Supplementary: Generative Event Pretraining with Foundation Model Alignment

Jianwen Cao Jiaxu Xing Nico Messikommer Davide Scaramuzza

Robotics and Perception Group, University of Zurich

1. Overview

This supplementary document provides additional implementation details and qualitative results that complement the main paper "Generative Event Pretraining with Foundation Model Alignment". Specifically, Sec. 2 presents extended method details, including training hyperparameters and task-specific settings for semantic segmentation and depth estimation, while Sec. 3 reports additional qualitative results for depth prediction and attention visualization.

2. Extended Method Details

2.1. Alignment hyperparameters

During the alignment stage, we fine-tune the pretrained backbone with a lightweight optimization setup. We use the AdamW [4] optimizer with a batch size of 32 and an input resolution of 224×224. The base learning rate is set to 3×10^{-5} without weight decay, and we employ a linear warmup for the first 100 steps followed by a cosine decay schedule down to a minimum learning rate of 0. Training is performed for 2.5×10^5 steps.

To improve robustness, we apply moderate data augmentation, including polarity swap and horizontal flipping, each with probability 0.5. In addition, we use random resized cropping with scale range $[0.25, 1.0]$ and a random upscale operation (scale factor 2) with probability 0.1. All alignment hyperparameters are summarized in Table 1.

Table 1. Hyperparameters for the alignment stage.

Setting	Value
optimizer	AdamW
Batch size	32
Learning rate	3×10^{-5}
Weight decay	0
Minimum learning rate	0
Warmup steps	100
Training steps	2.5×10^5
Input resolution	224×224
Learning rate schedule	linear warmup cosine decay
Data augmentation	
Polarity swap [9]	prob = 0.5
Horizontal flip	prob = 0.5
Random resized crop	scale = [0.25, 1.00]
Random upscale	prob = 0.1, scale = 2

2.2. Pretraining hyperparameters

For the pretraining stage, we adopt a similar optimization strategy but with settings tailored to longer context and larger-scale training. We use AdamW with a batch size of 8 and a context window size of 4096 (Following GPT2 [6]). The base learning rate is 5×10^{-4} with weight decay of 1×10^{-5}. We apply linear warmup for the first 100 steps and then use a cosine decay schedule towards an asymptote at 0, training for 2.5×10^5 steps in total. The full set of pretraining hyperparameters is listed in Table 2.

Table 2. Hyperparameters for the pretraining stage.

Setting	Value
optimizer	AdamW
Batch size	8
Window size	4096
Learning rate	5×10^{-4}
Weight decay	1×10^{-5}
Minimum learning rate	0
Warmup steps	100
Training steps	2.5×10^5
Learning rate schedule	linear warmup cosine decay

2.3. Semantic Segmentation Settings

For semantic segmentation, we adopt a lightweight decoding head that converts the sequence of visual tokens produced by the backbone into a full-resolution prediction map.

The design consists of two main components: a linear patch-wise decoding stage and a shallow convolutional refinement stage. We use cross-entropy loss and dice loss [8] with equal weights.

Linear Patch-wise Decoding. Vision Transformers (ViTs) operate on a grid of non-overlapping image patches. Given an input image with spatial resolution $H \times W$ and a patch size P, the encoder produces a sequence of

$$L = \frac{H}{P} \times \frac{W}{P} \tag{1}$$

tokens, each with feature dimension D. In contrast, dense prediction tasks provide supervision at the pixel level, i.e., one label per location in the original $H \times W$ grid.

To fully exploit this pixel-wise supervision, we adopt a simple *linear patch-wise decoder*. For each token, we apply a shared linear projection that maps the D-dimensional feature into CP^2 channels, where C is the number of classes. By reshaping and rearranging the outputs of all L tokens, we recover a dense prediction map of shape $C \times H \times W$.

Formally, let $F \in \mathbb{R}^{L \times D}$ denote the token features from the ViT encoder. Our decoder learns a weight matrix

$$W_{\text{dec}} \in \mathbb{R}^{D \times (CP^2)}, \tag{2}$$

and computes

$$\hat{Y} = \text{reshape}\big(FW_{\text{dec}}\big) \in \mathbb{R}^{C \times H \times W}, \tag{3}$$

where the reshape operation rearranges the per-token CP^2 outputs into their corresponding $P \times P$ spatial locations in the original image grid. This linear patch-wise decoding is computationally lightweight and allows us to leverage all pixel-level supervision without resorting to a heavy decoder.

Convolutional Refinement. To enhance spatial coherence and refine local details, the reconstructed feature map is further processed by a lightweight convolutional refinement module. This stage applies several 3×3 convolutions with nonlinear activations while preserving the original spatial resolution. The module introduces only 0.2% (ViT-Small) and 0.07% (ViT-Base) additional parameters, resulting in negligible impact on overall performance and computational cost, and is mainly employed to improve the visual quality of the predictions.

2.4. Depth Estimation Settings

Following prior work on monocular depth estimation [7, 10], we supervise training with the scale-invariant loss of [1] and the multi-scale scale-invariant gradient matching loss of [2], using fixed loss weights of 1 and 0.25, respectively. The network is optimized to predict normalized log-depth values during training, and these predictions are converted back to metric depth at inference time.

Decoder backbone. Given a batch of encoder tokens $T \in \mathbb{R}^{B \times L \times C}$, we treat B as the batch size, L as the number of tokens per image, and C as the channel dimension of each token. These tokens lie on a patch grid of size $H_p \times W_p$, obtained from an input image of spatial resolution $H \times W$ using a patch size p, so that $H_p = H/p$ and $W_p = W/p$.

We first apply a linear projection to each token and reshape the result into a low-resolution feature map $F_0 \in \mathbb{R}^{B \times C_0 \times H_p \times W_p}$, where C_0 is the number of feature channels used in the decoder. On top of F_0, we build a convolutional decoder with S stages, indexed by $s = 1, \ldots, S$. Each stage is implemented as a residual-like block, denoted DecBlock_s, that consists of two 3×3 convolutions with GELU activations and an identity skip connection. The input to stage s is F_{s-1} and the output is F_s.

Each decoder stage has a target spatial resolution (H_s, W_s). These resolutions gradually increase from the patch resolution (H_p, W_p) at the first stage up to the full image resolution (H, W) at the last stage. If the output F_s of stage s does not match (H_s, W_s), we use a bilinear interpolation operator, denoted $\text{Interp}(\cdot; H_s, W_s)$, to upsample it to the target size and obtain a feature map $\tilde{F}_s$ at resolution (H_s, W_s).

To aggregate information across scales, all stage outputs are first brought to full resolution (H, W) using the same interpolation operator and then concatenated along the channel dimension. This yields a fused feature map $F_{\text{fuse}} \in \mathbb{R}^{B \times C_{\text{fuse}} \times H \times W}$, where C_{fuse} is the total number of channels after concatenation.

Finally, a small convolutional prediction head is applied to F_{fuse}. This head consists of two 3×3 convolutions with GELU activations, followed by a 1×1 convolution and a sigmoid function. The output is a normalized log-depth map $Y_{\text{norm}} \in [0,1]^{B \times 1 \times H \times W}$, where the single channel corresponds to normalized log-depth for each pixel.

Log-depth parameterization. We convert the normalized log-depth Y_{norm} into metric depth using a fixed minimum and maximum depth range. Let $d_{\min} > 0$ and $d_{\max} > d_{\min}$ denote, respectively, the minimum and maximum metric depth considered during training. We define their logarithms $\ell_{\min} = \log d_{\min}$ and $\ell_{\max} = \log d_{\max}$, and the log-range $\Delta\ell = \ell_{\max} - \ell_{\min}$. For each pixel, the normalized log-depth prediction Y_{norm} is first mapped to the corresponding log-depth value $\ell = Y_{\text{norm}}\,\Delta\ell + \ell_{\min}$, and the final metric depth prediction is obtained as

$$D = \exp(\ell) \in \mathbb{R}^{B \times 1 \times H \times W} \tag{4}$$

Thus, D contains the predicted depth (in meters) at each pixel. This log-depth parameterization improves numerical stability over large depth ranges and makes the predictions more robust to global scale shifts, since uniform shifts

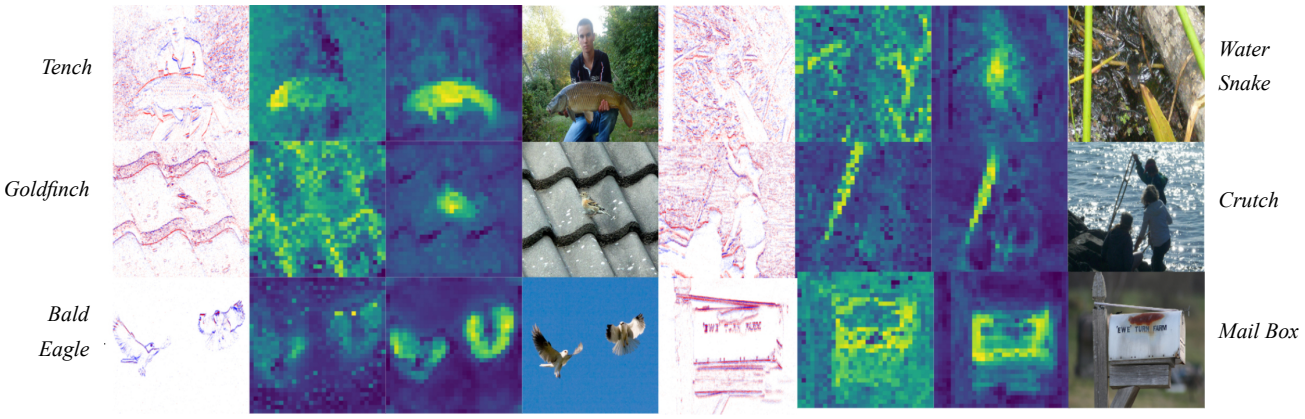


Figure 1. Visual comparison of attention maps on N-ImageNet [3] validation event inputs. We compare the attention responses of DINOv2 [5] (b, f) and our method (c, g) given input events (a, e), alongside reference RGB images (d, h). Our approach demonstrates more precise localization of semantic structures compared to the diffuse activations of DINOv2.

in log-depth correspond to multiplicative changes in metric depth.

Multi-scale log-depth gradient loss. To further regularize the spatial structure of the prediction, we match log-depth gradients at multiple resolutions. We consider a set of scales $\mathcal{S}$ (e.g., different downsampling factors). For each scale $s \in \mathcal{S}$, we downsample the predicted depth, the ground-truth depth, and the validity mask with a fixed operator $\mathrm{Down}_s(\cdot)$:

$$D^{(s)} = \mathrm{Down}_s(D), \tag{5}$$

$$(D^\star)^{(s)} = \mathrm{Down}_s(D^\star), \tag{6}$$

$$M^{(s)} = \mathbf{1}\{\mathrm{Down}_s(M) > 0.5\}. \tag{7}$$

Training loss. Supervision is provided by LiDAR depth measurements $D^\star \in \mathbb{R}^{B\times 1\times H\times W}$ and a binary validity mask $M \in \{0,1\}^{B\times 1\times H\times W}$, where $M_i = 1$ indicates that a valid ground-truth depth value is available at pixel i and $M_i = 0$ otherwise. All losses are computed only on pixels with $M_i = 1$.

To concisely express masked averages, we define a masked mean operator. Given any quantity X defined per pixel (and per batch element), and a corresponding binary mask M, we write

$$\langle X \rangle_M = \frac{\sum_i M_i X_i}{\sum_i M_i + \varepsilon}, \tag{8}$$

where i indexes all spatial positions and batch elements, and ε is a small constant used to avoid division by zero when no valid pixels are present.

Scale-invariant log RMSE. Following Eigen et al [1]., we measure the discrepancy between predicted and ground-truth depths in log space. For each valid pixel i, we define the log-depth residual

$$y_i = \log D_i - \log D_i^\star,$$

where D_i and $D_i^\star$ denote the predicted and ground-truth depth, respectively. Using the masked mean operator in Eq. 8, the scale-invariant log-depth loss is

$$L_{\mathrm{silog}} = \sqrt{\langle y_i^2 \rangle_M - \lambda \langle y_i \rangle_M^2}. \tag{9}$$

Here, $\lambda \in [0,1]$ is a scalar hyperparameter that controls the degree of scale invariance; in our experiments we use $\lambda = 0.85$. The first term encourages small squared log-errors, while the second term subtracts a global bias term, making the loss invariant (up to λ) to additive shifts in log-depth, i.e., multiplicative changes in depth scale.

At scale s, we compute horizontal and vertical log-depth gradient residuals using finite differences:

$$G_x^{(s)} = \nabla_x \log D^{(s)} - \nabla_x \log (D^\star)^{(s)}, \tag{10}$$

$$G_y^{(s)} = \nabla_y \log D^{(s)} - \nabla_y \log (D^\star)^{(s)}. \tag{11}$$

The corresponding masked mean absolute errors are

$$E_x^{(s)} = \langle |G_x^{(s)}| \rangle_{M^{(s)}}, \tag{12}$$

$$E_y^{(s)} = \langle |G_y^{(s)}| \rangle_{M^{(s)}}, \tag{13}$$

where $\langle \cdot \rangle_{M^{(s)}}$ denotes the masked mean defined analogously to Eq. (8), but using the mask $M^{(s)}$.

The gradient loss at scale s and the final multi-scale gradient loss are

$$L_{\mathrm{grad}}^{(s)} = E_x^{(s)} + E_y^{(s)}, \tag{14}$$

$$L_{\mathrm{ms\_grad}} = \frac{1}{|\mathcal{S}|} \sum_{s\in\mathcal{S}} L_{\mathrm{grad}}^{(s)}, \tag{15}$$

where $|\mathcal{S}|$ is the number of scales in $\mathcal{S}$.

Final objective. The total depth training objective is a weighted sum of the scale-invariant log RMSE and the multi-scale gradient loss:

$$L_{\text{total}} = w_{\text{silog}}\, L_{\text{silog}} + w_{\text{ms\_grad}}\, L_{\text{ms\_grad}}, \tag{16}$$

where w_{silog} and $w_{\text{ms\_grad}}$ are non-negative scalar hyperparameters that balance the two terms. Gradients are back-propagated only through pixels with $M_i = 1$ (and their downsampled counterparts), so invalid or missing depth values do not influence learning.

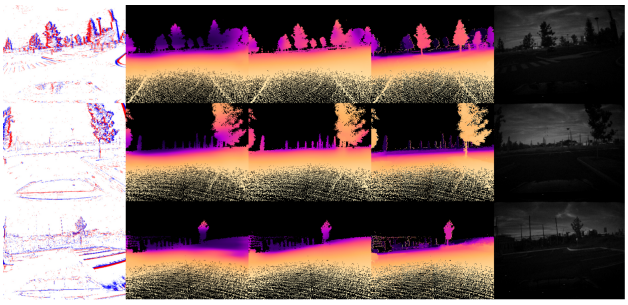
3. More Qualitative Results

3.1. Depth Estimation

Figure 2 presents qualitative depth estimation results on several outdoor driving scenes. Compared to the ECDDP baseline, our method recovers a smoother road surfaces and sharper discontinuities at object boundaries, while preserving thin structures such as trees and poles. The predicted depth maps align better with the LiDAR ground truth, illustrating the benefit of our representation for event-based depth estimation.

3.2. Attention Map

In addition to depth estimation, we further analyze the discriminative power of our learned features by visualizing attention maps in Figure 1. For each sample, we show the input event frame, the attention responses produced by DINOv2 [5] (our teacher VFM during alignment) and our method, and the corresponding RGB image. As observed in the figure, our approach focuses more accurately on semantically meaningful structures such as object contours, fine edges, and distinctive regions. In contrast, DINOv2 exhibits more diffuse or noisy activations and often fails to localize these structures precisely. These qualitative results indicate that our representation extracts more discriminative features from event data and aligns better with the underlying scene semantics.



(a) Input (b) ECDDP (c) Ours (d) Label (e) RGB

Figure 2. Qualitative comparison of depth estimation on MVSEC [11]. From left to right: (a) event-based input, (b) ECDDP baseline, (c) our method, (d) ground-truth depth, and (e) corresponding reference RGB images (not used). Our model produces depth maps that more closely match the ground truth, with sharper object boundaries and fewer artifacts on thin structures such as trees and poles.

References

[1] 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. 2, 3

[2] Javier Hidalgo-Carrió, Daniel Gehrig, and Davide Scaramuzza. Learning monocular dense depth from events. In *2020 International Conference on 3D Vision (3DV)*, pages 534–542. IEEE, 2020. 2

[3] Junho Kim, Jaehyeok Bae, Gangin Park, Dongsu Zhang, and Young Min Kim. N-imagenet: Towards robust, fine-grained object recognition with event cameras. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 2146–2156, 2021. 3

[4] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017. 1

[5] Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, et al. Dinov2: Learning robust visual features without supervision. *arXiv preprint arXiv:2304.07193*, 2023. 3, 4

[6] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019. 1

[7] René Ranftl, Alexey Bochkovskiy, and Vladlen Koltun. Vision transformers for dense prediction. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 12179–12188, 2021. 2

[8] Carole H Sudre, Wenqi Li, Tom Vercauteren, Sebastien Ourselin, and M Jorge Cardoso. Generalised dice overlap as a deep learning loss function for highly unbalanced segmentations. In *International Workshop on Deep Learning in Medical Image Analysis*, pages 240–248. Springer, 2017. 2

[9] Yan Yang, Liyuan Pan, and Liu Liu. Event camera data pre-training. In *Proceedings of the IEEE/CVF international conference on computer vision*, 2023. 1

[10] Yan Yang, Liyuan Pan, and Liu Liu. Event camera data dense pre-training. In *European Conference on Computer Vision*, pages 292–310. Springer, 2024. 2

[11] 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. 4