pre-computed once per task and cached, leaving only the image-encoder forward pass on the critical path of every inference call. This decoupling is one of the largest practical advantages of PDRPT for edge deployment and is reflected in the numerical results: the per-frame work is ~13 ms while the per-task work (run once at deployment time) ranges from 65 ms to several seconds depending on the size of the candidate composition vocabulary.

Table 4. Per-image end-to-end inference latency and peak resident memory of the PDRPT inference pipeline on Apple M1 (8 GB unified memory). The ViT-B/32 numbers are measured directly on M1 MPS at FP16 precision; ViT-L/14 numbers are reported analytically (Section 3.4). PDRPT cannot be compared head-to-head against full fine-tuning at inference time because the runtime computational graph is identical to a frozen-backbone deployment; the savings of PDRPT are realized at adaptation time (Table 5) and at storage time (rightmost column).

Backbone	Source	Image enc. Latency (ms)	Peak Resident Memory (MB)	Backbone Storage FP16 (MB)	Per-Task Adapter FP16
ViT-B/32	measured (M1 MPS, FP16)	13.4 ± 0.4	~219	303	11 KB–1.29 MB
ViT-B/16	analytical (FLOPs x4)	~54	~300	299	11 KB–1.29 MB
ViT-L/14	analytical (FLOPs x9.3)	~125	~870	855	16.5 KB–1.94 MB

The ViT-L/14 latency estimates above are obtained by scaling the measured ViT-B/32 latency by the ratio of theoretical image-encoder GFLOPs ($80.7/8.7 \approx 9.3$); they are conservative because ViT-L/14 also benefits from larger-batch arithmetic intensity. We emphasize that under PDRPT the dominant per-frame cost is the frozen-CLIP image-encoder forward pass alone—the prompt-embedding lookups that constitute PDRPT’s adaptation are vector indexes whose cost is in the microsecond range and is dominated by the encoder by orders of magnitude.

We further clarify that all reported profiling numbers are FP16. Since PDRPT does not update backbone weights, it is compatible with an INT8 or INT4 backbone that has been quantized and cached once, but we do not report direct quantized-backbone accuracy or latency in this revision.

3.4.2. Parameter and FLOP Analysis

Table 5 compares the trainable parameter count and forward-pass FLOPs of PDRPT to those of full fine-tuning, LoRA-r8 adapters, and prefix tuning (depth-12) on the same CLIP-L/14 backbone. PDRPT updates only the state and object embedding tables of size $|S \cup O| \times d$ where $d = 768$ for CLIP-L/14, totalling between 0.002% (AO-Clevr, 8.4 K parameters) and 0.24% (C-GQA, 1.0 M parameters) of the backbone depending on the size of the primitive vocabulary. The forward FLOPs of PDRPT and a frozen-backbone deployment are identical because the runtime computational graph is the same; the difference between methods at *inference time* is therefore entirely determined by the frozen-backbone forward, with PDRPT adding only the negligible cost of the prompt embedding lookups.

Table 5. Trainable parameters and per-image forward FLOPs of competing parameter-efficient adaptation strategies on a CLIP-L/14 backbone (428 M total parameters, 80.7 GFLOPs per image). For PDRPT the trainable adapter size is data-dependent; we report the smallest (AO-Clevr) and largest (C-GQA) configurations from our four benchmarks. The best results are shown in bold.

Method	Trainable Params (Smallest Task)	Trainable Params (Largest Task)	% of CLIP-L/14	Backbone Status	Forward GFLOPs
Full fine-tuning	428 M	428 M	100%	Trainable	80.7
LoRA-r8 (attn)	4.7 M	4.7 M	~1.10%	Frozen wts./trainable adapt.	~80.9
Prefix tuning (depth 12)	0.59 M	0.59 M	~0.14%	Frozen	~81.2
PDRPT	8.4 K (AO-Clevr)	1016 K (C-GQA)	0.002–0.24%	Frozen	80.7

3.4.3. Adaptation Bandwidth

For each benchmark dataset, the prompt parameters that PDRPT must transmit to adapt an already-deployed edge node to that task are exactly $|S \cup O| \times d \times b$ bytes, where $b = 4$ for FP32 and $b = 2$ for FP16. The resulting per-task downloads on a ViT-L/14 backbone (which uses $d = 768$) are 16.5 KB for AO-Clevr, 42 KB for UT-Zappos, 540 KB for MIT-States, and 1.94 MB for C-GQA at FP16. Even the largest case is roughly three orders of magnitude smaller than the 855 MB FP16 backbone and remains practical for low-bandwidth edge updates. For the smaller ViT-B variants ($d = 512$) these numbers shrink by a further one third.

3.4.4. Cross-Platform Extrapolation

The relative scaling of CLIP variants across edge accelerators can be approximated from image-encoder GFLOPs ratios. The ratios in Table 4 (8.7 for ViT-B/32, 35.1 for ViT-B/16, 80.7 for ViT-L/14) provide a defensible first-order extrapolation of latency to higher-capacity backbones on the same hardware and to other edge accelerators with comparable arithmetic throughput. Public edge-AI benchmark suites such as MLPerf Mobile [42] support the broader observation that optimized vision inference on modern mobile and edge SoCs can meet interactive latency budgets. Two practical takeaways follow from these numbers: (i) ViT-B/32 + PDRPT is real-time-ready (>70 FPS) on commodity 8 GB consumer SoCs and is the recommended deployment configuration; (ii) ViT-L/14 + PDRPT, while strictly more accurate, exceeds the active-memory budget of 8 GB unified-memory consumer devices once typical OS and application memory is accounted for, and is more appropriately deployed on dedicated edge accelerators such as Jetson Orin Nano (with FP16 quantization) or on shared/cloud-edge tiers where memory pressure is less acute.

4. Discussion

The accuracy improvements reported in Table 1 are unevenly distributed across datasets, and we interpret this through the lens of entanglement variance.

UT-Zappos and AO-Clevr have moderate average entanglement rates with high entanglement variance—a small set of state primitives appears in many compositions, while most appear in few—and PDRPT’s largest gains (5.4% AUC on UT-Zappos, 3.8% on AO-Clevr) are observed precisely on these datasets where joint optimization is most distorted by traction force. C-GQA, in contrast, has very low average entanglement because most state-object pairs occur only once, and the gain there is a more modest 0.6% AUC; this is consistent with our hypothesis that disentanglement matters most when entanglement is uneven, not merely when it is large.

This vocabulary-scaling behavior suggests that a hard time-sliced schedule can become a bottleneck as the state-object graph approaches open-world density. With hundreds of states and objects, many primitives participate in sparse but semantically overlapping neighborhoods, so the fixed $St[o] \rightarrow St[s] \rightarrow St[so]$ cycle may underfit fine-grained cross-dependencies for seen compositions while still improving unseen calibration. A natural extension is to replace the fixed status machine with an adaptive scheduler that allocates update time according to per-primitive degree, gradient norm, or validation-calibration error.

From a deployment perspective, PDRPT combines a frozen, quantization-friendly CLIP backbone with a compact per-task adapter and batch-1 inference measured on the Apple M1 MPS GPU backend.

This design lets PDRPT serve as the prompt-tuning component of a wider edge VLM stack. Compared with parameter-efficient alternatives such as LoRA and adapter modules, soft prompts uniquely keep all model weights frozen, which is critical when the backbone has been pre-quantized and shipped to edge nodes and when re-flashing weight files is prohibitively expensive. PDRPT’s gains are therefore complementary to backbone compression; a

quantized MobileCLIP variant with PDRPT prompts on top would likely realize even tighter latency and storage budgets, and we leave a full study of this combination to future work.

This interpretation is narrower than a direct quantization benchmark. PDRPT is quantization-friendly in the sense that it preserves the frozen backbone file and updates only prompt embeddings; validating actual INT8/INT4 accuracy-latency trade-offs on each hardware backend remains an important future experiment.

Several limitations remain.

The accuracy results of PDRPT are obtained with the CLIP-L/14 backbone, while the empirical edge measurements in Section 3.4 are obtained with the smaller CLIP-B/32 backbone because the unified-memory budget of 8 GB Apple M1 does not comfortably accommodate the FP16 weights and PyTorch activations of CLIP-L/14 alongside typical OS and application memory; we bridge the gap with an analytical FLOPs-ratio extrapolation. A more thorough characterization that retrains PDRPT prompts on smaller CLIP variants and reports accuracy together with measured edge latency on the same backbone would strengthen the deployment story; we leave this to future work. Our edge measurements are also conducted on a unified-memory SoC, and results on heterogeneous edge platforms (separate CPU and discrete accelerator with explicit memory transfers) may differ in the latency of memory-bound primitives. Finally, we do not study the privacy properties of transmitting prompt embeddings between cloud and edge, which is an important direction for federated edge VLM deployments.

A second limitation is that the present edge profiling uses Apple M1 rather than deeply embedded accelerators. Therefore, the results should be read as evidence that the PDRPT inference graph is lightweight and deployable on consumer-integrated hardware, not as a complete benchmark suite across all edge-AI systems.

Future work should extend PDRPT in two directions.

First, the round length K and the status transition policy are currently fixed hyperparameters; an automatic schedule based on the per-prompt gradient norm trajectory could remove a degree of freedom and likely tighten convergence. Second, our edge-side adaptation pipeline currently assumes that the cloud has produced the prompt parameters; an on-device fine-tuning regime in which a small number of locally collected samples updates the prompts incrementally would directly address the streaming long-tail scenarios that motivate this work, and the small parameter footprint of PDRPT makes such on-device tuning practical even on memory-constrained devices.

5. Conclusions

We presented PDRPT, an edge-efficient prompt tuning framework for vision-language models that addresses prompt entanglement under long-tailed compositional distributions through three-stage progressive disentangled optimization, entanglement-aware reweighting, and text-prompt regularization. Across four standard CZSL benchmarks PDRPT improves AUC by 0.6–5.4% over previous state-of-the-art methods while updating only a small fraction of the CLIP-L/14 parameters and producing compact per-task adapters that are far smaller than the full backbone. Edge deployment measurements on Apple M1 hardware confirm that the resulting pipeline runs within the latency, memory, and bandwidth budgets that consumer-grade edge AI accelerators provide. Together these results establish disentangled soft prompts as a practical mechanism for compositional VLM adaptation on the edge.

The revised manuscript explicitly states that ViT-L/14 is used for accuracy evaluation, while directly measured edge profiling uses ViT-B/32 on Apple M1 and reports ViT-L/14 through analytical extrapolation.

Author Contributions

X.L.: conceptualization, methodology, software, validation, formal analysis, investigation, writing original draft preparation, writing-review and editing. C.H.: conceptualization, methodology, software, validation, writing review and editing. Z.L.: software, data curation, validation. F.H.: visualization, investigation. S.B.: data curation, investigation. T.H.: investigation, writing review and editing. J.Z.: supervision, project administration, conceptualization, methodology, writing review and editing. All authors have read and agreed to the published version of the manuscript.

Funding

This research received no external funding.

Institutional Review Board Statement

Not applicable.

Informed Consent Statement

Not applicable.

Data Availability Statement

The four benchmark datasets used in this study are publicly available: MIT-States (https://web.mit.edu/phillipi/Public/states_and_transformations/index.html, accessed on 10 July 2026), UT-Zappos (<https://vision.cs.utexas.edu/projects/finegrained/utzap50k>, accessed on 10 July 2026), C-GQA (<https://github.com/ExplainableML/czsl>, accessed on 10 July 2026), and AO-Clevr (<https://github.com/atzmonyuv/causal-czsl>, accessed on 10 July 2026). Source code, training scripts, pretrained PDRPT prompt checkpoints, Core ML conversion scripts, and the edge-benchmark harness used in Section 3.4 will be made publicly available upon acceptance.

Acknowledgments

The authors thank colleagues and collaborators who provided feedback on earlier drafts of this manuscript.

Conflicts of Interest

The authors declare no conflict of interest.

Use of AI and AI-Assisted Technologies

During the preparation of this work, the authors used OpenAI Codex to assist with language editing and proof correction. After using this tool, the authors reviewed and edited the content as needed and take full responsibility for the content of the published article.

Appendix A. Dataset Statistics

We report the composition counts and image counts of the four benchmark datasets used in this study in Table A1. Here $|\mathcal{S}|$ and $|\mathcal{O}|$ denote the number of state and object primitives, $|\mathcal{C}^{se}|$ is the number of seen compositions used for training, $|\mathcal{C}^{un}|$ is the number of unseen compositions used for evaluation, and $|\mathcal{I}|$ is the number of images in each split.

Table A1. Dataset statistics for MIT-States, UT-Zappos, C-GQA, and AO-Clevr.

Dataset	$ \mathcal{S} $	$ \mathcal{O} $	Train $ \mathcal{C}^{se} $	Train $ \mathcal{I} $	Val. $ \mathcal{C}^{se} $	Val. $ \mathcal{C}^{un} $	Val. $ \mathcal{I} $	Test $ \mathcal{C}^{se} $	Test $ \mathcal{C}^{un} $	Test $ \mathcal{I} $
MIT-States [36]	115	245	1262	30338	300	300	10420	400	400	12995
UT-Zappos [37]	16	12	83	22998	15	15	3214	18	18	2914
C-GQA [35]	453	870	6963	26920	1173	1368	7280	1022	1047	5098
AO-Clevr [12]	8	3	16	102482	4	4	38870	4	4	38648

Appendix B. Hyperparameters

We choose the hyperparameters of PDRPT for each dataset based on the best validation loss. Table A2 reports the resulting settings: learning rate, batch size, number of epochs, dropout rate, weight decay, round range K , scale parameter λ , and entanglement-rebalancing coefficient w .

Validation protocol. For each dataset, K , λ , C , and α are selected on the validation split by maximizing validation AUC after the same calibration sweep used for test evaluation. Calibration values are swept over the standard seen-unseen bias range, and test AUC is computed only once using the selected hyperparameters. Baseline numbers in Table 1 follow the reported CLIP ViT-L/14 protocol when available; for CLIP, CoOp, and CSP we use the same backbone and split protocol as PDRPT to ensure a like-for-like comparison.

Table A2. PDRPT hyperparameters for the four benchmark datasets.

Hyperparameter	UT-Zappos	MIT-States	AO-Clevr	C-GQA
Learning rate	5×10^{-4}	1×10^{-4}	5×10^{-5}	5×10^{-5}
Batch size	128	128	128	128
Number of epochs	15	45	20	45
Dropout rate	0.3	0.3	0.3	0.3
Weight decay	1×10^{-5}	1×10^{-5}	1×10^{-5}	1×10^{-5}
Round range K	3	10	3	5
Scale parameter λ	5	5	2	5
Coefficient w	5	0.5	1	0.1

Appendix C. Additional Qualitative Results

We provide top-5 image-to-text and text-to-image retrieval results in Figures A1 and A2 for samples randomly drawn from UT-Zappos and C-GQA. Successful primitive predictions are highlighted in green and incorrect ones in blue. Even in failure cases the retrieved candidates often differ only in a single primitive (e.g., correct object, mistaken state), which is consistent with the entanglement-driven analysis in the main text.

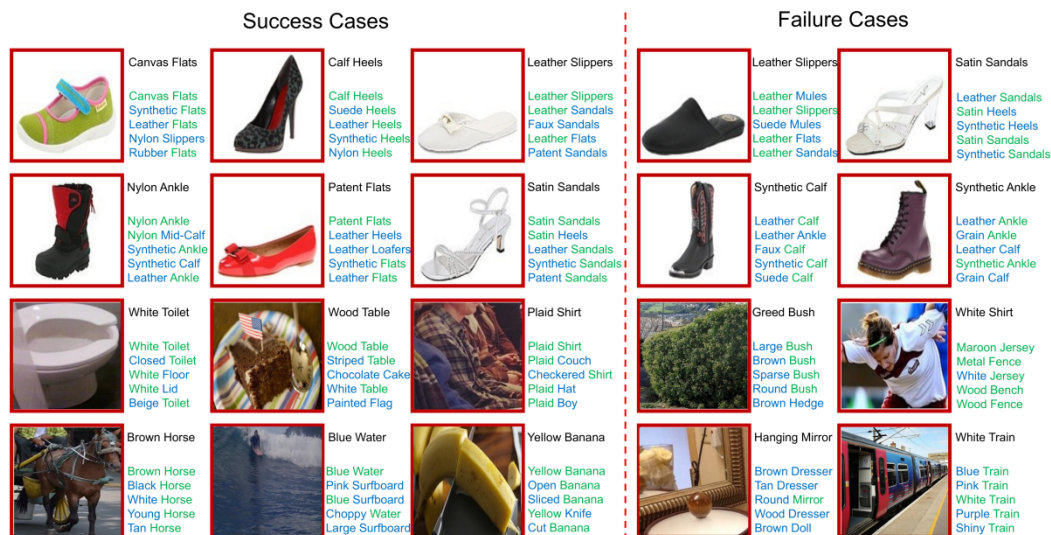


Figure A1. Qualitative results for top-5 image-to-text retrieval on UT-Zappos and C-GQA. Wrong predictions are shown in blue.



Figure A2. Qualitative results for top-5 text-to-image retrieval on UT-Zappos. Wrong predictions are shown in blue.

References

- Deng, S.; Zhao, H.; Fang, W.; et al. Edge Intelligence: The Confluence of Edge Computing and Artificial Intelligence. *IEEE Internet Things J.* **2020**, *7*, 7457–7469.
- Shi, W.; Cao, J.; Zhang, Q.; et al. Edge Computing: Vision and Challenges. *IEEE Internet Things J.* **2016**, *3*, 637–646.

3. Radford, A.; Kim, J.W.; Hallacy, C.; et al. Learning Transferable Visual Models from Natural Language Supervision. *Computer Vision and Pattern Recognition. arXiv* **2021**, arXiv:2103.00020.
4. Nayak, N.V.; Yu, P.; Bach, S.H. Learning to Compose Soft Prompts for Compositional Zero-Shot Learning. *arXiv* **2022**, arXiv:2204.03574.
5. Lu, X.; Guo, S.; Liu, Z.; et al. Decomposed Soft Prompt Guided Fusion Enhancing for Compositional Zero-Shot Learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Vancouver, Canada, 18–22 June 2023; pp. 23560–23569.
6. Li, Y.L.; Xu, Y.; Mao, X.; et al. Symmetry and Group in Attribute-Object Compositions. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, Seattle, WA, USA, 13–19 June 2020.
7. Misra, I.; Gupta, A.; Hebert, M. From Red Wine to Red Tomato: Composition with Context. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, Honolulu, HI, USA 21–26 July 2017.
8. Mancini, M.; Naeem, M.F.; Xian, Y.; et al. Learning Graph Embeddings for Open World Compositional Zero-Shot Learning. *IEEE Trans. Pattern Anal. Mach. Intell.* **2024**, *46*, 1545–1560.
9. Karthik, S.; Mancini, M.; Akata, Z. KG-SP: Knowledge Guided Simple Primitives for Open World Compositional Zero-Shot Learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, New Orleans, LA, USA 19–24 June 2022; pp. 9336–9345.
10. Li, X.; Yang, X.; Wei, K.; et al. Siamese Contrastive Embedding Network for Compositional Zero-Shot Learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, New Orleans, LA, USA 19–24 June 2022; pp. 9326–9335.
11. Yang, Y.; Pan, R.; Li, X.; et al. Dual-Stream Contrastive Learning for Compositional Zero-Shot Recognition. *IEEE Trans. Multimed.* **2023**, *26*, 1909–1919.
12. Atzmon, Y.; Kreuk, F.; Shalit, U.; et al. A Causal View of Compositional Zero-Shot Recognition. In *Proceedings of the Advances in Neural Information Processing Systems (NeurIPS)*, Online, 6–12 December 2020; pp. 1462–1473.
13. Purushwalkam, S.; Nickel, M.; Gupta, A.; et al. Task-Driven Modular Networks for Zero-Shot Compositional Learning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, Seoul, Republic of Korea 27 October–2 November 2019; pp. 3593–3602.
14. Wang, X.; Yu, F.; Darrell, T.; et al. Task-Aware Feature Generation for Zero-Shot Compositional Learning. *Computer Vision and Pattern Recognition. arXiv* **2019**, arXiv:1906.04854.
15. Nagarajan, T.; Grauman, K. Attributes as Operators: Factorizing Unseen Attribute-Object Compositions. *Computer Vision and Pattern Recognition. arXiv* **2018**, arXiv:1803.09851.
16. Nan, Z.; Liu, Y.; Zheng, N.; et al. Recognizing Unseen Attribute-Object Pair with Generative Model. *Proc. AAAI Conf. Artif. Intell.* **2019**, *33*, 8811–8818.
17. Yang, M.; Deng, C.; Yan, J.; et al. Learning Unseen Concepts via Hierarchical Decomposition and Composition. In *Proceedings of the Computer Vision and Pattern Recognition*, Seattle, WA, USA, 13–19 June 2020.
18. Biederman, I. Recognition-by-Components: A Theory of Human Image Understanding. *Psychol. Rev.* **1987**, *94*, 115.
19. Hoffman, D.D.; Richards, W.A. Parts of Recognition. *Cognition* **1984**, *18*, 65–96.
20. Bach, S.H.; Sanh, V.; Yong, Z.-X.; et al. Promptsource: An Integrated Development Environment and Repository for Natural Language Prompts. *arXiv* **2022**, arXiv:2202.01279.
21. Bommasani, R.; Hudson, D.A.; Adeli, E.; et al. On the Opportunities and Risks of Foundation Models. *arXiv* **2021**, arXiv:2108.07258.
22. Brown, T.; Mann, B.; Ryder, N.; et al. Language Models Are Few-Shot Learners. *Adv. Neural Inf. Process. Syst.* **2020**, *33*, 1877–1901.
23. Sanh, V.; Webson, A.; Raffel, C.; et al. Multitask Prompted Training Enables Zero-Shot Task Generalization. *arXiv* **2021**, arXiv:2110.08207.
24. Vu, T.; Lester, B.; Constant, N.; et al. Spot: Better Frozen Model Adaptation Through Soft Prompt Transfer. *arXiv* **2021**, arXiv:2110.07904.
25. Zhou, K.; Yang, J.; Loy, C.C.; et al. Learning to Prompt for Vision-Language Models. *Int. J. Comput. Vis.* **2022**, *130*, 2337–2348.
26. Qin, G.; Eisner, J. Learning How to Ask: Querying LMs with Mixtures of Soft Prompts. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, Online, 6–11 June 2021; pp. 5203–5212.
27. Hu, E.J.; Shen, Y.; Wallis, P.; et al. LoRA: Low-Rank Adaptation of Large Language Models. In *International conference on learning representations (ICLR)*. *arXiv* **2022**, arXiv:2106.09685.
28. Houlsby, N.; Giurgiu, A.; Jastrzebski, S.; et al. Parameter-Efficient Transfer Learning for NLP. In *Proceedings of the 36th International Conference on Machine Learning (ICML)*, Long Beach, CA, USA, 9–15 June 2019; pp. 2790–2799.

29. Li, X.L.; Liang, P. Prefix-Tuning: Optimizing Continuous Prompts for Generation. In Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics (ACL), Online 1–6 August 2021; pp. 4582–4597.
30. Li, E.; Zeng, L.; Zhou, Z.; et al. On-Demand Accelerating Deep Neural Network Inference via Edge Computing. *IEEE Trans. Wirel. Commun.* **2019**, *19*, 447–457.
31. Vasu, P.K.A.; Pouransari, H.; Faghri, F.; et al. MobileCLIP: Fast Image-Text Models Through Multi-Modal Reinforced Training. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Seattle, WA, USA, 17–21 June 2024; pp. 15963–15974.
32. Wu, K.; Peng, H.; Zhou, Z.; et al. TinyCLIP: CLIP Distillation via Affinity Mimicking and Weight Inheritance. In Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV), Paris, France, 1–6 October 2023; pp. 21970–21980.
33. Guo, T.; Guo, S.; Wang, J.; et al. PromptFL: Let Federated Participants Cooperatively Learn Prompts Instead of Models—Federated Learning in Age of Foundation Model. *IEEE Trans. Mob. Comput.* **2023**, *23*, 1–14.
34. Xian, Y.; Lampert, C.H.; Schiele, B.; et al. Zero-Shot Learning—A Comprehensive Evaluation of the Good, the Bad and the Ugly. *IEEE Trans. Pattern Anal. Mach. Intell.* **2019**, *41*, 2251–2265.
35. Naeem, M.F.; Xian, Y.; Tombari, F.; et al. Learning Graph Embeddings for Compositional Zero-Shot Learning. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Nashville, TN, USA, 20–25 June 2021; pp. 953–962.
36. Isola, P.; Lim, J.J.; Adelson, E.H. Discovering States and Transformations in Image Collections. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Nashville, TN, USA, 20–25 June 2015; pp. 1383–1391.
37. Yu, A.; Grauman, K. Fine-Grained Visual Comparisons with Local Learning. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Columbus, OH, USA, 23–28 June 2014; pp. 192–199.
38. Mancini, M.; Naeem, M.F.; Xian, Y.; et al. Open World Compositional Zero-Shot Learning. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition; Nashville, TN, USA, 20–25 June 2021; pp. 5222–5230.
39. Paszke, A.; Gross, S.; Massa, F.; et al. Pytorch: An Imperative Style, High-Performance Deep Learning Library. *Adv. Neural Inf. Process. Syst.* **2019**, *32*, 8026–8037.
40. Nagarajan, T.; Grauman, K. Attributes as Operators: Factorizing Unseen Attribute-Object Compositions. In Proceedings of the European Conference on Computer Vision (ECCV), Munich, Germany, 8–14 September 2018; pp. 169–185.
41. Zhang, T.; Liang, K.; Du, R.; et al. Learning Invariant Visual Representations for Compositional Zero-Shot Learning. In Proceedings of the Computer Vision—ECCV 2022: 17th European Conference, Tel Aviv, Israel, 23–27 October 2022; proceedings, part XXIV.; Springer: Berlin/Heidelberg, Germany, 2022; pp. 339–355.
42. Reddi, V.J.; Kanter, D.; Mattson, P.; et al. MLPerf Mobile Inference Benchmark: An Industry-Standard Open-Source Machine Learning Benchmark for on-Device AI. In Proceedings of the Machine Learning and Systems (MLSys), Santa Clara, CA, USA, 29 August–1 September 2022.