TEACHING PRETRAINED EVENT ENCODERS TO DIGEST ANY REPRESENTATION

Anonymous authors
Paper under double-blind review

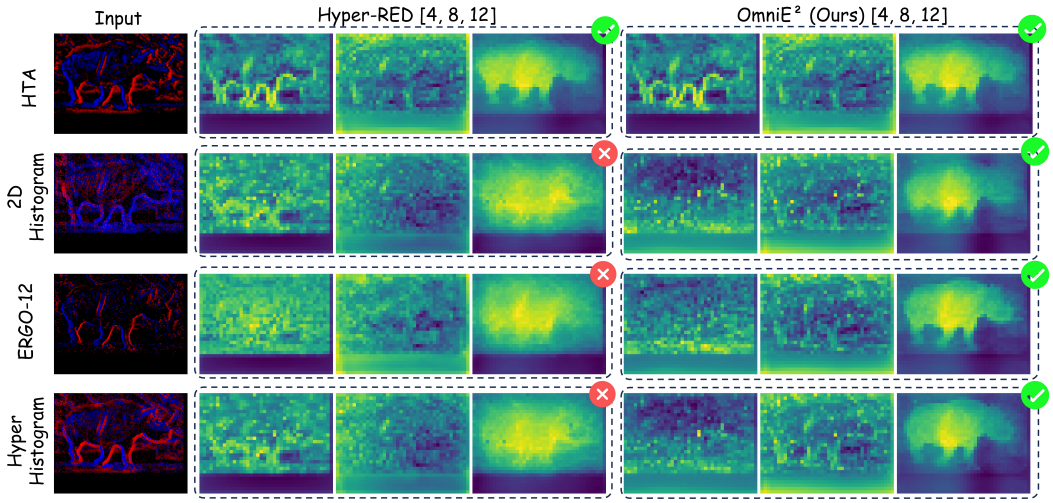


Figure 1: **OmniE^2 adapts pretrained event encoders across representations with original parameters frozen.** Rows show representations of the same event stream (left) and feature heatmaps at Transformer blocks 4, 8, and 12 from HTA-pretrained Hyper-RED (middle) and OmniE^2 (right). Non-HTA inputs yield more diffuse final-block responses in Hyper-RED.

ABSTRACT

Pretrained event encoders have demonstrated strong transferability across a wide range of event-based vision tasks. Existing methods mainly focus on developing stronger pretraining architectures or distilling knowledge from powerful image encoders into the event domain, but typically rely on a fixed representation and overlook the differences among event representations. We observe that changing the input representation of a pretrained event encoder can substantially degrade downstream performance, even when the underlying event stream remains unchanged. We term this phenomenon *representation lock-in*. To address this issue, we present Omnivorous Event Encoder (OmniE^2), a pretraining framework for adapting pretrained event encoders across diverse representations through global and dense feature alignment. To accommodate unseen representations, we further introduce the Cross-Representation Adapter (CRA), which uses a shared operator-conditioned generator to produce native input projections and low-rank corrections to early Transformer blocks while keeping the pretrained event encoder frozen, thereby enabling its semantic knowledge to transfer across representations. Experiments across diverse event representations and datasets show that OmniE^2 substantially improves cross-representation transfer, achieving consistent gains in semantic segmentation, object classification, and depth estimation. [CODE]

1 INTRODUCTION

Pretraining learns reusable visual priors from broader data, providing a shared foundation for diverse event-based tasks while reducing reliance on downstream supervision (Liu et al., 2025). This

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.

is particularly valuable for event vision, where sparse, asynchronous streams of brightness changes (Gallego et al., 2020; Chakravarthi et al., 2024) make robust feature learning challenging with limited task-specific annotations (Wu et al., 2024).

Existing event-based pretraining primarily uses self-supervised learning (Mohammadi et al., 2025; Zhu et al., 2025) and knowledge distillation from visual foundation models (Chen et al., 2026; Cao et al., 2026; Bartolomei et al., 2025) to learn transferable features for downstream tasks. However, existing methods typically assume a fixed representation during pretraining and transfer, despite the diversity of event representations. This leaves a basic requirement unresolved: reusing pretrained knowledge when the event representation changes. With the same downstream representation and training protocol, an HTA-pretrained encoder consistently underperforms a representation-matched encoder (Fig. 2). We term this transfer degradation *representation lock-in*, which limits the reuse of pretrained knowledge across event representations. This creates a practical dilemma between exploiting strong pretrained features and accommodating downstream event representations that often differ from those used during pretraining. We therefore ask: *can we adapt a pretrained event encoder to diverse event representations while largely preserving its feature extraction capabilities?*

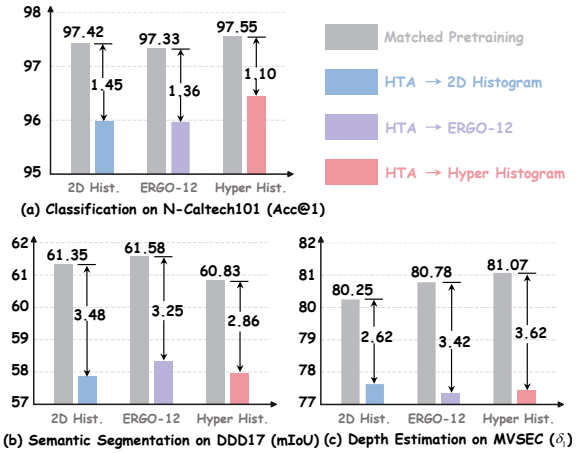


Figure 2: **Downstream evaluation across event representations.** HTA-to-target transfer (colored bars) yields lower performance than representation-matched pretraining (gray bars) across classification, semantic segmentation, and depth estimation.

To explore this possibility, we draw inspiration from efforts to adapt image-pretrained models to heterogeneous inputs. ViT-Lens (Lei et al., 2024) and the Omnivorous Vision Encoder (Kabra et al., 2026) demonstrate that adaptation and feature alignment can extend pretrained visual knowledge to new input modalities. In event vision, alternative representations of the same event stream provide paired views, allowing features from the original pretraining representation to guide adaptation. This motivates combining lightweight input and feature adaptation with cross-representation alignment, while keeping all original encoder parameters frozen.

To this end, we introduce *Omnivorous Event Encoder* (OmniE^2), a cross-representation pretraining framework that extends pretrained event encoders to diverse input representations. It combines global alignment for sample-level semantic consistency with dense alignment of spatially corresponding patch features, using the original pretraining representation as reference. To accommodate unseen representations, we introduce the *Cross-Representation Adapter* (CRA), which describes representations through channel-wise construction rules rather than discrete representation identities and uses a shared operator-conditioned generator to produce native input projections and low-rank corrections to early Transformer blocks. Experiments across representations and datasets demonstrate consistent gains in classification, semantic segmentation, and depth estimation.

Our contributions are fourfold:

- We identify and characterize *representation lock-in*, revealing that mismatches between pretraining and downstream event representations limit the transferability of pretrained event encoders.
- We propose OmniE^2, a cross-representation pretraining framework that enables pretrained event encoders to support diverse representations through global and dense alignment.
- We introduce CRA, a shared operator-conditioned adapter that generates input projections for each representation's native channels and low-rank corrections to early Transformer blocks, while keeping all original encoder parameters frozen during adapter training.
- We evaluate OmniE^2 across diverse event representations on five datasets covering three downstream tasks: object classification, semantic segmentation, and monocular depth estimation. The results demonstrate strong performance in cross-representation transfer.

2 Related Work

Representation learning is central to event-based visual understanding, yet diverse event encodings complicate the reuse of pretrained knowledge. We review three complementary directions: event representation learning, pretraining for event vision, and cross-representation and parameter-efficient adaptation, focusing on transferring pretrained semantics across alternative representations of the same event stream.

2.1 Event Representation Learning

Event Representations. Event streams are commonly converted into frames, histograms or time surfaces (Gallego et al., 2020). HATS (Sironi et al., 2018) aggregates local time surfaces, while voxel grids (Zhu et al., 2019) discretize events over time. Hyper Histogram (Peng et al., 2023) preserves polarity and temporal statistics, whereas HTA (Wang et al., 2026a) aggregates intra- and inter-window temporal information into three channels. Task-driven methods incorporate downstream objectives: EST (Gehrig et al., 2019) jointly learns representations and downstream networks, while ERGO (Zubić et al., 2023) selects task-relevant event measurements to construct ERGO-12.

Remark. Existing event representations emphasize different aspects of event streams, with no universally adopted encoding. This diversity can lead to mismatches between the representations used during pretraining and those available in downstream datasets, hindering direct reuse of the rich semantic knowledge learned by pretrained event encoders.

2.2 Pretraining for Event Vision

Event Self-Supervision. Event pretraining methods adapt self-supervised objectives to sparse and asynchronous event data. MEM (Klenk et al., 2024) reconstructs masked event content, while ECDDP (Yang et al., 2024b) models contextual relations among event patches. STP (Liang et al., 2025) improves pretraining efficiency through prompt-based context fusion, and TESPEC (Mohammadi et al., 2025) emphasizes long-range temporal modeling.

Cross-modal Event Pretraining. Image-to-event transfer provides an alternative source of supervision. EvDistill (Wang et al., 2021) combines image reconstruction with bidirectional distillation, ECDP (Yang et al., 2023) aligns synchronized image–event views, and ESS (Sun et al., 2022) transfers image-based supervision to event segmentation. Recent methods further distill patch-level or dense features from visual foundation models, including EventSAM (Chen et al., 2024), ScaleEvent (Chen et al., 2026) and Hyper-RED (Wang et al., 2026b).

Remark. These methods learn strong event encoders, but generally assume a fixed event representation during pretraining and transfer. Our work instead investigates whether pretrained semantic knowledge can be preserved when the input representation changes, without retraining the backbone.

2.3 Cross-Representation and Parameter-Efficient Adaptation

Unified Representation Learning. Unified representation learning aims to capture shared semantics across heterogeneous inputs. ImageBind (Girdhar et al., 2023) and UniBind (Lyu et al., 2024) align multiple modalities in shared embedding spaces, with UniBind introducing LLM-augmented alignment centers. ViT-Lens (Lei et al., 2024) connects modality-specific inputs to a frozen pretrained ViT, while the Omnivorous Vision Encoder (Kabra et al., 2026) adapts DINOv2 through cross-modal alignment and distillation from a frozen teacher.

Remark. These studies suggest that feature alignment and lightweight adaptation can facilitate pretrained knowledge transfer across heterogeneous inputs, motivating their application to alternative representations of the same event stream.

3 Analysis and Theoretical Motivation

We characterize representation lock-in through feature responses and downstream performance (Figs. 1 and 2), then derive a bound linking cross-representation feature discrepancy to downstream risk deviation, motivating feature alignment.

3.1 Characterizing Representation Lock-in

Comparison setup. To isolate the effect of pretraining–representation mismatch, we compare two encoders for each target representation: one pretrained directly on that representation and one pretrained on HTA and subsequently transferred to it. The matched and HTA-pretrained encoders use the same architecture and are pretrained with identical data, objectives, and training schedules.

Observation 1: Performance drops under representation mismatch. In Fig. 2, HTA-to-target transfer underperforms representation-matched pretraining across all nine task–representation combinations. Classification Acc@1 drops by 1.10–1.45 percentage points on N-Caltech101, and segmentation mIoU drops by 2.86–3.48 points on DDD17, with the same trend in depth estimation.

Observation 2: More diffuse feature responses under representation mismatch. Fig. 1 compares feature heatmaps at blocks 4, 8, and 12 of an HTA-pretrained Hyper-RED encoder for different representations of the same event stream. Compared with the matched HTA input, direct inference on non-HTA representations produces less distinct object structure at block 4 and more spatially diffuse responses at block 12.

Interpretation and motivation. Because each downstream comparison fixes the target representation, the performance gaps suggest that transfer depends on compatibility with the pretraining encoding. We call this dependence *representation lock-in*. The persistent feature differences and closer heatmaps obtained with OmniE^2 suggest a partially correctable mismatch within the encoder. These observations motivate lightweight adaptation to reduce cross-representation feature discrepancies while keeping the pretrained backbone frozen. The following theoretical analysis relates the remaining feature discrepancy to a bound on downstream risk deviation.

3.2 Theoretical Motivation

We relate cross-representation feature discrepancies before downstream fine-tuning to the expected task loss under a shared fixed predictor.

Let $(\mathcal{E}, y) \sim \mathcal{P}$ denote an event sequence and its supervision target, with HTA and target representations $g_0(\mathcal{E})$ and $g_r(\mathcal{E})$. Keeping the original backbone parameters frozen, denote the HTA reference feature by $\mathbf{z}_0$, the direct-transfer feature by $\mathbf{z}_r^{\mathrm{fr}}$, and the feature produced by adaptation mechanism A by $\mathbf{z}_r^A$. We define

$$\Delta_r^{\mathrm{fr}} := \left\|\mathbf{z}_r^{\mathrm{fr}} - \mathbf{z}_0\right\|_{\mathcal{P}}, \qquad \Delta_r(A) := \left\|\mathbf{z}_r^A - \mathbf{z}_0\right\|_{\mathcal{P}}, \tag{1}$$

where $\|\mathbf{v}\|_{\mathcal{P}} := (\mathbb{E}_{\mathcal{P}}[\|\mathbf{v}\|_2^2])^{1/2}$. For a shared fixed predictor h and loss ℓ, define $\mathcal{R}_r(A) := \mathbb{E}_{\mathcal{P}}[\ell(h(\mathbf{z}_r^A), y)]$, with $\mathcal{R}_0$ and $\mathcal{R}_r^{\mathrm{fr}}$ defined analogously.

Theorem 3.1 (Feature-discrepancy risk bound)**.** *Assume paired representations share the same supervision target and $\ell(h(\cdot), y)$ is uniformly L-Lipschitz with respect to the Euclidean feature norm, where $L > 0$. Then, for any adaptation mechanism A with finite risks and feature discrepancies,*

$$|\mathcal{R}_r(A) - \mathcal{R}_0| \le L\Delta_r(A), \qquad \left|\mathcal{R}_r^{\mathrm{fr}} - \mathcal{R}_0\right| \le L\Delta_r^{\mathrm{fr}}. \tag{2}$$

Theorem 3.1 shows that $\Delta_r(A) < \Delta_r^{\mathrm{fr}}$ yields a smaller upper bound on deviation from the HTA reference risk. This motivates feature alignment without guaranteeing lower task loss or gains after downstream fine-tuning. Our global and dense InfoNCE objectives encourage sample- and patch-level agreement, but are not assumed to directly minimize $\Delta_r(A)$. The choice to adapt the input interface and early Transformer blocks is motivated by the observed feature discrepancies rather than implied by the bound. The proof is provided in the Appendix A.

4 Method

Our goal is to enable pretrained event encoders to process diverse event representations while preserving their semantic knowledge. As illustrated in Fig. 3, OmniE^2 learns a shared Cross-Representation Adapter (CRA) that generates input projections and early low-rank corrections conditioned on representation construction rules. CRA is trained through global and dense feature alignment with the HTA reference, while all original encoder parameters remain frozen during adaptation.

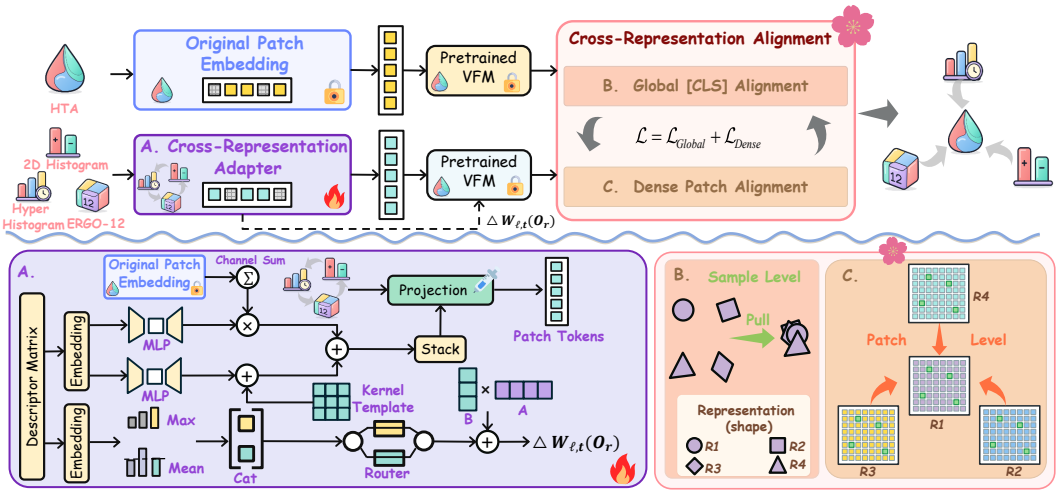


Figure 3: **Overview of OmniE^2.** An HTA reference and a target representation of the same event stream share a pretrained event encoder. **(A)** CRA generates target-specific patch projections and early low-rank corrections from operator descriptors, while HTA follows the original encoder path. **(B)** Global `[CLS]` alignment encourages sample-level consistency, and **(C)** dense alignment matches spatially corresponding patch features.

4.1 Cross-Representation Adapter

To alleviate representation lock-in, CRA conditions input projection and early feature adaptation on representation construction rules, while keeping all pretrained encoder parameters frozen. For an input $X_r \in \mathbb{R}^{C_r \times H \times W}$, we construct a descriptor matrix $O_r = [o_{r,1}, \ldots, o_{r,C_r}]^\top \in \mathbb{R}^{C_r \times 17}$. Each row encodes a channel's polarity, measured quantity, aggregation rule, and partition information. These descriptors depend on the representation operator rather than sample content, and condition weight generators shared across representations. Further details are provided in Appendix B.

CRA first generates a patch projection that accommodates the native channel configuration. Let $W^0 \in \mathbb{R}^{D \times 3 \times P \times P}$ be the frozen HTA patch kernel, and define $\bar{W}^0 = \sum_{q=1}^{3} W^0_{:,q,:,:}$. A learnable descriptor embedding MLP ϕ, shared across channels and representations, is applied row-wise to O_r to produce $h_{r,c} = \phi(o_{r,c})$. Two prediction heads then generate channel weights $\alpha_{r,c} = \mathrm{softmax}_c(a(h_{r,c}))$ and residual coefficients $\beta_{r,c,k} = \tanh(b_k(h_{r,c}))/C_r$. The resulting kernel is

$$W^{(r)}_{:,c,:,:} = \alpha_{r,c}\bar{W}^0 + \sum_{k=1}^{8} \beta_{r,c,k} B_k, \tag{3}$$

where $B_k \in \mathbb{R}^{D \times P \times P}$ are trainable bases shared across representations. This parameterization combines the pretrained projection with corrections determined by channel semantics. Following channel-wise standardization, a stride-P convolution with $W^{(r)}$ and the frozen original bias maps X_r directly to patch tokens $Z_0 \in \mathbb{R}^{N \times D}$, where N is the number of patches.

To accommodate feature shifts beyond input projection, CRA also generates low-rank corrections for the first n Transformer blocks, with $n = 4$ by default. A separate learnable descriptor embedding MLP ψ is applied row-wise to O_r, producing $\tilde{h}_{r,c} = \psi(o_{r,c})$. The two embedding MLPs have independent parameters, while each is shared across channels and representations. We summarize the embeddings $\{\tilde{h}_{r,c}\}_{c=1}^{C_r}$ through mean and max pooling, concatenated with a log-scaled channel count. A shared router maps this summary to block-wise coefficients $\gamma_\ell(O_r) = [1, \tanh v_{\ell,2}, \ldots, \tanh v_{\ell,4}]$. For a linear projection t in block ℓ, the effective weight becomes $W^0_{\ell,t} + \Delta W_{\ell,t}(O_r)$, where

$$\Delta W_{\ell,t}(O_r) = \frac{1}{8} \sum_{k=1}^{4} \gamma_{\ell,k}(O_r) L^{\mathrm{out}}_{\ell,t,k} L^{\mathrm{in}}_{\ell,t,k}. \tag{4}$$

Each factor pair defines a rank-at-most-eight component. The fixed first coefficient provides a common correction, while the remaining coefficients introduce representation-dependent adjustments.

We apply these updates to the joint query/key/value projection, the attention output projection, and both feed-forward projections. Projections within a block share mixing coefficients but retain separate low-rank factors.

We initialize $\alpha_{r,c} = 1/C_r$, the patch residual bases to zero, and the low-rank output factors to zero. Thus, adaptation starts from the baseline's mean-inflated patch projection with no Transformer corrections. Only CRA's generators and basis parameters are optimized; the original HTA path bypasses CRA.

4.2 Cross-Representation Training

Training pairs an HTA reference with a non-HTA view of the same sample, aligned in spatial coordinates and reference timestamp. At each step, we select one non-HTA representation, cycling through the training representations to optimize the shared generator. The original HTA encoder supplies fixed reference features. Thus, each selected representation is supervised directly by HTA, with gradients updating only the adapter parameters.

We align the final features at global and patch levels using symmetric InfoNCE with a fixed temperature $\tau = 0.07$. Let $\mathbf{c}_i^a$ denote the ℓ_2-normalized final `[CLS]` feature of sample i from branch a, where $a = 0$ denotes the HTA reference and $a = r$ the adapted target branch. For a distributed batch of B samples, the global objective matches the two views of the same sample and uses other samples as negatives:

$$\mathcal{L}_{\text{global}} = -\frac{1}{2B} \sum_{(a,b)\in\mathcal{D}_r} \sum_{i=1}^{B} \log \frac{\exp(\langle \mathbf{c}_i^a, \mathbf{c}_i^b \rangle/\tau)}{\sum_{j=1}^{B} \exp(\langle \mathbf{c}_i^a, \mathbf{c}_j^b \rangle/\tau)}, \tag{5}$$

where $\mathcal{D}_r = \{(0, r), (r, 0)\}$ specifies the two alignment directions. HTA features remain detached in both directions.

To align local features, we sample $M = 64$ spatial positions per sample and use matching position indices in both views. Let $\mathbf{d}_{i,p}^a$ denote the ℓ_2-normalized final patch feature at sampled position p. For query (i, p), the candidate set is $\mathcal{C}_{i,p} = \{(i, p)\} \cup \{(j, q) : j \neq i,\ q = 1, \ldots, M\}$, where j ranges over batch samples. This set contains the corresponding positive and sampled patches from other samples; non-corresponding patches within the same sample are excluded. The dense objective is

$$\mathcal{L}_{\text{dense}} = -\frac{1}{2BM} \sum_{(a,b)\in\mathcal{D}_r} \sum_{i=1}^{B} \sum_{p=1}^{M} \log \frac{\exp(\langle \mathbf{d}_{i,p}^a, \mathbf{d}_{i,p}^b \rangle/\tau)}{\sum_{(j,q)\in\mathcal{C}_{i,p}} \exp(\langle \mathbf{d}_{i,p}^a, \mathbf{d}_{j,q}^b \rangle/\tau)}. \tag{6}$$

We optimize the objective $\mathcal{L}_{\text{CRA}} = \lambda_g \mathcal{L}_{\text{global}} + \lambda_d \mathcal{L}_{\text{dense}}$, combining sample-level discrimination with local correspondence.

Because the HTA path bypasses the adapter and remains frozen, its features are preserved exactly throughout adapter pretraining. For a fixed target representation, the descriptors are constant, so the generated patch kernel and low-rank updates can be materialized once for downstream use. Inference then requires only the target representation.

5 Experiments

5.1 Cross-Representation Retrieval

To assess whether different event representations are mapped into a shared feature space, we perform cross-representation retrieval. Given a query encoded with a source representation (e.g., HTA), the task retrieves the corresponding sample encoded with a target representation (e.g., ERGO-12). Since the query and its target are generated from the same underlying event stream, retrieval performance directly reflects the alignment between their representations.

Evaluation Protocol. We extract features for all samples in the full test sets of DSEC (Gehrig et al., 2021), DDD17 (Binas et al., 2017), MVSEC (Zhu et al., 2018), and N-Caltech101 (Orchard et al., 2015). We consider four event representations: HTA (Wang et al., 2026a), 2D Histogram (Gallego et al., 2020), ERGO-12 (Zubić et al., 2023), and Hyper Histogram (Peng et al., 2023). For

Table 1: **Cross-representation retrieval on full test sets.** N: number of samples. GAP: global average pooling of dense features; TOK: CLS token embedding. Parentheses indicate improvements over Hyper-RED (reductions for MedR).

Dataset	Feature	Model	R@1 ↑	R@5 ↑	mAP ↑	MedR ↓
DSEC ($N = 7{,}167$)	GAP	Hyper-RED ViT-B/16	41.28	66.97	52.85	4.58
		+ OmniE^2 ViT-B/16	59.70 (+18.42)	77.38 (+10.41)	67.69 (+14.84)	3.91 (+0.67)
	TOK	Hyper-RED ViT-B/16	49.08	71.90	59.32	3.00
		+ OmniE^2 ViT-B/16	75.33 (+26.25)	94.15 (+22.25)	83.31 (+23.99)	1.25 (+1.75)
DDD17 ($N = 3{,}890$)	GAP	Hyper-RED ViT-B/16	43.71	61.46	52.21	6.58
		+ OmniE^2 ViT-B/16	56.03 (+12.32)	69.19 (+7.73)	62.48 (+10.27)	6.16 (+0.42)
	TOK	Hyper-RED ViT-B/16	49.19	64.18	56.45	6.33
		+ OmniE^2 ViT-B/16	63.40 (+14.21)	79.42 (+15.24)	70.69 (+14.24)	2.50 (+3.83)
MVSEC ($N = 21{,}125$)	GAP	Hyper-RED ViT-B/16	33.34	43.61	38.48	165.75
		+ OmniE^2 ViT-B/16	53.16 (+19.82)	61.71 (+18.10)	57.59 (+19.11)	29.00 (+136.75)
	TOK	Hyper-RED ViT-B/16	39.98	52.33	46.13	33.41
		+ OmniE^2 ViT-B/16	60.73 (+20.75)	75.33 (+23.00)	67.55 (+21.42)	3.33 (+30.08)
N-Caltech101 ($N = 1{,}887$)	GAP	Hyper-RED ViT-B/16	45.01	59.84	52.30	8.50
		+ OmniE^2 ViT-B/16	92.60 (+47.59)	97.73 (+37.89)	94.91 (+42.61)	1.00 (+7.50)
	TOK	Hyper-RED ViT-B/16	49.93	64.48	56.98	5.75
		+ OmniE^2 ViT-B/16	95.68 (+45.75)	98.79 (+34.31)	97.08 (+40.10)	1.00 (+4.75)

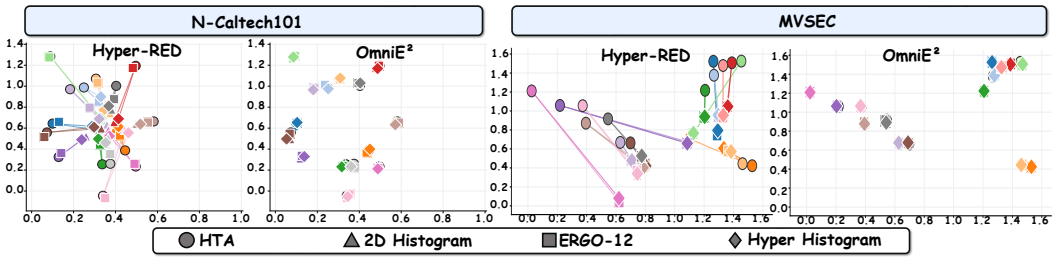


Figure 4: **Cross-representation feature visualization on N-Caltech101 and MVSEC.** Colors denote samples, and marker shapes denote event representations. Compared with Hyper-RED, OmniE^2 produces tighter clusters of same-colored points across different marker shapes.

each sample, we extract the final-layer `[CLS]` token embedding (TOK) and apply global average pooling over the spatial patch tokens (GAP). All feature vectors are ℓ_2-normalized, and the pairwise cosine similarities between the query and gallery sets are computed. We report standard retrieval metrics, including Recall at k (R@k, $k \in \{1, 5\}$), mean Average Precision (mAP), and Median Rank (MedR). The four representations form six unordered combinations, each evaluated in both directions, resulting in 12 directed representation pairs. To evaluate the overall consistency of the shared feature space, Table 1 reports the macro-average over all 12 directions.

Results. Table 1 shows that OmniE^2 consistently improves all retrieval metrics over frozen Hyper-RED across four datasets and both feature readouts. On MVSEC, MedR decreases from 165.75 to 29.00 with GAP and from 33.41 to 3.33 with TOK. N-Caltech101 exhibits the largest R@1 gains, increasing from 45.01% to 92.60% with GAP and from 49.93% to 95.68% with TOK. These consistent gains indicate more reliable matching of corresponding samples across event representations.

Feature Visualization. Fig. 4 provides a qualitative view of cross-representation alignment on N-Caltech101 and MVSEC. With Hyper-RED, features from different representations of the same sample are dispersed. OmniE^2 brings these corresponding features into compact groups, while different samples remain distinguishable. This pattern is consistent with the retrieval improvements, suggesting that OmniE^2 reduces representation-induced feature discrepancies while retaining information useful for distinguishing samples.

5.2 Cross-Representation Transfer Across Datasets and Tasks

We evaluate whether our adapter alleviates representation lock-in beyond feature retrieval while preserving the semantic and geometric knowledge of the pretrained event encoder. An HTA-pretrained Hyper-RED backbone is evaluated on semantic segmentation, object classification, and depth estimation across HTA and three alternative event representations. Unless specified otherwise, averages are computed over the non-HTA representations. All variants follow the same downstream protocol.

Table 2: **Downstream performance across event representations.** We evaluate an HTA-pretrained Hyper-RED backbone on HTA and three non-HTA representations. DSEC and DDD17 are evaluated for semantic segmentation, while N-Caltech101 is evaluated for object classification.

Input	Model	DSEC		DDD17		N-Caltech101	
		Acc. ↑	mIoU ↑	Acc. ↑	mIoU ↑	Acc@1 ↑	Acc@5 ↑
HTA	Hyper-RED	**93.14**	**65.35**	92.52	**65.58**	**97.62**	99.68
	+ OmniE^2	93.03	65.17	**92.58**	64.86	97.46	**99.74**
2D Histogram	Hyper-RED	91.96	62.94	90.75	57.87	**95.97**	98.89
	+ OmniE^2	**92.23**	**63.34**	**91.58**	**60.59**	95.76	**98.89**
ERGO-12	Hyper-RED	92.24	63.79	90.64	58.33	95.97	98.99
	+ OmniE^2	**92.53**	**64.77**	**91.29**	**60.97**	**96.66**	**99.52**
Hyper Histogram	Hyper-RED	92.05	62.76	90.21	57.97	96.45	99.26
	+ OmniE^2	**92.51**	**64.34**	**91.42**	**61.24**	**96.61**	**99.42**
Average	Hyper-RED	92.08	63.16	90.53	58.06	96.13	99.05
	+ OmniE^2	**92.43**	**64.15**	**91.43**	**60.93**	**96.34**	**99.28**

Table 3: **Depth estimation performance across event representations.** We evaluate an HTA-pretrained Hyper-RED backbone on HTA and three non-HTA representations.

Input	Model	MVSEC-Depth					DSEC-Depth				
		δ_1 ↑	δ_2 ↑	AbsRel ↓	RMSE ↓	$\text{RMSE}_{\log}$ ↓	δ_1 ↑	δ_2 ↑	AbsRel ↓	RMSE ↓	$\text{RMSE}_{\log}$ ↓
HTA	Hyper-RED	**80.36**	**94.25**	**0.156**	**3.450**	**0.206**	86.39	97.41	0.112	4.194	0.159
	+ OmniE^2	80.15	93.96	0.158	3.488	0.209	**87.13**	**97.63**	**0.109**	**4.090**	**0.155**
2D Histogram	Hyper-RED	77.64	93.95	0.168	3.616	0.215	85.13	97.18	0.119	4.295	0.164
	+ OmniE^2	**78.86**	**94.27**	**0.165**	**3.505**	**0.210**	**86.33**	**97.42**	**0.113**	**4.166**	**0.159**
ERGO-12	Hyper-RED	77.36	94.09	0.172	3.612	0.217	85.99	97.36	0.114	4.240	0.161
	+ OmniE^2	**79.34**	**94.43**	**0.163**	**3.474**	**0.209**	**86.62**	**97.65**	**0.111**	**4.127**	**0.156**
Hyper Histogram	Hyper-RED	77.45	94.10	0.169	3.593	0.215	85.92	97.44	0.116	4.212	0.161
	+ OmniE^2	**79.10**	**94.35**	**0.163**	**3.493**	**0.209**	**86.57**	**97.52**	**0.112**	**4.139**	**0.157**
Average	Hyper-RED	77.48	94.05	0.169	3.607	0.216	85.68	97.33	0.116	4.249	0.162
	+ OmniE^2	**79.10**	**94.35**	**0.164**	**3.490**	**0.209**	**86.51**	**97.53**	**0.112**	**4.144**	**0.158**

Semantic Segmentation. Table 2 shows improvements in accuracy and mIoU for all non-HTA representations. Average mIoU increases from 63.16% to 64.15% on DSEC and from 58.06% to 60.93% on DDD17. The largest gain is 3.27 percentage points for Hyper Histogram on DDD17.

Object Classification. On N-Caltech101, average Acc@1 and Acc@5 across non-HTA representations increase from 96.13% to 96.34% and from 99.05% to 99.28%, respectively. ERGO-12 gains 0.69 Acc@1 points and 0.53 Acc@5 points, while Hyper Histogram also improves both metrics. For 2D Histogram, Acc@1 decreases by 0.21 points and Acc@5 remains unchanged. HTA classification performance changes only slightly. CRA also improves classification with a frozen encoder and shared HTA-trained head (Appendix D.1).

Depth Estimation. Table 3 shows δ_1 gains for all non-HTA representations. On MVSEC-Depth, average δ_1 increases from 77.48% to 79.10%, while RMSE decreases from 3.607 to 3.490. On DSEC-Depth, δ_1 rises from 85.68% to 86.51%, with AbsRel and RMSE decreasing from 0.116 to 0.112 and from 4.249 to 4.144, respectively. ERGO-12 achieves the largest δ_1 gain on MVSEC-Depth at 1.98 percentage points. MVSEC δ_2 also improves for all non-HTA representations, with its average increasing from 94.05% to 94.35%.

5.3 Unseen-Representation Transfer

To evaluate generalization to unseen representations, we completely exclude each target from OmniE^2 pretraining. We use leave-one-representation-out pretraining for the three registered targets (2D Histogram, ERGO-12, and Hyper Histogram). We additionally test Time Surface (Gallego et al., 2020) and SBT-5 (Wang et al., 2019), which are absent from both the representation registry and pretraining. We compare against the mean-inflated Hyper-RED baseline under the same

Table 4: **Unseen-representation transfer on DDD17 and N-Caltech101.** Targets are excluded from OmniE^2 pretraining and evaluated after supervised fine-tuning; novel targets are also absent from the registry. C: input channels.

Target	Model	DDD17		N-Caltech101	
		Acc. ↑	mIoU ↑	Acc@1 ↑	Acc@5 ↑
Registered targets					
2D Histogram ($C = 2$)	Hyper-RED	90.75	57.87	**95.97**	**98.89**
	+ OmniE^2	**91.47**	**61.17**	94.97	98.41
ERGO-12 ($C = 12$)	Hyper-RED	90.64	58.33	95.97	**98.99**
	+ OmniE^2	**90.76**	**59.54**	**96.08**	**98.99**
Hyper Histogram ($C = 16$)	Hyper-RED	90.21	57.97	96.45	**99.26**
	+ OmniE^2	**91.27**	**60.41**	**96.50**	99.21
Novel targets					
Time Surface ($C = 2$)	Hyper-RED	90.24	58.26	95.44	98.73
	+ OmniE^2	**90.66**	**59.41**	**96.50**	**99.58**
SBT-5 ($C = 5$)	Hyper-RED	**91.17**	60.08	94.12	98.52
	+ OmniE^2	90.99	**60.68**	**97.19**	**99.47**

Table 5: **Adapter design ablation.** MVSEC results averaged over all four representations.

Adapter	δ_1 ↑	RMSE ↓	Params. (M)
No Adapter	78.20	3.568	0.00
Basis+LoRA	78.58	3.549	3.15
Patch+LoRA	78.52	3.552	7.47
ID-CRA	79.10	3.522	3.20
CRA (Ours)	**79.36**	**3.490**	3.20

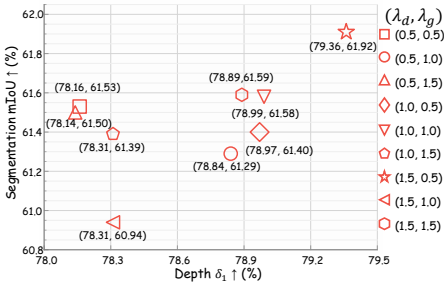


Figure 5: **Ablation on loss weights.**

supervised downstream fine-tuning protocol. This setting evaluates representation-level zero-shot initialization followed by supervised task adaptation.

Table 4 shows that OmniE^2 improves segmentation mIoU for all three held-out registered targets, by 3.30, 1.21, and 2.44 points for 2D Histogram, ERGO-12, and Hyper Histogram, respectively. Classification results are more mixed: ERGO-12 and Hyper Histogram show small Acc@1 gains, whereas 2D Histogram drops by 1.00 percentage point. Thus, transfer benefits depend on both the target representation and the downstream task.

The gains extend to the two novel representations. For Linear Time Surface and SBT-5, Acc@1 improves by 1.06 and 3.07 percentage points, respectively, alongside mIoU gains of 1.15 and 0.60 points. These results suggest that CRA can provide useful initializations beyond the representations encountered during pretraining, including targets outside the predefined registry.

5.4 Ablation Studies

All ablation results are averaged over four event representations: HTA, 2D Histogram, ERGO-12, and Hyper Histogram. We evaluate classification on N-Caltech101 using Acc@1, depth estimation on MVSEC using δ_1, and semantic segmentation on DDD17 using mIoU.

Adapter Design. Table 5 shows that CRA achieves the best depth estimation performance, improving δ_1 by 1.16 percentage points and reducing RMSE by 0.078 over No Adapter. At the same parameter budget, CRA surpasses ID-CRA by 0.26 points in δ_1 and reduces RMSE by 0.032, supporting the benefit of conditioning on representation construction rules. Both shared-LoRA baselines yield smaller gains, including Patch+LoRA with 7.47M parameters compared with CRA's 3.20M. These results suggest that parameter count alone does not explain CRA's advantage and demonstrate a favorable performance–parameter trade-off. Detailed configurations are provided in Appendix C.3.

Alignment Loss Weights. We vary λ_d and λ_g over $\{0.5, 1.0, 1.5\}$. Fig. 5 shows that $(\lambda_d, \lambda_g) = (1.5, 0.5)$ performs best among the nine settings, achieving 79.36% depth δ_1 and 61.92% segmentation mIoU. These exceed equal weighting $(1.0, 1.0)$ by 0.37 and 0.34 points, respectively. Larger weights do not consistently improve performance, suggesting that the balance between global and dense alignment matters. Additional ablation studies are provided in the Appendix D.

6 Conclusion

We introduced OmniE^2 to alleviate *representation lock-in* through operator-conditioned CRA and global and dense feature alignment. CRA pretraining freezes the original encoder and optimizes parameters equivalent to 3.6% of the backbone. Experiments show improved cross-representation retrieval and downstream transfer, including to unseen representations after supervised fine-tuning.

AI USE STATEMENT

Generative AI tools were used to correct grammatical errors, improve the wording and readability of the manuscript, and assist with literature searches for recent methods. All AI-assisted content was reviewed by the authors, who take full responsibility for the manuscript, including its claims and references.

ETHICS STATEMENT

Our experiments use existing publicly available event-vision datasets, without collecting new data or conducting human-subject studies. Deployment in safety-critical applications, such as autonomous driving, requires additional evaluation of robustness and reliability under real-world conditions beyond the reported benchmarks.

REPRODUCIBILITY STATEMENT

Detailed experimental configurations, including data splits, CRA pretraining, adapter comparisons, and downstream fine-tuning, are provided in Appendix C. Code and configuration files are available at `https://anonymous.4open.science/r/OmniE2-3B61`.

REFERENCES

Luca Bartolomei, Enrico Mannocci, Fabio Tosi, Matteo Poggi, and Stefano Mattoccia. Depth anyevent: A cross-modal distillation paradigm for event-based monocular depth estimation. In *2025 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 19669–19678. IEEE, 2025.

Jonathan Binas, Daniel Neil, Shih-Chii Liu, and Tobi Delbruck. Ddd17: End-to-end davis driving dataset. *arXiv preprint arXiv:1711.01458*, 2017.

Yohann Cabon, Naila Murray, and Martin Humenberger. Virtual kitti 2. *arXiv preprint arXiv:2001.10773*, 2020.

Jianwen Cao, Jiaxu Xing, Nico Messikommer, and Davide Scaramuzza. Generative event pretraining with foundation model alignment. *arXiv preprint arXiv:2603.23032*, 2026.

Bharatesh Chakravarthi, Aayush Atul Verma, Kostas Daniilidis, Cornelia Fermuller, and Yezhou Yang. Recent event camera innovations: A survey. In *European conference on computer vision*, pp. 342–376. Springer, 2024.

Zhiwen Chen, Zhiyu Zhu, Yifan Zhang, Junhui Hou, Guangming Shi, and Jinjian Wu. Segment any event streams via weighted adaptation of pivotal tokens. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3890–3900. IEEE, 2024.

Zhiwen Chen, Junhui Hou, Zhiyu Zhu, Jinjian Wu, and Guangming Shi. Scaling dense event-stream pretraining from visual foundation models. *arXiv preprint arXiv:2603.03969*, 2026.

Alexey Dosovitskiy. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

David Eigen, Christian Puhrsch, and Rob Fergus. Depth map prediction from a single image using a multi-scale deep network. *Advances in neural information processing systems*, 27, 2014.

Guillermo Gallego, Tobi Delbrück, Garrick Orchard, Chiara Bartolozzi, Brian Taba, Andrea Censi, Stefan Leutenegger, Andrew J Davison, Jörg Conradt, Kostas Daniilidis, et al. Event-based vision: A survey. *IEEE transactions on pattern analysis and machine intelligence*, 44(1):154–180, 2020.

Daniel Gehrig, Antonio Loquercio, Konstantinos G Derpanis, and Davide Scaramuzza. End-to-end learning of representations for asynchronous event-based data. In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 5633–5643, 2019.

Mathias Gehrig, Willem Aarents, Daniel Gehrig, and Davide Scaramuzza. Dsec: A stereo event camera dataset for driving scenarios. *IEEE Robotics and Automation Letters*, 6(3):4947–4954, 2021.

Rohit Girdhar, Alaaeldin El-Nouby, Zhuang Liu, Mannat Singh, Kalyan Vasudev Alwala, Armand Joulin, and Ishan Misra. Imagebind one embedding space to bind them all. In *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 15180–15190. IEEE, 2023.

Rishabh Kabra, Maks Ovsjanikov, Drew A Hudson, Ye Xia, Skanda Koppula, Andre Araujo, Joao Carreira, and Niloy J Mitra. A mixed diet makes dino an omnivorous vision encoder. *arXiv preprint arXiv:2602.24181*, 2026.

Tommie Kerssies, Niccolo Cavagnero, Alexander Hermans, Narges Norouzi, Giuseppe Averta, Bastian Leibe, Gijs Dubbelman, and Daan De Geus. Your vit is secretly an image segmentation model. In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 25303–25313. IEEE, 2025.

Simon Klenk, David Bonello, Lukas Koestler, Nikita Araslanov, and Daniel Cremers. Masked event modeling: Self-supervised pretraining for event cameras. In *2024 IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pp. 2367–2377. IEEE, 2024.

Weixian Lei, Yixiao Ge, Kun Yi, Jianfeng Zhang, Difei Gao, Dylan Sun, Yuying Ge, Ying Shan, and Mike Zheng Shou. Vit-lens: Towards omni-modal representations. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 26637–26647. IEEE, 2024.

Quanmin Liang, Qiang Li, Shuai Liu, Xinzi Cao, Jinyi Lu, Feidiao Yang, Wei Zhang, Kai Huang, and Yonghong Tian. Efficient event camera data pretraining with adaptive prompt fusion. In *2025 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 8656–8667. IEEE, 2025.

Haotian Liu, Guo Yu, Hu Cao, Sanqing Qu, Fan Lu, Yan Zhong, Zhichao Lu, Luziwei Leng, and Guang Chen. I2ekd: Efficient and versatile image-to-event knowledge distillation. *IEEE Transactions on Circuits and Systems for Video Technology*, 36(1):292–303, 2025.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.

Yuanhuiyi Lyu, Xu Zheng, Jiazhou Zhou, and Lin Wang. Unibind: Llm-augmented unified and balanced representation space to bind them all. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 26742–26752. IEEE, 2024.

Mohammad Mohammadi, Ziyi Wu, and Igor Gilitschenski. Tespec: Temporally-enhanced self-supervised pretraining for event cameras. In *2025 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 7782–7793. IEEE, 2025.

Garrick Orchard, Ajinkya Jayawant, Gregory K Cohen, and Nitish Thakor. Converting static image datasets to spiking neuromorphic datasets using saccades. *Frontiers in neuroscience*, 9:437, 2015.

Yansong Peng, Yueyi Zhang, Peilin Xiao, Xiaoyan Sun, and Feng Wu. Better and faster: Adaptive event conversion for event-based object detection. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pp. 2056–2064, 2023.

Amos Sironi, Manuele Brambilla, Nicolas Bourdis, Xavier Lagorce, and Ryad Benosman. Hats: Histograms of averaged time surfaces for robust event-based object classification. In *2018 IEEE/CVF conference on computer vision and pattern recognition*, pp. 1731–1740. IEEE, 2018.

Zhaoning Sun, Nico Messikommer, Daniel Gehrig, and Davide Scaramuzza. Ess: Learning event-based semantic segmentation from still images. In *European Conference on Computer Vision*, pp. 341–357. Springer, 2022.

Lin Wang, Yo-Sung Ho, Kuk-Jin Yoon, et al. Event-based high dynamic range image and very high frame rate video generation using conditional generative adversarial networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 10081–10090, 2019.

Lin Wang, Yujeong Chae, Sung-Hoon Yoon, Tae-Kyun Kim, and Kuk-Jin Yoon. Evdistill: Asynchronous events to end-task learning via bidirectional reconstruction-guided cross-modal knowledge distillation. In *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 608–619. IEEE, 2021.

Meisen Wang, Hao Deng, Wei Bao, Ma Yuanxiao, Chengjie Wang, Zhiqiang Tian, Shaoyi Du, and Siqi Li. Rethinking event-based object dtection through representation-level temporal aggregation and model-level hypergraph reasoning. *arXiv preprint arXiv:2605.08825*, 2026a.

Meisen Wang, Zhiqiang Tian, Wei Bao, Chengjie Wang, Shaoyi Du, and Siqi Li. Hyper-red: Scalable event pre-training via semantic hypergraph distillation. *arXiv preprint arXiv:2609.16811*, 2026b.

Ziyi Wu, Mathias Gehrig, Qing Lyu, Xudong Liu, and Igor Gilitschenski. Leod: Label-efficient object detection for event cameras. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 16933–16942. IEEE, 2024.

Lihe Yang, Bingyi Kang, Zilong Huang, Zhen Zhao, Xiaogang Xu, Jiashi Feng, and Hengshuang Zhao. Depth anything v2. *Advances in neural information processing systems*, 37:21875–21911, 2024a.

Yan Yang, Liyuan Pan, and Liu Liu. Event camera data pre-training. In *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 10665–10675. IEEE, 2023.

Yan Yang, Liyuan Pan, and Liu Liu. Event camera data dense pre-training. In *European Conference on Computer Vision*, pp. 292–310. Springer, 2024b.

Alex Zihao Zhu, Dinesh Thakur, Tolga Özaslan, Bernd Pfrommer, Vijay Kumar, and Kostas Daniilidis. The multivehicle stereo event camera dataset: An event camera dataset for 3d perception. *IEEE Robotics and Automation Letters*, 3(3):2032–2039, 2018.

Alex Zihao Zhu, Liangzhe Yuan, Kenneth Chaney, and Kostas Daniilidis. Unsupervised event-based learning of optical flow, depth, and egomotion. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 989–997. IEEE, 2019.

Lin Zhu, Ruonan Liu, Xiao Wang, Lizhi Wang, and Hua Huang. Revealing latent information: A physics-inspired self-supervised pre-training framework for noisy and sparse events. In *Proceedings of the 33rd ACM International Conference on Multimedia*, pp. 7490–7499, 2025.

Nikola Zubić, Daniel Gehrig, Mathias Gehrig, and Davide Scaramuzza. From chaos comes order: Ordering event representations for object recognition and detection. In *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 12800–12810. IEEE, 2023.

APPENDIX

CONTENTS OF THE APPENDIX

A Detailed Theoretical Analysis **14**
A.1 Setup and Feature Gaps . . . 14
A.2 Assumptions . . . 14
A.3 Proof of Theorem 3.1 . . . 14
A.4 Learned Adapters and Optimized Heads . . . 15
A.5 Baseline Inclusion in the CRA Function Class . . . 15

B Operator Descriptor Specification **15**
B.1 Descriptor Schema . . . 16
B.2 Partition and Aggregation Conventions . . . 16
B.3 Representations Observed during CRA Pretraining . . . 17
B.4 Unseen Representations . . . 18
B.5 Conditioning the CRA Generators . . . 18
B.6 Reference Encoding . . . 19

C Training and Evaluation Details **19**
C.1 Pretraining Data and Input Processing . . . 19
C.2 CRA Pretraining . . . 20
C.3 Adapter Comparison Configurations . . . 21
C.4 Downstream Evaluation . . . 22
C.5 Cross-Representation Evaluation . . . 23

D More Detailed Results **23**
D.1 Frozen-Encoder Evaluation . . . 23
D.2 Number of Adapted Layers . . . 24
D.3 Alignment Loss Weights . . . 24
D.4 Adapter Design . . . 24
D.5 Qualitative Results . . . 25

E Limitations **26**

A Detailed Theoretical Analysis

A.1 Setup and Feature Gaps

We analyze backbone features before downstream fine-tuning. Let $(\mathcal{E}, y) \sim \mathcal{P}$ denote an event sequence and its supervision target. The HTA reference $g_0(\mathcal{E})$ and target representation $g_r(\mathcal{E})$ are generated from the same event sequence.

Given the frozen pretrained backbone F, define the reference feature as

$$\mathbf{z}_0 := F(g_0(\mathcal{E})). \tag{7}$$

Let $\mathbf{z}_r^{\mathrm{fr}}$ denote the direct-transfer feature, including the input interface required for the target representation, and let $\mathbf{z}_r^A$ denote the feature produced by adaptation mechanism A. In both cases, the original parameters of F remain unchanged. All features lie in a common Euclidean space and denote the inputs to the downstream predictor.

For a feature function $\mathbf{v}(\mathcal{E})$, define

$$\|\mathbf{v}\|_{\mathcal{P}} := \left(\mathbb{E}_{\mathcal{P}}\left[\|\mathbf{v}(\mathcal{E})\|_2^2\right]\right)^{1/2}. \tag{8}$$

The direct-transfer and adapted feature discrepancies are

$$\Delta_r^{\mathrm{fr}} := \left\|\mathbf{z}_r^{\mathrm{fr}} - \mathbf{z}_0\right\|_{\mathcal{P}}, \qquad \Delta_r(A) := \left\|\mathbf{z}_r^A - \mathbf{z}_0\right\|_{\mathcal{P}}. \tag{9}$$

For a shared fixed predictor h and task loss ℓ, define

$$\begin{aligned} \mathcal{R}_0(h) &:= \mathbb{E}_{\mathcal{P}}\left[\ell(h(\mathbf{z}_0), y)\right], \\ \mathcal{R}_r^{\mathrm{fr}}(h) &:= \mathbb{E}_{\mathcal{P}}\left[\ell(h(\mathbf{z}_r^{\mathrm{fr}}), y)\right], \\ \mathcal{R}_r(A, h) &:= \mathbb{E}_{\mathcal{P}}\left[\ell(h(\mathbf{z}_r^A), y)\right]. \end{aligned} \tag{10}$$

The main text suppresses the fixed predictor h in this notation. All risks and feature discrepancies considered below are assumed finite.

A.2 Assumptions

A1: Task-semantic consistency. Paired representations share the same supervision target:

$$p(y \mid \mathcal{E}, r) = p(y \mid \mathcal{E}).$$

This permits paired risk comparisons without requiring each representation to preserve all information in $\mathcal{E}$.

A2: Lipschitz task loss. For the shared fixed predictor h, there exists $L > 0$ such that

$$|\ell(h(\mathbf{z}), y) - \ell(h(\mathbf{z}'), y)| \le L \left\|\mathbf{z} - \mathbf{z}'\right\|_2, \tag{11}$$

uniformly over the relevant reference, direct-transfer, and adapted features and supervision targets. This assumption concerns the task loss, not discontinuous reporting metrics such as Acc@1, mIoU, or δ_1.

A.3 Proof of Theorem 3.1

Under A1, the reference and target risks are evaluated on the same paired sample $(\mathcal{E}, y)$. The triangle inequality, A2, and the Cauchy–Schwarz inequality yield

$$\begin{aligned} |\mathcal{R}_r(A, h) - \mathcal{R}_0(h)| &\le \mathbb{E}_{\mathcal{P}}\left[\left|\ell(h(\mathbf{z}_r^A), y) - \ell(h(\mathbf{z}_0), y)\right|\right] \\ &\le L\mathbb{E}_{\mathcal{P}}\left[\left\|\mathbf{z}_r^A - \mathbf{z}_0\right\|_2\right] \\ &\le L\left(\mathbb{E}_{\mathcal{P}}\left[\left\|\mathbf{z}_r^A - \mathbf{z}_0\right\|_2^2\right]\right)^{1/2} \\ &= L\Delta_r(A). \end{aligned} \tag{12}$$

Applying the same argument to direct transfer gives

$$\left|\mathcal{R}_r^{\mathrm{fr}}(h) - \mathcal{R}_0(h)\right| \le L\Delta_r^{\mathrm{fr}}. \tag{13}$$

Together, Eqs. (12) and (13) establish Theorem 3.1.

Since $L > 0$, the condition $\Delta_r(A) < \Delta_r^{\mathrm{fr}}$ gives a smaller upper bound on deviation from the HTA reference risk. It does not guarantee a reduction in the actual task loss, and whether an adapter satisfies this condition must be assessed empirically.

A.4 Learned Adapters and Optimized Heads

For a learned adapter $\widehat{A}$, treated as fixed when evaluating population risks, the bound becomes

$$\left|\mathcal{R}_r(\widehat{A}, h) - \mathcal{R}_0(h)\right| \le L\Delta_r(\widehat{A}). \tag{14}$$

Our global and dense InfoNCE objectives encourage sample- and patch-level agreement, but are not assumed to directly minimize $\Delta_r(\widehat{A})$. In particular, alignment of normalized features does not by itself establish reduced discrepancy in an unnormalized feature space.

To consider optimized heads while holding each feature map fixed, let $\mathcal{H}$ be a nonempty common predictor class satisfying A2 uniformly with the same constant L. Define the finite optimal population risks

$$\mathcal{R}_0^\star := \inf_{h\in\mathcal{H}} \mathcal{R}_0(h), \qquad \mathcal{R}_r^\star(A) := \inf_{h\in\mathcal{H}} \mathcal{R}_r(A, h). \tag{15}$$

For every $h \in \mathcal{H}$, the uniform bound implies

$$\mathcal{R}_0(h) - L\Delta_r(A) \le \mathcal{R}_r(A, h) \le \mathcal{R}_0(h) + L\Delta_r(A).$$

Taking infima therefore gives

$$|\mathcal{R}_r^\star(A) - \mathcal{R}_0^\star| \le L\Delta_r(A). \tag{16}$$

This compares optimal population risks with fixed feature maps, not the outcomes of finite-sample optimization or end-to-end backbone fine-tuning.

A.5 Baseline Inclusion in the CRA Function Class

Let $\mathcal{A}_{\mathrm{CRA}}$ denote the CRA function class and A_{init} its initialization. With uniform channel weights $\alpha_{r,c} = 1/C_r$, zero patch residual bases, and zero low-rank output factors, CRA reproduces the baseline's mean-inflated patch projection and introduces no Transformer corrections. Under identical input preprocessing, each non-HTA representation therefore satisfies

$$\mathbf{z}_r^{A_{\mathrm{init}}} = \mathbf{z}_r^{\mathrm{fr}}. \tag{17}$$

Since $A_{\mathrm{init}} \in \mathcal{A}_{\mathrm{CRA}}$,

$$\epsilon_r^{\mathrm{CRA}} := \inf_{A\in\mathcal{A}_{\mathrm{CRA}}} \Delta_r(A) \le \Delta_r(A_{\mathrm{init}}) = \Delta_r^{\mathrm{fr}}. \tag{18}$$

Thus, the CRA function class contains a baseline-equivalent solution. This inclusion neither guarantees a strict discrepancy reduction nor ensures that training finds a better solution. The choice to adapt the input interface and early Transformer blocks remains an empirically motivated architectural decision, rather than a consequence of the risk bound.

B Operator Descriptor Specification

This section specifies the operator descriptors used by CRA, including their field order, channel assignments, and representation construction conventions. The same schema is used for the representations observed during CRA pretraining and the unseen targets evaluated in Sec. 5.3. The definitions below describe the implementations used in our experiments.

B.1 Descriptor Schema

For an input representation $X_r \in \mathbb{R}^{C_r \times H \times W}$, we construct $O_r = [o_{r,1}, \ldots, o_{r,C_r}]^\top \in \mathbb{R}^{C_r \times 17}$, with descriptor rows following the input channel order. Each descriptor is computed deterministically from the channel construction rules and remains constant across samples using those rules. The descriptors themselves are not learned; the embedding networks and weight generators conditioned on them are learned during CRA pretraining.

A channel is specified by a polarity selector p, a measurement m, an aggregation rule a, a partition rule q, a normalized support interval $[s, e)$, and a relative-timestamp flag ρ. Its descriptor is

$$o_{r,c} = \mathrm{concat}\left(e_p, e_m, e_a, e_q, s, e, \frac{s+e}{2}, e-s, \rho\right), \tag{19}$$

where e_p, e_m, e_a, e_q are one-hot vectors with dimensions $3, 3, 4, 2$, respectively. Together with the five scalar entries, this gives $3 + 3 + 4 + 2 + 5 = 17$ dimensions. We store descriptors in float32, with $0 \le s < e \le 1$ and $\rho \in \{0, 1\}$.

Table 6: **The 17-dimensional operator descriptor.** Indices are one-based. Each categorical group contains exactly one active entry, with categories ordered as listed.

Dimensions	Field	Ordered categories or definition
1–3	Polarity selector	Negative, positive, both
4–6	Measurement	Count, timestamp, signed polarity
7–10	Aggregation	Sum, mean, maximum, population variance
11–12	Partition	Window time, event index
13	Support start	s
14	Support end	e
15	Support center	$(s+e)/2$
16	Support width	$e-s$
17	Relative timestamp	$\rho = 1$ for relative timestamp measurements; otherwise $\rho = 0$

The polarity selector and measurement are distinct: selecting both polarities and summing counts yields the total number of events, whereas summing signed polarities allows positive and negative events to cancel. Likewise, ρ indicates relative timestamp measurements, not temporal partitioning. SBT-5 uses time bins but accumulates signed polarities, so its flag is zero.

Descriptors contain no representation names or IDs, dataset IDs, class labels, sample event counts, or image statistics. Names used to retrieve metadata do not enter the embedding networks. The channel count C_r is supplied separately to the low-rank conditioning branch.

B.2 Partition and Aggregation Conventions

Let $[T_0, T_1)$ be a sample window with $T_1 > T_0$, containing events (x_i, y_i, t_i, p_i) with polarities mapped to $p_i \in \{-1, +1\}$. For window-time partitioning, $[s, e)$ selects events in

$$[T_0 + s(T_1 - T_0),\ T_0 + e(T_1 - T_0))\,. \tag{20}$$

For event-index partitioning, we discard events outside the sensor bounds and retain temporal order. Among the N remaining events, $[s, e)$ selects zero-based indices from $\lfloor sN \rfloor$ to $\lfloor eN \rfloor - 1$. An event-index interval therefore need not span the corresponding fraction of elapsed time.

We first select the support interval, then apply the polarity selector, and finally aggregate the measurements at each pixel. Count measurements contribute 1 per event, while signed-polarity measurements contribute p_i. Empty pixels return zero. Variance uses the population convention $\mathrm{mean}(v^2) - \mathrm{mean}(v)^2$, with negative values caused by numerical roundoff clamped to zero. Consequently, count with mean aggregation is a binary occupancy statistic: it equals one at nonempty pixels and zero elsewhere.

Partition boundaries are computed from the exact fractions and rounding rules specified below, rather than reconstructed from float32 descriptor entries. The relative-timestamp flag does not encode the

normalization reference: ERGO-12 uses the first and last valid events in the complete sample window, Hyper Histogram uses individual time-block boundaries, and Time Surface uses the sample-window boundaries. These conventions are specified explicitly below. In denominators of the form $\max(\Delta t, 1)$, the constant 1 denotes one unit of the stored timestamp scale.

B.3 Representations Observed during CRA Pretraining

HTA retains its original three-channel input projection and bypasses CRA; we do not construct an HTA descriptor. The three non-HTA training representations use the shared schema in Table 6. Channel indices below are one-based, while block indices are zero-based.

2D Histogram. The two channels count negative and positive events, respectively, over the complete sample window. Their specifications $(p, m, a, q, [s, e), \rho)$ are

$$\begin{aligned} c = 1: &\quad (\text{negative}, \text{count}, \text{sum}, \text{window_time}, [0, 1), 0), \\ c = 2: &\quad (\text{positive}, \text{count}, \text{sum}, \text{window_time}, [0, 1), 0). \end{aligned} \tag{21}$$

ERGO-12. All 12 channels use event-index partitioning. Table 7 specifies their order, measurements, aggregations, and support intervals. For timestamp channels, we first compute

$$\widetilde{t}_i = \frac{t_i - t_{\text{first}}}{\max(t_{\text{last}} - t_{\text{first}}, 1)}, \tag{22}$$

where t_{first} and t_{last} refer to the complete valid event sequence in the sample window. We then select the specified event-index interval and aggregate the selected measurements. Timestamps are not renormalized within each index interval. An empty event window produces an all-zero representation.

Table 7: **Complete ERGO-12 channel specifications.** Every channel uses event-index partitioning. Support centers and widths follow from the listed endpoints.

c	Polarity	Measurement	Aggregation	$[s, e)$	ρ
1	Both	Polarity	Variance	$[0, 1)$	0
2	Negative	Timestamp	Variance	$[2/3, 1)$	1
3	Negative	Count	Mean	$[1/3, 2/3)$	0
4	Both	Polarity	Sum	$[7/8, 1)$	0
5	Positive	Count	Mean	$[3/4, 1)$	0
6	Both	Count	Sum	$[7/8, 1)$	0
7	Positive	Timestamp	Mean	$[1/3, 2/3)$	1
8	Negative	Count	Mean	$[3/4, 1)$	0
9	Negative	Timestamp	Maximum	$[0, 1/3)$	1
10	Positive	Timestamp	Maximum	$[0, 1)$	1
11	Both	Timestamp	Maximum	$[1/2, 1)$	1
12	Both	Count	Mean	$[0, 1/3)$	0

Hyper Histogram. We use $K = 4$ equal-duration blocks, giving $C_r = 16$ channels. Block $j \in \{0, 1, 2, 3\}$ has support $[j/4, (j + 1)/4)$ and four channels, ordered as positive count, negative count, positive timestamp sum, and negative timestamp sum. Table 8 specifies all channels through their block indices. Every channel uses window-time partitioning and sum aggregation.

Table 8: **Hyper Histogram channel specifications.** The four rows apply to each $j \in \{0, 1, 2, 3\}$, defining all 16 channels. Every channel uses sum aggregation and window-time partitioning.

c	Polarity	Measurement	$[s, e)$	ρ
$4j + 1$	Positive	Count	$[j/4, (j + 1)/4)$	0
$4j + 2$	Negative	Count	$[j/4, (j + 1)/4)$	0
$4j + 3$	Positive	Timestamp	$[j/4, (j + 1)/4)$	1
$4j + 4$	Negative	Timestamp	$[j/4, (j + 1)/4)$	1

Let $b_j = T_0 + j(T_1 - T_0)/4$, with $j = 0, \ldots, 4$, denote block boundaries. Within block j, timestamp channels accumulate the measurements

$$\widetilde{t}_{i,j} = \frac{t_i - b_j}{\max(b_{j+1} - b_j, 1)}, \tag{23}$$

while count channels accumulate one per event. Timestamp sums therefore differ from both event counts and latest-timestamp measurements.

For example, channel 7 sums positive timestamps in the second block. Its complete descriptor is

$$o_{\mathrm{HH},7} = \big[0, 1, 0 \mid 0, 1, 0 \mid 1, 0, 0, 0 \mid 1, 0 \mid 0.25, 0.5, 0.375, 0.25, 1\big]^\top. \tag{24}$$

Vertical bars separate field groups and add no dimensions. For a window $[0, 100)$, this channel covers $[25, 50)$. Two positive events at the same pixel with timestamps 30 and 40 contribute $(30 - 25)/25 + (40 - 25)/25 = 0.8$. Changing the events changes the pixel value but leaves the descriptor unchanged.

B.4 Unseen Representations

Time Surface and SBT-5 use existing categorical fields and new combinations of those fields and support intervals. Neither requires a new representation-ID embedding or an expanded descriptor vocabulary. Their definitions below specify the variants used in our experiments.

Time Surface. We use a two-channel, polarity-separated Time Surface. Let $\tau_i = (t_i - T_0)/(T_1 - T_0)$ for events in $[T_0, T_1)$. For polarity p, the corresponding channel is

$$X_p^{\mathrm{TS}}(x, y) = \max(\{\tau_i : (x_i, y_i) = (x, y),\ p_i = p\} \cup \{0\})\,. \tag{25}$$

Channels are ordered negative then positive. Each sample is constructed independently, without carrying a time surface across windows or applying exponential decay. Both channels use timestamp measurements, maximum aggregation, window-time partitioning, support $[0, 1)$, and $\rho = 1$. Their complete descriptors are

$$\begin{aligned} o_{\mathrm{TS},1} &= \big[1, 0, 0 \mid 0, 1, 0 \mid 0, 0, 1, 0 \mid 1, 0 \mid 0, 1, 0.5, 1, 1\big]^\top, \\ o_{\mathrm{TS},2} &= \big[0, 1, 0 \mid 0, 1, 0 \mid 0, 0, 1, 0 \mid 1, 0 \mid 0, 1, 0.5, 1, 1\big]^\top. \end{aligned} \tag{26}$$

Although Time Surface and 2D Histogram both have two channels, their measurement, aggregation, and relative-timestamp fields differ.

SBT-5. We divide the sample window into five nonoverlapping, equal-duration hard bins, ordered from earliest to latest. Each channel accumulates signed polarities from both event polarities:

$$X_{j+1}^{\mathrm{SBT}}(x, y) = \sum_{i:(x_i, y_i) = (x, y)} p_i\, \mathbf{1}\left[\frac{j}{5} \le \tau_i < \frac{j+1}{5}\right], \qquad j \in \{0, \ldots, 4\}. \tag{27}$$

We use no interpolation between bins or count clipping. Bin assignment uses integer timestamps to compute $j_i = \lfloor 5(t_i - T_0)/(T_1 - T_0) \rfloor$; events at T_1 are excluded. All channels share $(p, m, a, q, \rho) = (\mathrm{both}, \mathrm{polarity}, \mathrm{sum}, \mathrm{window_time}, 0)$, with complete descriptors

$$o_{\mathrm{SBT},j+1} = \left[0, 0, 1 \mid 0, 0, 1 \mid 1, 0, 0, 0 \mid 1, 0 \mid \frac{j}{5}, \frac{j+1}{5}, \frac{2j+1}{10}, \frac{1}{5}, 0\right]^\top. \tag{28}$$

For example, the third channel has support $[0.4, 0.6)$ and descriptor $[0, 0, 1 \mid 0, 0, 1 \mid 1, 0, 0, 0 \mid 1, 0 \mid 0.4, 0.6, 0.5, 0.2, 0]^\top$.

B.5 Conditioning the CRA Generators

The same O_r is processed by two embedding networks with independent parameters. Each network shares its parameters across all channels and representations. The patch branch uses $\phi : \mathbb{R}^{17} \to \mathbb{R}^{128} \to \mathbb{R}^{128}$, with GELU after each linear layer, to generate the channel weights and patch-kernel residual coefficients. Weight generation does not depend on the current input sample.

The low-rank branch uses $\psi : \mathbb{R}^{17} \to \mathbb{R}^{128}$, consisting of a linear layer followed by GELU. Its representation-level condition is

$$u_r = \mathrm{concat}\left(\mathrm{mean}_c\, \psi(o_{r,c}), \max_c \psi(o_{r,c}), \frac{\log(1 + C_r)}{\log 33}\right) \in \mathbb{R}^{257}, \tag{29}$$

where the maximum is elementwise. A shared MLP maps u_r to the mixing coefficients for the first n Transformer blocks. The denominator $\log 33$ is a fixed scaling constant, not a constraint limiting inputs to 32 channels.

Per-channel normalization statistics (μ_r, σ_r) are separate metadata used to standardize X_r; they are not included in O_r or passed to the weight generators. An unseen representation is supplied through (O_r, μ_r, σ_r), with statistics estimated from its designated training split without task labels or test samples. For a fixed representation and trained CRA, generated weights can be materialized once.

The construction is equivariant to channel reordering. If the input channels, descriptor rows, and normalization statistics are permuted together, the generated patch kernel undergoes the same channel permutation, while mean and maximum pooling leave u_r unchanged. Thus, with fixed parameters at inference, the encoder output is unchanged up to numerical roundoff. This follows from the shared row-wise mappings and symmetric pooling. Renaming a representation while keeping its descriptors fixed also leaves its generated weights unchanged.

B.6 Reference Encoding

The following reference implementation encodes the descriptor fields in the order specified in Table 6. Applying it to the channel specifications above and stacking the outputs in input-channel order produces all 37 descriptor rows for the five non-HTA representations. HTA follows the original encoder path without descriptor conditioning.

```
import math
import numpy as np

VOCAB = (
    ("negative", "positive", "both"),
    ("count", "timestamp", "polarity"),
    ("sum", "mean", "max", "variance"),
    ("window_time", "event_index"),
)

def encode_channel(p, m, a, q, support, relative=False):
    s, e = map(float, support)
    assert math.isfinite(s) and math.isfinite(e)
    assert 0.0 <= s < e <= 1.0
    values = []
    for field, choices in zip((p, m, a, q), VOCAB):
        field = str(field).strip().lower()
        assert field in choices
        values.extend(float(field == v) for v in choices)
    values.extend((s, e, (s + e) / 2, e - s,
                   float(bool(relative))))
    return np.asarray(values, dtype=np.float32)

# Third SBT-5 channel; no representation ID is supplied.
example = encode_channel(
    "both", "polarity", "sum", "window_time", (0.4, 0.6)
)
```

C Training and Evaluation Details

C.1 Pretraining Data and Input Processing

Model Configuration. Hyper-RED (Wang et al., 2026b) and OmniE^2 start from the same HTA-pretrained ViT-B/16 checkpoint (Dosovitskiy, 2020). For non-HTA inputs with C_r channels, the

baseline sums the original patch kernel over its three input channels, divides by C_r, and replicates the result across target channels. HTA retains its original patch projection.

Our event pretraining and downstream evaluation setup follows prior work (Wang et al., 2026b; Chen et al., 2026). CRA pretraining uses 58,361 training sample entries across five dataset–task settings: N-Caltech101 (Orchard et al., 2015), DSEC-Semantic (Sun et al., 2022; Gehrig et al., 2021), DDD17 (Binas et al., 2017), MVSEC (Zhu et al., 2018), and DSEC-Depth (Gehrig et al., 2021). The corresponding splits are summarized in Table 9. CRA pretraining and supervised fine-tuning use the same training splits, and all training stages exclude test samples. We evaluate cross-representation retrieval on the full test sets without downstream fine-tuning.

Table 9: **Dataset splits.** Adapter pretraining and supervised fine-tuning use the same training samples. Counts denote event samples.

Task	Dataset	Train	Test
Classification	N-Caltech101	5,652	1,887
Segmentation	DSEC-Semantic	8,058	2,800
	DDD17	15,950	3,890
Depth estimation	MVSEC	9,484	21,125
	DSEC-Depth	19,217	7,167

Event windows and pairing. Event windows span 50 ms, ending at the reference timestamp for DSEC and DDD17 segmentation and at the depth timestamp for DSEC depth and MVSEC. N-Caltech101 uses complete event sequences.

Input preprocessing. Non-HTA inputs are native multichannel floating-point tensors, standardized using per-representation, per-channel statistics from the training split. HTA retains its original preprocessing. Pretraining uses 640×480 inputs without random spatial augmentation.

Dataset splits. For direct comparability with prior work, we retain the dataset partitions used in the established evaluation protocols (Wang et al., 2026b; Chen et al., 2026), without reassigning samples across splits. For each dataset–task pair, Hyper-RED and OmniE^2 use identical training and test samples. CRA pretraining uses only samples from the corresponding training partitions, without downstream annotations. Supervised fine-tuning uses these training partitions with task annotations. The split sizes are summarized in Table 9.

Split separation. CRA optimization uses only training samples, with source-event identities kept disjoint across pretraining splits. Test samples are excluded from CRA optimization and checkpoint selection. Neither stage uses downstream annotations.

C.2 CRA Pretraining

Each update pairs HTA with one non-HTA representation, cycling through the three targets in a fixed order. The original encoder is frozen, HTA reference features are detached, and only CRA is optimized. We use AdamW (Loshchilov & Hutter, 2017) with the settings in Table 11, excluding biases and one-dimensional parameters from weight decay. Training uses FP16 mixed precision on NVIDIA RTX 3090.

Table 10: **Adapter configurations.** Low-rank components are specified per adapted projection. Parameters refer to adapter pretraining.

Adapter	Patch projection	Low-rank update	Params. (M)
No Adapter	Fixed mean inflation	None	0.00
Basis+LoRA	8 shared bases	Shared rank 32	3.15
Patch+LoRA	Separate full kernels	Shared rank 32	7.47
ID-CRA	8 shared bases	ID-conditioned 4×8	3.20
CRA (Ours)	8 shared bases	Operator-conditioned 4×8	3.20

Table 11: **CRA pretraining hyperparameters.** Learning rates are peak values; warm-up is linear.

Optimization		Training and alignment	
Optimizer	AdamW	Total updates	60,000
Adam (β_1, β_2)	$(0.9, 0.98)$	Updates per target	20,000
Adam ϵ	10^{-6}	Global batch size	64
Patch learning rate	10^{-4}	Input resolution	640×480
Low-rank learning rate	2×10^{-5}	Loss weights (λ_g, λ_d)	$(0.5, 1.5)$
Weight decay	0.05	Temperature τ	0.07
Warm-up updates	6,000	Dense positions per sample	64
Learning-rate decay	Cosine	Validation interval	500 updates
Gradient norm clip	1.0	Random seed	42

C.3 Adapter Comparison Configurations

We compare CRA with three architectural controls and a no-adapter baseline to examine the roles of patch parameterization and representation conditioning. All variants start from the same HTA-pretrained ViT-B/16 checkpoint and retain the native input channels: 2 for 2D Histogram, 12 for ERGO-12, and 16 for Hyper Histogram. During adapter pretraining, the original encoder parameters remain frozen. All learned variants adapt the patch projection and the first four Transformer blocks, targeting the joint query/key/value projection, attention output projection, and both feed-forward projections. HTA bypasses all adapters and retains the original encoder.

No Adapter. This is the Hyper-RED baseline described above. Non-HTA inputs use the fixed mean-inflated patch kernel, without learned patch corrections or low-rank updates. This variant requires no additional adapter pretraining; its encoder and task head are still optimized during supervised downstream fine-tuning.

Basis+LoRA. This control retains CRA's eight shared patch-kernel bases, but replaces descriptor-conditioned coefficient generation with directly learned parameters for each representation. Channel weights are obtained by applying softmax to learned channel logits, while basis coefficients use the same $\tanh(\cdot)/C_r$ parameterization as CRA. Transformer adaptation uses one rank-32 LoRA update per projection, shared across the three non-HTA representations and independent of their descriptors. Factors remain separate across blocks and projections. Rank 32 matches the factor parameter budget of CRA's four rank-8 components, yielding an approximately parameter-matched control without descriptor-conditioned weight generation.

Patch+LoRA. This variant replaces the basis-based patch projection with a separately learned full kernel $W^{(r)} \in \mathbb{R}^{768 \times C_r \times 16 \times 16}$ for each non-HTA representation. Each kernel is initialized from the mean-inflated source projection, while the original patch bias remains frozen. Transformer adaptation is identical to Basis+LoRA, using shared rank-32 updates. This is a higher-capacity control rather than a parameter-matched alternative.

ID-CRA. This control preserves CRA's weight generators, eight patch bases, and four rank-8 components per adapted projection, but replaces operator semantics with learned representation identities and channel-layout information. Specifically, each representation has a learned embedding $e_r \in \mathbb{R}^{12}$, and its channel condition is

$$\begin{aligned} \widetilde{o}_{r,c} &= \left[e_r;\, u_c;\, u_c^2;\, \sin(2\pi u_c);\, \cos(2\pi u_c);\, \frac{\log(1+C_r)}{\log 33} \right] \in \mathbb{R}^{17}, \\ u_c &= \frac{c - \frac{1}{2}}{C_r}, \qquad c = 1, \ldots, C_r. \end{aligned} \tag{30}$$

Thus, the generators receive representation identity, channel position, and channel count, but no explicit polarity, measurement, or aggregation semantics. The three learned identity embeddings introduce only 36 additional parameters.

CRA (Ours). CRA uses the operator descriptors defined in Section 4.1, with eight shared patch bases and four rank-8 components per adapted projection. ID-CRA and CRA use the same generator architectures and hidden dimension of 128. Their block-wise mixing coefficients are shared

across projections within each block, while low-rank factors remain projection-specific. Unlike ID-CRA, CRA conditions adaptation on representation construction semantics rather than learned representation identities.

Initialization and parameter accounting. All learned variants start from the mean-inflated patch projection with zero Transformer corrections. For basis-based projections, channel weights are initially uniform and patch bases are zero-initialized. Low-rank output factors are initialized to zero. Both the shared rank-32 updates and conditional rank-8 mixtures use an overall scaling factor of $1/8$. Table 10 counts all trainable adapter parameters jointly across the three non-HTA representations, excluding the frozen source encoder and downstream heads.

Comparison protocol. The three new controls use 60,000 pretraining updates, a global batch size of 64. The loss weights are $(\lambda_g, \lambda_d) = (0.5, 1.5)$. Patch parameters use a peak learning rate of 10^{-4}; low-rank parameters and the ID embeddings use 2×10^{-5}. The remaining optimizer settings follow Table 11.

Completed No Adapter and CRA runs are reused. The new controls use their final pretraining checkpoints, whereas the reused CRA results retain their original checkpoint-selection protocol. HTA results are reused without additional adapter training. Reported averages assign equal weight to HTA, 2D Histogram, ERGO-12, and Hyper Histogram.

C.4 Downstream Evaluation

For each target representation, CRA weights are generated once and merged into the encoder before supervised fine-tuning. We then jointly optimize the full encoder and task head; backbone freezing applies only to CRA pretraining. Inference requires only the target representation and does not execute the weight generators. Hyper-RED and OmniE^2 use matching dataset partitions, task heads, training budgets, and optimization settings for each dataset–representation pair. Table 12 summarizes the training settings and checkpoint-selection criteria. Training stops after 30 consecutive evaluations without improvement in the corresponding selection metric. For each model and dataset–representation pair, all reported metrics are computed using the same selected checkpoint.

Table 12: **Supervised fine-tuning hyperparameters.** Batch sizes are effective values; epochs are maximum budgets. All tasks use AdamW and stepwise learning-rate decay.

Hyperparameter	Classification	Segmentation		Depth estimation	
	N-Caltech101	DSEC	DDD17	MVSEC	DSEC
Input resolution	256×256	640×440	704×416	352×272	640×480
Batch size	40	12	32	40	12
Maximum epochs	80	80	80	80	80
Encoder learning rate	2×10^{-5}	4×10^{-6}	2×10^{-6}	2×10^{-6}	10^{-6}
Task-head learning rate	1.5×10^{-4}	1.2×10^{-5}	5×10^{-6}	4×10^{-6}	2×10^{-6}
Weight decay	8.75×10^{-5}	5×10^{-5}	10^{-4}	10^{-4}	10^{-4}
LR decay factor	0.95	0.97	0.9	0.9	0.9
Decay interval (epochs)	5	10	5	5	5
Gradient norm clip	1.0	0.1	0.1	0.1	0.1
Checkpoint criterion	Acc@1 ↑	mIoU ↑	mIoU ↑	RMSE ↓	RMSE ↓

Object classification. N-Caltech101 classification excludes `background_google`, retaining 100 classes. An MLP with hidden dimension 768 maps the final `[CLS]` feature to class logits. We train on the designated training partition using cross-entropy loss and BF16 precision, without random augmentation. Acc@1 and Acc@5 are evaluated on the 1,887 test samples.

Semantic segmentation. DSEC and DDD17 use segmentation heads based on EoMT (Kerssies et al., 2025), with 11 and 6 classes, respectively. Decoder initialization is matched between Hyper-RED and OmniE^2 within each dataset. The mask, Dice, and classification losses have weights of 5, 5, and 2, respectively, with a no-object class weight of 0.1.

Depth estimation. MVSEC and DSEC use the metric-depth decoder of Depth Anything V2 (Yang et al., 2024a), initialized with its Virtual KITTI 2 (Cabon et al., 2020) pretrained weights. We optimize the logarithmic depth loss of Eigen et al. (2014), with a mean-term coefficient of 0.5. Evaluation uses valid pixels with depths within 0.1–30 m on MVSEC and 0.1–80 m on DSEC. We report δ_1, δ_2, AbsRel, RMSE, and $\text{RMSE}_{\log}$. The threshold accuracies use thresholds of 1.25 and 1.25^2, respectively, and $\text{RMSE}_{\log}$ uses natural logarithms. Metrics are computed per image and then averaged with equal image weights, without median scaling.

C.5 Cross-Representation Evaluation

Cross-representation retrieval. We evaluate retrieval on the full test sets of DSEC, DDD17, MVSEC, and N-Caltech101 without downstream fine-tuning. We use the final `[CLS]` feature (TOK) and globally averaged patch features (GAP), with ℓ_2 normalization and cosine similarity. Each query has one positive: the same event sample encoded in the target representation. All other gallery samples are negatives.

The four representations yield 12 directed retrieval pairs. R@1, R@5, mAP, and MedR are computed for each pair and averaged with equal weights. With one positive per query, mAP equals mean reciprocal rank. Reported MedR is the mean of the 12 direction-specific median ranks.

For unseen-representation transfer, we generate target weights through zero-shot initialization without target-specific adapter training, followed by supervised fine-tuning on DDD17 and N-Caltech101. OmniE^2 and the mean-inflated Hyper-RED baseline use matching downstream settings in both settings below.

Registered targets. We hold out 2D Histogram, ERGO-12, or Hyper Histogram from CRA optimization and checkpoint selection. HTA is paired with the other two non-HTA representations for 40,000 updates, preserving 20,000 updates per observed target. The held-out target's operator descriptors condition weight generation, while its training-split statistics are used for input normalization. Its encoded inputs do not participate in CRA optimization or checkpoint selection.

Novel targets. Time Surface (Gallego et al., 2020) ($C = 2$) and SBT-5 (Wang et al., 2019) ($C = 5$) are absent from both shared CRA pretraining and the original registry of representation operators. Their operator descriptors are supplied to the trained generators to initialize the target encoders before downstream fine-tuning.

D More Detailed Results

We complement the averaged ablations in Sec. 5.4 with per-representation results, frozen-encoder evaluations, and qualitative comparisons. The adaptation-depth study covers classification, depth estimation, and semantic segmentation, while the loss-weight study focuses on the two dense prediction tasks. Ablation averages assign equal weight to HTA, 2D Histogram, ERGO-12, and Hyper Histogram.

D.1 Frozen-Encoder Evaluation

We conduct two complementary evaluations on all 1,887 N-Caltech101 test samples, using CRA with $n = 4$ and $(\lambda_{\text{d}}, \lambda_{\text{g}}) = (1.5, 0.5)$. The pretrained encoder and CRA remain frozen. A single linear classification head is trained on HTA features and then fixed for evaluation across all representations.

Transfer with a Shared Classification Head. We compare direct transfer from Hyper-RED with CRA adaptation, using the same HTA-trained head for both. Table 13 shows that CRA improves Acc@1 for all three target representations, with gains ranging from 12.03 to 40.12 percentage points. Although target accuracies remain below the HTA reference, these improvements demonstrate that CRA enables more effective reuse of the pretrained features with a fixed classifier.

Table 13: **Cross-representation classification with a shared frozen head.** Acc@1 (%) on N-Caltech101 using one HTA-trained linear classifier. The HTA reference achieves 95.81%. Gains are in percentage points and computed before rounding.

Target representation	Direct transfer	CRA	Gain
2D Histogram	50.45	**90.57**	$+40.12$
ERGO-12	82.35	**94.38**	$+12.03$
Hyper Histogram	64.02	**93.16**	$+29.15$

Table 14: **Ablation on adaptation depth across event representations.** n: initial Transformer blocks with CRA low-rank updates; †: default setting. Cls.: N-Caltech101 Acc@1; Depth: MVSEC δ_1; Seg.: DDD17 mIoU (all in %). Best results within each representation are bold, including ties at the displayed precision.

(a) HTA

n	Cls. ↑	Depth ↑	Seg. ↑
0	**97.62**	80.15	64.42
2	97.56	**80.18**	64.81
4†	97.46	80.15	**64.86**
6	97.56	80.15	64.73
8	**97.62**	**80.18**	64.53

(b) 2D Histogram

n	Cls. ↑	Depth ↑	Seg. ↑
0	95.50	77.17	60.17
2	96.13	77.01	60.10
4†	95.76	78.86	**60.59**
6	95.81	78.54	59.98
8	**96.29**	**79.01**	60.45

(c) ERGO-12

n	Cls. ↑	Depth ↑	Seg. ↑
0	96.61	79.32	59.96
2	**97.14**	78.20	59.59
4†	96.66	**79.34**	**60.97**
6	96.98	77.61	60.32
8	97.03	78.52	60.31

(d) Hyper Histogram

n	Cls. ↑	Depth ↑	Seg. ↑
0	96.66	77.16	60.13
2	**96.98**	77.09	59.98
4†	96.61	**79.10**	**61.24**
6	**96.98**	78.02	60.53
8	96.71	78.17	60.87

D.2 Number of Adapted Layers

Table 14 shows that the preferred adaptation depth depends on both the representation and the task. With $n = 4$, ERGO-12 and Hyper Histogram achieve their highest depth and segmentation scores, while 2D Histogram achieves its highest segmentation score. For 2D Histogram, depth performance instead peaks at $n = 8$, indicating that additional adapted blocks can benefit specific representation–task combinations.

Classification exhibits a different pattern. Its best results occur at $n = 8$ for 2D Histogram, $n = 2$ for ERGO-12, and $n = 2$ or $n = 6$ for Hyper Histogram. Averaged over all four representations, $n = 4$ yields the highest depth and segmentation scores, although it does not maximize every individual result. Increasing the adaptation depth to six or eight blocks adds parameters without improving these average dense prediction scores, supporting $n = 4$ as our default setting.

D.3 Alignment Loss Weights

Table 15 reports the joint variation of λ_{d} and λ_{g} with adaptation depth fixed at $n = 4$. The default setting $(\lambda_{\mathrm{d}}, \lambda_{\mathrm{g}}) = (1.5, 0.5)$ achieves the highest mean depth and segmentation scores across the four representations. It also yields the best depth score for 2D Histogram, both metrics for ERGO-12, and the best segmentation score for Hyper Histogram.

The optimal weights nevertheless differ across individual representation–task combinations. For example, 2D Histogram achieves its highest segmentation mIoU at $(0.5, 0.5)$, whereas Hyper Histogram obtains its highest depth δ_1 at $(1.5, 1.5)$. Increasing either coefficient does not consistently improve performance. These results support selecting a shared configuration based on aggregate performance while recognizing that individual representations can favor different balances between global and dense alignment.

D.4 Adapter Design

Table 16 compares adapter designs on MVSEC for three non-HTA representations. Detailed configurations are provided in Appendix C.3.

Performance across representations. CRA improves both depth metrics over No Adapter across all three representations. For 2D Histogram, ERGO-12, and Hyper Histogram, it increases δ_1 by 1.22, 1.98, and 1.65 percentage points, respectively, while reducing RMSE by 0.111, 0.138, and

Table 15: **Ablation on loss weights across event representations.** (λ_d, λ_g): weights of the dense and global losses; the adaptation depth is fixed at $n = 4$; †: default setting. Depth: MVSEC δ_1; Seg.: DDD17 mIoU (both in %). Best results within each representation are bold, including ties at the displayed precision.

(a) HTA

(λ_d, λ_g)	Depth ↑	Seg. ↑
$(0.5, 0.5)$	80.17	64.68
$(0.5, 1.0)$	**80.18**	64.82
$(0.5, 1.5)$	80.17	64.31
$(1.0, 0.5)$	80.15	64.66
$(1.0, 1.0)$	80.16	64.77
$(1.0, 1.5)$	80.16	64.75
$(1.5, 0.5)^\dagger$	80.15	**64.86**
$(1.5, 1.0)$	**80.18**	64.34
$(1.5, 1.5)$	80.15	64.45

(b) 2D Histogram

(λ_d, λ_g)	Depth ↑	Seg. ↑
$(0.5, 0.5)$	78.01	**61.22**
$(0.5, 1.0)$	77.93	59.83
$(0.5, 1.5)$	77.50	60.49
$(1.0, 0.5)$	78.20	60.62
$(1.0, 1.0)$	78.57	60.79
$(1.0, 1.5)$	78.12	60.30
$(1.5, 0.5)^\dagger$	**78.86**	60.59
$(1.5, 1.0)$	75.78	59.80
$(1.5, 1.5)$	78.68	60.72

(c) ERGO-12

(λ_d, λ_g)	Depth ↑	Seg. ↑
$(0.5, 0.5)$	77.14	60.46
$(0.5, 1.0)$	79.20	60.32
$(0.5, 1.5)$	78.16	60.51
$(1.0, 0.5)$	79.24	60.16
$(1.0, 1.0)$	77.71	60.37
$(1.0, 1.5)$	77.88	59.34
$(1.5, 0.5)^\dagger$	**79.34**	**60.97**
$(1.5, 1.0)$	78.33	59.44
$(1.5, 1.5)$	77.16	60.19

(d) Hyper Histogram

(λ_d, λ_g)	Depth ↑	Seg. ↑
$(0.5, 0.5)$	77.32	59.75
$(0.5, 1.0)$	78.06	60.19
$(0.5, 1.5)$	76.71	60.68
$(1.0, 0.5)$	78.27	60.17
$(1.0, 1.0)$	79.52	60.38
$(1.0, 1.5)$	77.06	61.15
$(1.5, 0.5)^\dagger$	79.10	**61.24**
$(1.5, 1.0)$	78.96	60.19
$(1.5, 1.5)$	**79.57**	61.01

Table 16: **Ablation on adapter design across non-HTA representations.** We report MVSEC δ_1 (%) and RMSE. Params.: total trainable adapter parameters (M). Best δ_1 and RMSE within each representation are bold, including ties at the displayed precision.

(a) 2D Histogram

Adapter	δ_1 ↑	RMSE ↓	Params.
No Adapter	77.64	3.616	0.00
Basis+LoRA	77.29	3.601	3.15
Patch+LoRA	78.52	3.561	7.47
ID-CRA	78.07	3.545	3.20
CRA (Ours)	**78.86**	**3.505**	3.20

(b) ERGO-12

Adapter	δ_1 ↑	RMSE ↓	Params.
No Adapter	77.36	3.612	0.00
Basis+LoRA	78.71	3.538	3.15
Patch+LoRA	78.30	3.564	7.47
ID-CRA	**79.54**	**3.472**	3.20
CRA (Ours)	79.34	3.474	3.20

(c) Hyper Histogram

Adapter	δ_1 ↑	RMSE ↓	Params.
No Adapter	77.45	3.593	0.00
Basis+LoRA	77.97	3.608	3.15
Patch+LoRA	76.89	3.632	7.47
ID-CRA	78.43	3.620	3.20
CRA (Ours)	**79.10**	**3.493**	3.20

0.100. CRA achieves the best results on both metrics for 2D Histogram and Hyper Histogram, and outperforms both shared-LoRA baselines across all three representations.

Effect of operator conditioning. The comparison with ID-CRA examines the benefit of using representation construction rules at the same adapter parameter budget of 3.20M. On 2D Histogram, CRA improves δ_1 by 0.79 percentage points and reduces RMSE by 0.040. On Hyper Histogram, the corresponding improvements are 0.67 points and 0.127. However, ID-CRA performs slightly better on ERGO-12, achieving 79.54% δ_1 and 3.472 RMSE, compared with 79.34% and 3.474 for CRA. These results support the utility of operator conditioning while showing that its benefit varies across representations.

Parameter efficiency. CRA uses a similar number of trainable adapter parameters to Basis+LoRA (3.20M versus 3.15M) and fewer than half those of Patch+LoRA (7.47M). Despite its larger parameter budget, Patch+LoRA achieves lower δ_1 and higher RMSE than CRA on all three representations. These comparisons suggest that parameter count alone does not explain the observed gains and support CRA's favorable performance–parameter trade-off.

D.5 Qualitative Results

We provide qualitative comparisons between Hyper-RED and OmniE^2 on DSEC for semantic segmentation and depth estimation. Each figure presents RGB references, ground truth, and predictions using HTA, 2D Histogram, ERGO-12, and Hyper Histogram.

Semantic Segmentation. Figs. 6 and 7 compare the predicted semantic masks. In the first example, OmniE^2 recovers more complete sidewalk regions with 2D Histogram and Hyper Histogram, yielding closer visual agreement with the ground truth. The displayed examples also illustrate more consistent road–sidewalk separation across representations.

Depth Estimation. Figs. 8 and 9 present depth predictions in daytime and low-light scenes. OmniE^2 exhibits more consistent depth patterns across representations, particularly in building regions. For example, the 2D Histogram prediction in the final scene shows closer visual agreement with the ground truth around the building on the left.

E LIMITATIONS

Evaluation scope. Our experiments use an HTA-pretrained Hyper-RED ViT-B/16 and a limited set of event representations. Although CRA accommodates variable channel counts, adapting a new representation requires its construction rules to be expressed through the current operator descriptor schema. Generalization to other backbone architectures and substantially different representation families remains to be verified.

Theoretical scope. Our analysis bounds the deviation from reference risk for a shared predictor under a Lipschitz task-loss assumption. It does not guarantee improvements in downstream metrics after end-to-end fine-tuning. Furthermore, the analysis does not establish that optimizing the normalized InfoNCE objective reduces the unnormalized feature discrepancy used in the bound.

Complementarity across representations. Our framework adapts each event representation individually and does not explicitly exploit complementary information across representations. Future work could explore adaptive fusion or selection of representations to combine their distinct temporal and spatial cues for downstream tasks.

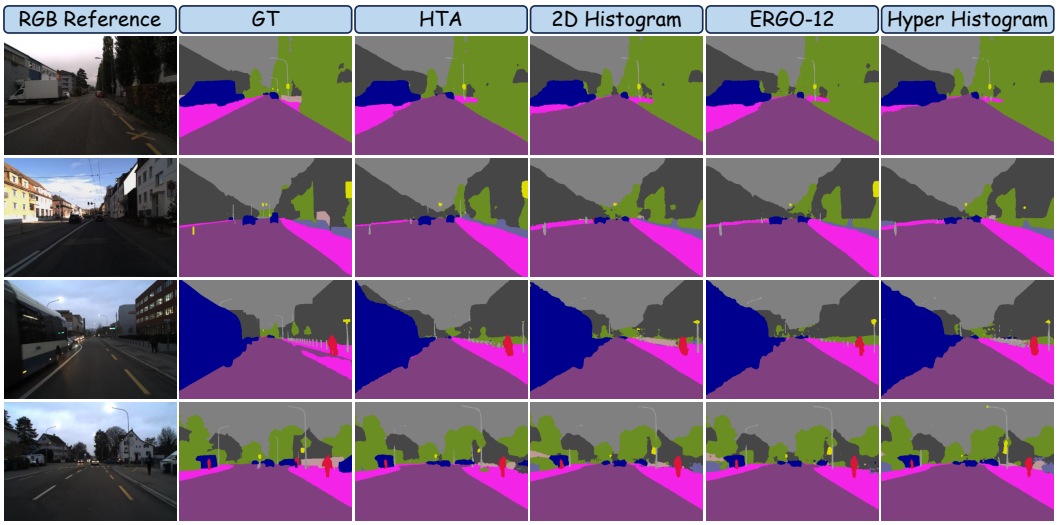


Figure 6: **Qualitative semantic segmentation results of Hyper-RED on DSEC.** Each row shows a different scene. From left to right: RGB reference, ground-truth semantic labels, and predictions using HTA, 2D Histogram, ERGO-12, and Hyper Histogram.

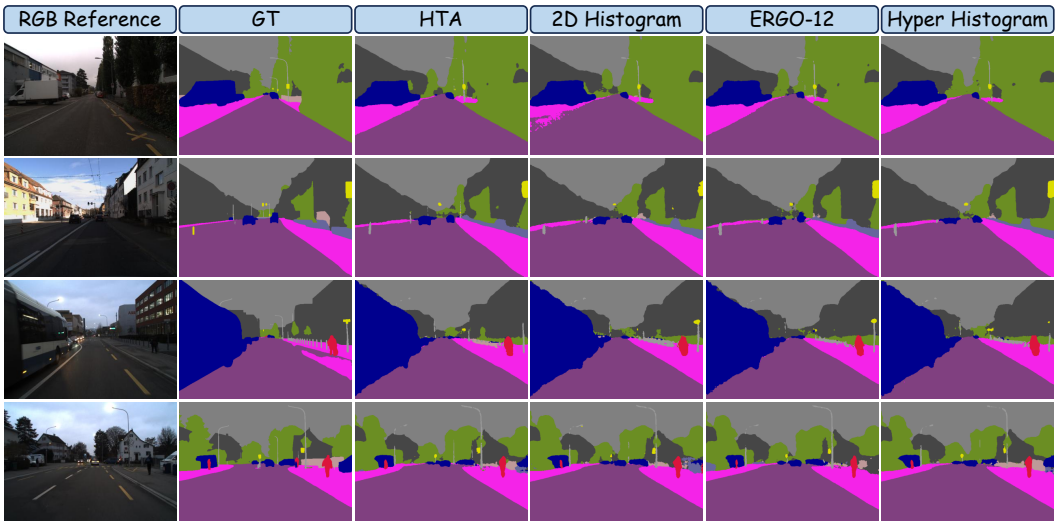


Figure 7: **Qualitative semantic segmentation results of OmniE^2 on DSEC.** Each row shows a different scene. From left to right: RGB reference, ground-truth semantic labels, and predictions using HTA, 2D Histogram, ERGO-12, and Hyper Histogram.

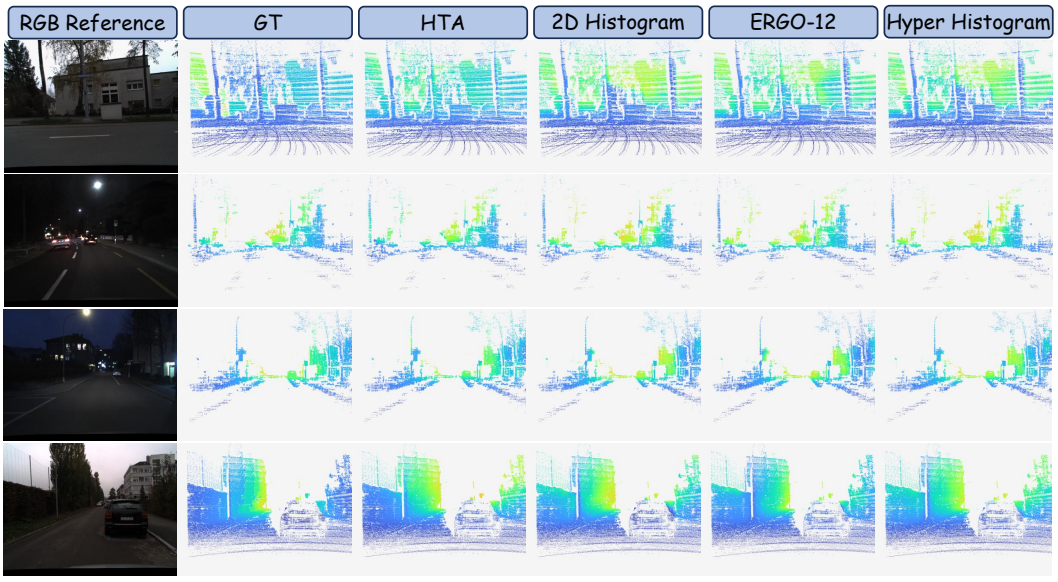


Figure 8: **Qualitative depth estimation results of Hyper-RED on DSEC.** Each row shows a different scene. From left to right: RGB reference, ground-truth depth, and predictions using HTA, 2D Histogram, ERGO-12, and Hyper Histogram.

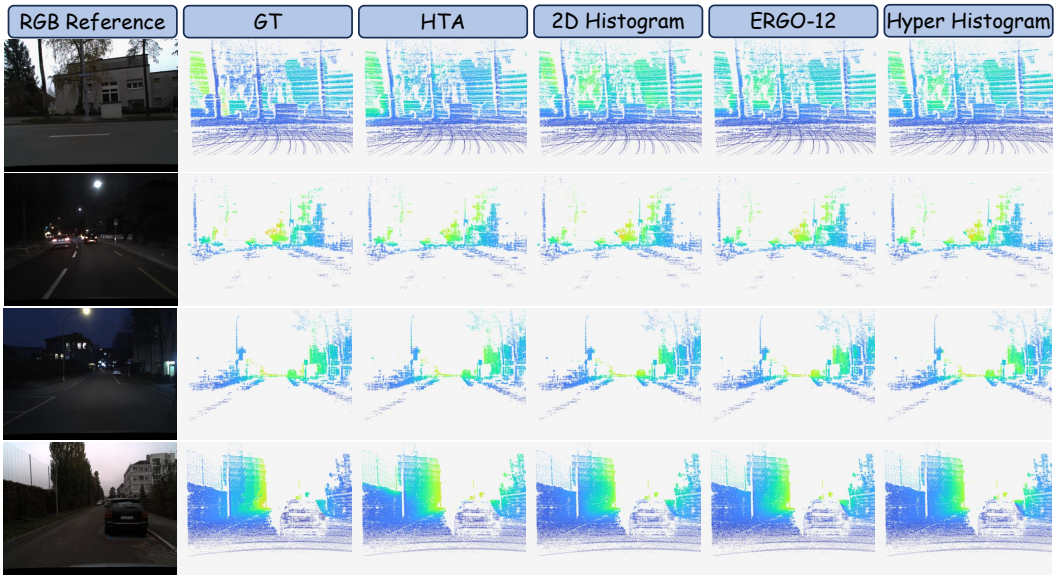


Figure 9: **Qualitative depth estimation results of OmniE^2 on DSEC.** Each row shows a different scene. From left to right: RGB reference, ground-truth depth, and predictions using HTA, 2D Histogram, ERGO-12, and Hyper Histogram.