I Have a Stream: Making Self-Supervised Learning Work on Continuous Video

Ivan Martinović[1,2] **Lukas Knobel**[2] **Yuki M. Asano**[2]
[1]Faculty of Electrical Engineering and Computing, University of Zagreb
[2]Fundamental AI Lab, University of Technology Nuremberg
ivan.martinovic@fer.hr {lukas.knobel,yuki.asano}@utn.de
https://martinovicivan.github.io/StreamMAE

Abstract

Self-supervised learning draws inspiration from infant visual development, yet standard training pipelines bear little resemblance to it: images are independently sampled and globally shuffled across epochs. We study self-supervised learning from continuous video streams, where frames are consumed in temporal order using strict sliding-window batches, without global reshuffling or multi-epoch replay. To this end, we construct WT++, a 95-hour urban walking-tour video dataset for streaming pretraining. Combined with a comprehensive evaluation suite we find that contrastive and distillation-based methods struggle in this setting, while MAE is more robust but still falls short of standard i.i.d. pretraining. We find that high *inter*-batch similarity, caused by sliding-window consumption across consecutive batches, does not explain this gap. The main challenge is high *intra*-batch similarity, where frames within each batch are near-duplicates. To mitigate this, we propose StreamMAE, which preserves the core MAE reconstruction objective while adapting the input pipeline with stream-aware regularization and motion-biased crop selection. StreamMAE outperforms streaming baselines, matches i.i.d. MAE trained on the same video data, remains competitive with ImageNet-pretrained MAE, and scales positively as the pretraining stream grows from 12 to 95 hours.

1 Introduction

Self-supervised learning (SSL) methods, such as MAE [28] and DINO [5, 37], have advanced visual representation learning. These models are trained on large image collections [15, 37, 42] that are shuffled, often revisited over multiple epochs, and sampled to form diverse batches. This approximates an independent-and-identically-distributed (i.i.d.) regime at the batch level: batches contain diverse, unrelated examples, supporting stable optimization and broad coverage. While SSL is often motivated by how infants and animals learn without explicit high-level supervision, such as language, this pretraining regime remains far from how visual experience arrives in practice.

Natural visual experience is inherently sequential. A camera, robot, or embodied agent observes the world as a temporally ordered stream, where consecutive frames are highly similar and visual content evolves slowly over time. This setting removes a central convenience of standard gradient-based SSL: the ability to globally shuffle data and construct diverse batches. We therefore ask: *Can SSL work well on continuous video streams and scale with more video data when trained from random initialization, without global shuffling, multi-epoch training, or a long-term replay buffer?*

Learning from streaming video is appealing not only as a fundamental research question, but also for embodied agents and edge devices, where data naturally arrive sequentially and need not be stored as a fixed training dataset. However, this regime challenges standard gradient-based training: instead of diverse batches that approximate the data distribution, streaming batches come from short consecutive

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

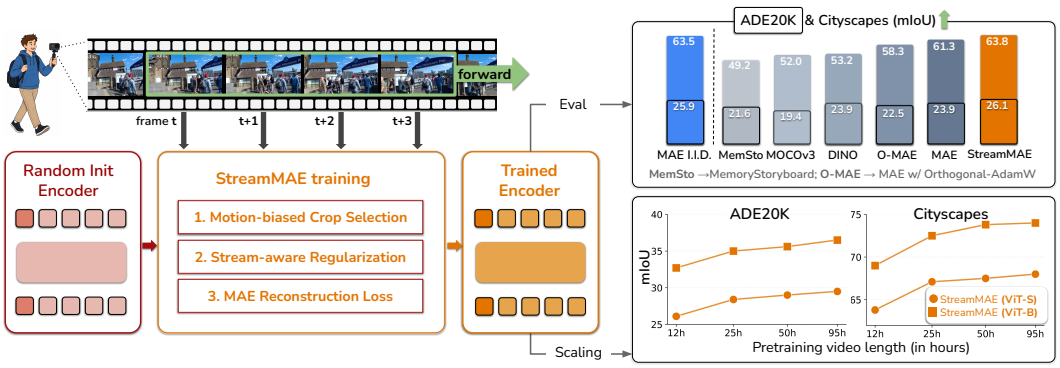

Figure 1: Left: StreamMAE adapts MAE to continuous video streams with stream-aware regularization and crop selection. Right: StreamMAE (ViT-S) improves dense performance over streaming SSL baselines (top) and improves with longer pretraining streams and a larger backbone (bottom).

video segments, are highly redundant, and often contain only small visual changes. As a result, SSL methods that work well on large image collections may not be the best fit for continuous video.

Several recent works study learning from continuous video, but typically relax the strict from-scratch streaming setting: they focus on online prediction or adaptation with pretrained initialization [6], address correlated updates without fully closing the from-scratch degradation [23], or use replay buffers to restore batch diversity [57]. In contrast, we study SSL from scratch: models consume video in temporal order using sliding-window batches and are trained without long-term replay buffers.

To study this setting, we extend the released WalkingTours (WT) dataset [50] into WT++, a 95-hour collection of walking-tour videos for streaming pretraining. We evaluate learned representations across several downstream vision tasks, covering both close-to-domain and out-of-domain benchmarks.

We first benchmark common SSL objectives under this streaming setup, including contrastive learning with MoCo v3 [10], self-distillation with DINO [5], and masked reconstruction with MAE [28]. Under streaming pretraining, MoCo v3 and DINO lag behind MAE (see Figure 1, right), making masked reconstruction the strongest starting point in our setting. However, streaming MAE still falls short of standard i.i.d. MAE trained on the same visual data. We investigate this gap by separating two effects: *inter-batch similarity*, i.e., similarity between consecutive batches caused by fixed-order sliding-window consumption, and *intra-batch similarity*, i.e., similarity among examples within the same batch. To isolate these effects, we pre-shuffle ImageNet-1K [15] once and then consume it as a fixed stream. MAE trained on this pre-shuffled stream matches standard i.i.d. MAE, showing that inter-batch similarity alone does not explain the gap. The main difficulty instead comes from high intra-batch similarity in continuous video, where each batch contains many near-duplicate frames.

Motivated by this analysis, we propose StreamMAE, a simple adaptation of MAE for streaming video. As summarized in Figure 1, StreamMAE preserves the MAE reconstruction objective while adapting the training pipeline to reduce the effect of temporally redundant batches. It combines stronger regularization, two-stage cropping, and motion-biased crop selection based on patch-level frame differences. StreamMAE outperforms streaming baselines, narrows the gap to standard i.i.d. MAE on the same video data, and scales with encoder capacity and pretraining duration (Figure 1, bottom right). These results suggest that masked reconstruction, combined with stream-aware sampling and regularization, is a strong foundation for self-supervised learning from continuous video streams.

2 Related Work

Self-supervised image encoders. Self-supervised image encoders have become standard backbones for vision tasks, transferring well to classification, segmentation, depth estimation, 3D geometry, and other downstream tasks [37, 54, 55]. They are trained with a range of objectives. Contrastive methods, such as MoCo [9, 10, 27], learn by contrasting different image views. Self-distillation methods, such as DINO and DINOv2 [5, 37], align student and teacher representations. Masked modeling methods, such as MAE [28], I-JEPA [4], and CAPI [14], learn from partially observed images by reconstructing pixels or predicting representations. Some works also train image encoders from video, often using temporal information through ordering or motion [44, 45], or by reconstructing future

patches [20]. Closely related to our data source, Venkataramanan et al. [50] and Han et al. [22] study SSL of image encoders from walking-tour videos. Despite their success, these methods are typically developed under i.i.d.-style pretraining on large image collections or offline video datasets, where samples can be shuffled and batches contain diverse examples. We instead study how standard image SSL objectives behave on continuous video streams with high intra-batch similarity.

Learning from streaming video. Video SSL has also been studied with explicitly temporal objectives, including temporal ordering [36], predictive coding [21], and masked video modeling [49, 52]. However, most video SSL methods still assume offline access to videos, where clips can be sampled randomly, shuffled into batches, and revisited over multiple epochs. In contrast, we keep the encoder image-based and study a streaming setting in which frames are consumed in temporal order.

While several methods study continual SSL [17, 29] or online learning from images [24, 25], only a few works consider self-supervised learning from continuous video streams directly. Prior methods often mitigate temporal correlation through replay or memory mechanisms, including infinite or minimum-redundancy replay buffers [40, 59] and reservoir-based [51] memory over temporally segmented videos [57]. Relatedly, Mall and Henriques [34] study supervised continual video classification using a rolling replay buffer of compressed video codes. Carreira et al. [6] study online learning from a single continuous stream, but focuses on online prediction and adaptation, with its strongest setups relying on ImageNet [15] initialization. Wang et al. [53] use masked reconstruction to adapt a task-trained model during deployment, making their approach complementary to our pretraining setting. Closest to our work, Han et al. [23] target correlated gradient updates, but training from scratch remains substantially below settings initialized from pretrained models. In contrast, we study from-scratch representation learning from ordered video streams and adapt MAE through streaming-aware regularization and crop selection, without relying on long-term replay buffers.

3 Learning from a Streaming Video

3.1 Streaming Sliding-Window Batches

We study a streaming learning setting in which a model is trained on a continuous video stream, with each batch formed from a sliding window that advances through the stream in fixed temporal order. Let the raw video be:

$$\mathbf{V} = (\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N), \qquad \mathbf{V} \in \mathbb{R}^{N \times H \times W \times 3}, \tag{1}$$

where each frame $\mathbf{x}_i$ is an RGB image. Given a batch size B and stride s, we form batches as sliding windows over the stream. With training steps indexed from $t = 0$, two consecutive batches are:

$$\mathbf{X}^{(t)} = (\mathbf{x}_{1+ts}, \ldots, \mathbf{x}_{B+ts}), \qquad \mathbf{X}^{(t+1)} = (\mathbf{x}_{1+(t+1)s}, \ldots, \mathbf{x}_{B+(t+1)s}). \tag{2}$$

Thus, when $s < B$, consecutive batches overlap by $B - s$ frames.

We assume access to a batch of examples at each training step, since learning from a single frame at a time would impose an overly restrictive setting. This setup differs from standard i.i.d. training, where samples are shuffled and batches are formed without preserving temporal structure. Figure 2 illustrates the described streaming setup for $s = 2$ and $B = 4$.

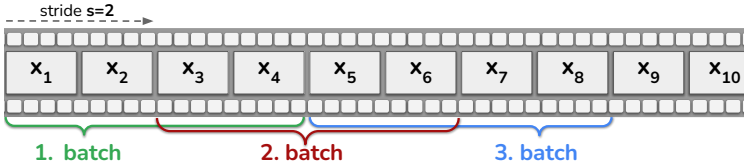

Figure 2: **Streaming pretraining setup.** Frames are observed in temporal order, and each batch is formed as a sliding window over the stream. Here: $B = 4$ and stride $s = 2$, but typically $s \ll B$.

3.2 WT++ Dataset

We train models on urban-scene walking-tour videos, following Venkataramanan et al. [50] and the released WalkingTours (WT) dataset. WT consists of long, single-shot videos recorded while a person walks through different cities. These videos contain diverse objects, lighting conditions, viewpoints, and scene transitions, making them a natural testbed for studying learning from video streams.

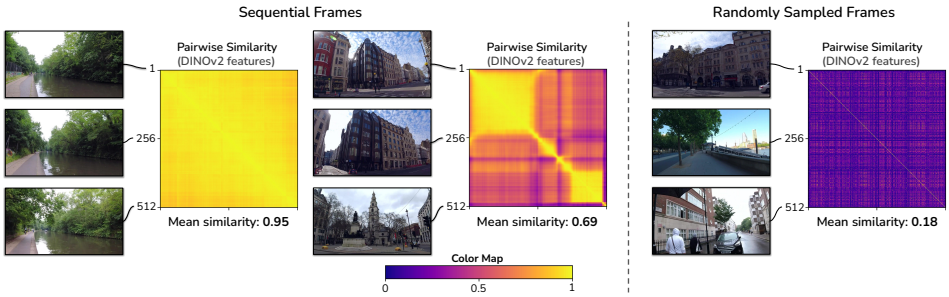

Figure 3: Pairwise DINOv2-L [37] feature similarities for 512 frames from `WT++London` show that consecutive frames at `15 FPS` can be highly similar (left) or contain structured scene changes (block diagonal, middle), whereas randomly sampled frames are markedly more diverse (right).

The WT dataset contains approximately 13 hours of video, which limits larger-scale streaming experiments. We therefore extend it and construct `WT++`, a 95-hour dataset comprising 58 public walking-tour videos. `WT++` includes all videos from the original WT dataset [50], except for the Wildlife safari video. Details are provided in Appendix C.

By default, models are trained on a single 12-hour video, `WT++London` (i.e., `WT++12h`). For larger-scale experiments, we concatenate multiple walking-tour videos into one ordered stream and train with the same streaming protocol. This approximates a long continuous stream, since publicly available single-shot walking-tour videos rarely span tens of hours. The stream remains temporally ordered within each video with discontinuities at video boundaries.

To illustrate the temporal structure of the data, we analyze frame similarity in `WT++London` using DINOv2 [37] features. We compare pairwise cosine similarities for 512 consecutive frames sampled from a local stream window against 512 frames sampled uniformly at random from the same video. As shown in Figure 3, consecutive 15 `FPS` frames can be highly similar, whereas randomly sampled frames are substantially more diverse. This illustrates the distributional gap between streaming and shuffled batches, and motivates methods that can learn from temporally local, correlated data.

4 Towards StreamMAE

Our goal is to design a self-supervised method that can learn from continuous video streams. As suggested by Figure 1, MAE is more robust under streaming pretraining than MoCo v3 [10] and DINO [5]. This is consistent with the nature of their objectives: masked reconstruction operates on each example independently, avoiding the need to contrast examples within similar batches [10] or to cluster visual concepts across a diverse pretraining corpus [5] - both of which become problematic when the data stream is temporally correlated. We therefore build on MAE and use standard i.i.d. sampled pretraining on the same visual data as a reference throughout this work.

However, streaming MAE still falls short of this i.i.d. reference (*cf.* Figure 1), and the gap becomes substantial when scaling to ViT-B/16, underperforming by nearly 10 mIoU points on Cityscapes (*cf.* Table 4). To understand this gap, we identify two key departures from standard i.i.d. pretraining. The first is *inter-batch similarity*: in streaming training, the fixed-order sliding-window consumption means that consecutive batches share most of their examples, leading to highly correlated gradient updates across successive optimization steps. The second is *intra-batch similarity*: because each batch is drawn from a local temporal window of the video stream, the examples within a single batch can be near-duplicate frames depicting the same scene with only minor visual variations.

Table 1: Batch similarity statistics.

Metric	i.i.d. IN-1K	Streaming IN-1K	Streaming `WT++12h`
μ_{intra}	0.004	0.004	0.665
μ_{inter}	0.325	0.989	1.000

To disentangle these two factors, we construct a controlled experiment. We pre-shuffle ImageNet-1K once and treat the resulting sequence as a fixed stream, forming batches with a sliding window of stride $s = 8$. This retains the fixed-order sliding-window consumption of the streaming protocol, and therefore high inter-batch similarity, while removing the near-duplicate frames that characterize video-stream batches, yielding low intra-batch similarity. For each setting (IN-1K-i.i.d., IN-1K pre-shuffled+streaming, and our `WT++12h`), we draw consecutive batches of frames and measure inter-batch and intra-batch cosine similarity using features from a pretrained DINOv2 [37] model: μ_{inter}, μ_{intra} (details in the Appendix E.1).

As shown in Table 1, the three settings span a spectrum. The IN-1K i.i.d. setting exhibits low similarity on both metrics, as expected from random sampling over a diverse dataset. The `WT++12h` stream yields substantially higher values for both: $\mu_{\text{intra}} = 0.665$ reflects high visual similarity within each temporal window, and $\mu_{\text{inter}} = 1.000$ reflects nearly complete overlap between successive sliding-window batches. Crucially, the IN-1K pre-shuffled streaming setting occupies an informative middle ground: it nearly matches the `WT++12h` stream in inter-batch similarity ($\mu_{\text{inter}} = 0.989$) due to the same sliding-window mechanics, but retains the low intra-batch similarity of i.i.d. training ($\mu_{\text{intra}} = 0.004$) because the underlying images are diverse. This decoupled setting allows us to isolate the effect of each factor, which we investigate next.

4.1 Preliminary Analysis

Does inter-batch similarity explain the gap? Given the gap between streaming MAE and standard i.i.d. MAE, we utilize the pre-shuffled IN-1K stream to analyze whether inter-batch similarity is leading to degraded representations. As shown in Table 2, a pre-shuffled IN-1K stream matches standard i.i.d. MAE across several downstream benchmarks. A controlled comparison on `WT++12h` shows the same trend (Appendix A.1). These results suggest that *for MAE, inter-batch similarity alone is not harmful when batches remain visually diverse. The main challenge is high intra-batch similarity induced by near-duplicate frames in a continuous video.*

Table 2: Fixed-order sliding-window training on diverse data. We pre-shuffle ImageNet-1K once and consume it as a fixed stream with stride $s = 8$. The pre-shuffled stream matches standard i.i.d. MAE, suggesting that inter-batch similarity alone does not explain the streaming gap.

Backbone	Training regime	IN-1K Acc@1 ↑	CS mIoU ↑	ADE20K mIoU ↑
ViT-S	Standard i.i.d.	77.4	64.0	26.9
	Pre-shuffled stream	77.4	63.6	27.2
ViT-B	Standard i.i.d.	81.5	72.3	35.9
	Pre-shuffled stream	81.6	73.6	36.5

How does high intra-batch similarity affect optimization? We next analyze whether the gap between standard i.i.d. and streaming pretraining is reflected in the optimization dynamics of MAE-based methods. Following Han et al. [23], we measure the cosine similarity between gradients from consecutive batches, using the MLP parameters in the last transformer block. Figure 4(a) shows the running average (with window 100) of this similarity throughout training, together with the temporal mean for each method. Interestingly, while Han et al. [23] emphasize highly positive gradient correlations in streaming training of DoRA [50] (a method closer to DINO), we find that vanilla streaming MAE can produce weakly negative consecutive-gradient similarity, whereas MAE with Orthogonal-AdamW [23] yields substantially positive similarity. Standard i.i.d. MAE, used as our reference, has more stable near-zero similarity. Here we use $B = 512$ for all methods.

Figure 4(b) relates dense downstream performance to the distance between each method's mean consecutive-gradient similarity and that of standard i.i.d. MAE. Within the MAE family, methods closer to the i.i.d. MAE reference achieve stronger dense performance. This suggests that aligning streaming optimization dynamics with standard i.i.d. MAE can be a useful design principle. Motivated by this observation, StreamMAE keeps the MAE objective unchanged, but adapts the training pipeline through stronger regularization, two-stage cropping, and motion-biased crop selection. As shown in Figure 4, these changes bring the gradient behavior closer to i.i.d. MAE and improve downstream performance. We describe the components next.

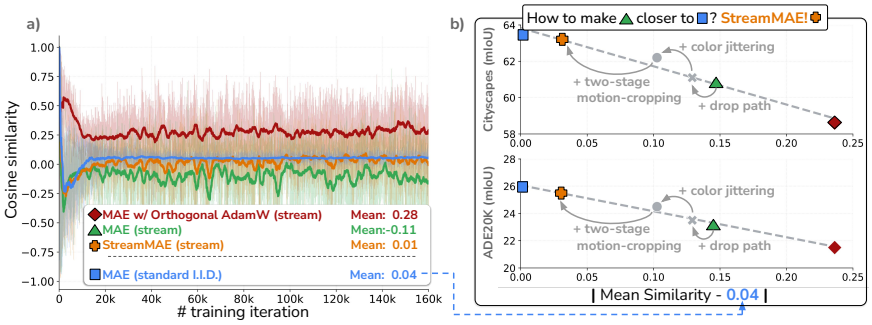

Figure 4: (a) Running average of consecutive batches gradient similarity, measured on last-block MLP parameters every 10 steps. (b) Dense downstream performance versus distance to standard i.i.d. MAE in mean gradient similarity. Methods closer to the i.i.d. MAE reference perform better.

4.2 Regularization under High Intra-Batch Similarity

Streaming video often produces batches with high intra-batch similarity: consecutive frames can contain near-duplicate views of the same scene (*cf.* Figure 3). In this regime, standard MAE may rely on short-term regularities, repeatedly reconstructing similar content with limited variation in appearance or structure. We therefore strengthen the MAE training pipeline with three lightweight mechanisms: color jitter, increased drop path rate, and DataDrop.

Color jitter. Color jitter increases appearance diversity by perturbing low-level statistics such as color, brightness, and contrast. This makes reconstruction less dependent on stable appearance cues that persist across neighboring frames.

Drop path. Increasing the drop path rate [30] regularizes the encoder by varying the effective network depth across updates. This helps reduce overfitting to recurring local patterns in batches with high intra-batch similarity.

DataDrop. In streaming pretraining, each batch corresponds to a sliding window from the ordered stream (Figure 2). When this window contains many highly similar frames, backpropagating through all examples can overemphasize near-duplicate content. We therefore use DataDrop: for each batch, we sample a binary mask over examples and compute the loss only on the retained subset. Dropped examples are not forwarded through the model and do not contribute to the current update. Training then proceeds to the next window according to the stream stride s. By default, we drop 75% of examples in each batch, reducing the effective batch size for backpropagation by a factor of four.

4.3 Stream-Aware Cropping

Two-stage cropping. MAE applies random resized cropping directly to the input frame. In streaming video, however, adjacent frames are often globally similar, so independently sampled crops from neighboring frames can still show near-duplicate content. We therefore use a two-stage cropping procedure that exposes the crop location as a separate design choice. First, we sample a fixed-size crop from the frame, defining a local spatial region. We then apply the standard MAE random resized crop within this region. This preserves the original MAE augmentation pipeline while making the sampled view depend on an explicit first-stage region selection.

This formulation separates *where* to crop from *how* to augment the final training view. A simple version samples the first-stage region uniformly at random. We next make this selection stream-aware by biasing it toward regions with recent visual change.

Motion-biased crop selection. Let $\mathbf{x}_t \in \mathbb{R}^{H\times W\times 3}$ denote the current frame and $\mathbf{x}_{t-1}$ the previous frame. We compute a per-pixel frame-difference map:

$$\mathbf{D}_t(h,w) = \left\|\mathbf{x}_t(h,w,:) - \mathbf{x}_{t-1}(h,w,:)\right\|_1, \tag{3}$$

where (h,w) indexes spatial locations and the ℓ_1 norm is taken over RGB channels. This provides a lightweight estimate of local temporal change without requiring optical flow [11, 48]. We aggregate $\mathbf{D}_t$ at the same patch granularity as the ViT encoder. Let Ω_p denote the set of pixel locations contained in patch p. The motion score of patch p is:

$$m_t(p) = \frac{1}{|\Omega_p|}\sum_{(h,w)\in\Omega_p} \mathbf{D}_t(h,w). \tag{4}$$

The resulting patch-level map $\mathbf{m}_t$ highlights regions of the current frame that differ most from the preceding frame (*cf.* Figure 5). We use this map to select the first-stage crop. Given K first-stage candidate crops $\mathcal{C} = \{c_1,\dots,c_K\}$, let $\mathcal{P}(c)$ denote the set of ViT patches covered by crop c. We select the candidate with the largest average patch-level motion:

$$c_t^\star = \arg\max_{c\in\mathcal{C}} \frac{1}{|\mathcal{P}(c)|}\sum_{p\in\mathcal{P}(c)} m_t(p). \tag{5}$$

We then apply the standard MAE random resized crop within $c_t^\star$. Thus, motion-biased crop selection preserves the MAE objective and final augmentation pipeline while biasing the first-stage region toward parts of the video that change over time. The case $K = 1$ reduces to the random two-stage cropping procedure described above. In practice, we apply motion-biased selection with probability 0.5 and otherwise fall back to standard random resized cropping. This stochastic choice avoids an overly deterministic focus on the same high-motion regions in video segments.



Figure 5: Motion-biased crop selection computes patch-level differences between consecutive frames $\mathbf{x}_t$ and $\mathbf{x}_{t-1}$, samples K crop candidates $\mathcal{C}$, and selects the crop $c_t^\star$ with the largest average ℓ_1 score.

5 Experiments

Training details. We use ViT-Small (ViT-S/16) [16] as our main architecture and report ViT-B/16 results for StreamMAE when scaling encoder size. All SSL methods are implemented in solo-learn [13] and trained from scratch. We use AdamW [32] with $\beta_1 = 0.9$, $\beta_2 = 0.95$, and base learning rates 8×10^{-5} for ViT-S and 4×10^{-5} for ViT-B. The learning rate is scaled linearly with the effective batch size, $B_{\text{eff}}/256$. Streaming runs use a constant learning rate with linear warm-up, consistent with the goal of continuing pretraining as new videos arrive.

Our default pretraining stream is `WT++12h`, a 12-hour London walking-tour video from `WT++`. For larger-scale pretraining, we sequentially append videos to form `WT++25h`, `WT++50h`, and `WT++95h`, corresponding to approximately 25, 50, and 95 hours of data. Each stream extends the preceding one by appending new videos while retaining all earlier videos in the same order (*cf.* Appendix C). The original videos are recorded at 60 FPS and processed at 1280×720. We temporally subsample them by a factor of $k = 16$, yielding 3.75 FPS. Streaming batches use stride $s = 1$, except for `WT++95h` experiments where we use $s = 2$ to reduce training time. We form local stream windows of size $B = 2048$ and apply DataDrop with drop probability 0.75, yielding effective batch size $B_{\text{eff}} = 512$.

For MAE-based methods, we use the standard masking ratio 0.75 with random uniform masking [28]. We compare this choice with two alternatives in Appendix A.6. The default StreamMAE uses color jitter ($p = 0.5$), a linear drop-path schedule reaching 0.25 in the final block, DataDrop, two-stage cropping, and motion-biased crop selection with $K = 4$ candidate crops. Further implementation details, including crop selection and compute requirements, are provided in Appendix E and G.

Evaluation. Following established MAE evaluation protocols [28] and recent recommendations for semantic segmentation [31], we use end-to-end fine-tuning as our primary evaluation protocol. We additionally report attentive probing for classification in Table 10 to evaluate the frozen representations. For classification, we use ImageNet-1K [15]. For dense prediction, we use close-to-domain outdoor benchmarks, Cityscapes [12] and KITTI [19], and out-of-domain benchmarks, ADE20K [58] and NYU-Depth-v2 [46]. We initialize the encoder from the pretrained checkpoint and fine-tune it with a task-specific head: a linear head for segmentation and DPT [41] for depth estimation. We report top-1 val accuracy for classification, mIoU for segmentation, and RMSE for depth estimation. For depth estimation, we report mean$_{\pm\text{std}}$ over two seeds. More details are provided in Appendix F.

5.1 Benchmarking Self-Supervised Methods under Streaming Pretraining

We first benchmark representative SSL methods under our streaming protocol and evaluate representations with full fine-tuning on downstream tasks. Details of baseline tuning and adaptations to temporal redundancy are provided in Appendices D and E.4. As shown in Table 3, MAE is a stronger streaming baseline than MoCo v3 [10] and DINO [5]. MoCo v3 even falls below random initialization on ImageNet-1K, consistent with contrastive learning being sensitive to high intra-batch similarity where near-duplicate frames can act as false negatives. DINO is more stable, but still trails MAE on most downstream tasks. Streaming-tailored baselines do not close the gap. MAE with Orthogonal-AdamW [23] underperforms standard streaming MAE, and our reproduction of MemoryStoryboard [57] with ViT-S remains below MAE despite using a replay buffer (*cf.* Appendix D). In contrast, StreamMAE improves over all streaming baselines and is comparable to standard IID MAE trained on the same `WT++12h` data. Scaling StreamMAE to `WT++95h` further improves transfer, especially on dense prediction. We also include an ImageNet-1K IID MAE reference trained for the same number of iterations as the `WT++12h` streaming runs.

Table 3: **Benchmarking SSL methods under streaming pretraining (ViT-S).** We compare WT++12h streaming methods to random initialization, i.i.d. MAE on WT++12h, and an iteration-matched ImageNet-1K i.i.d. MAE reference. † marks reproduced streaming-tailored methods.

Pretraining Method	Pretraining Data	IN-1K [15] Acc @ 1 ↑	City [12] mIoU ↑	ADE [58] mIoU ↑	NYUv2 [46] RMSE ↓	KITTI [19] RMSE ↓
Random Init	–	71.9	47.7	16.7	$0.877_{\pm 0.013}$	$6.108_{\pm 0.028}$
			Standard I.I.D. Setup			
MAE – standard IID	WT++12h	77.0	63.5	25.9	$0.701_{\pm 0.004}$	$4.070_{\pm 0.006}$
MAE – standard IID	ImageNet-1K	77.4	64.0	26.9	$0.656_{\pm 0.004}$	$3.854_{\pm 0.008}$
			Streaming Setup			
MOCO-v3 [10]	WT++12h	68.7	52.0	19.4	$0.760_{\pm 0.000}$	$4.882_{\pm 0.054}$
DINO [5]	WT++12h	72.7	53.2	23.9	$0.734_{\pm 0.003}$	$4.454_{\pm 0.034}$
MAE [28]	WT++12h	77.1	61.3	23.9	$0.742_{\pm 0.002}$	$4.211_{\pm 0.137}$
Orthogonal-MAE [23]†	WT++12h	75.9	58.3	22.5	$0.749_{\pm 0.004}$	$4.209_{\pm 0.036}$
MemoryStoryboard [57]†	WT++12h	71.0	49.2	21.6	$0.792_{\pm 0.001}$	$4.647_{\pm 0.038}$
StreamMAE (ours)	WT++12h	**77.5**	**63.8**	**26.1**	$\mathbf{0.694_{\pm 0.003}}$	$\mathbf{3.976_{\pm 0.012}}$
StreamMAE (ours)	WT++95h	**78.4**	**68.0**	**29.5**	$\mathbf{0.646_{\pm 0.003}}$	$\mathbf{3.765_{\pm 0.047}}$

Ablation study. We perform cumulative ablations using a ViT-S/16 backbone pretrained on WT++12h. As shown in Figure 6, the proposed design choices progressively improve dense transfer. DataDrop yields modest gains, while stronger regularization and crop selection account for most of the improvement. Motion-biased crop selection gives the largest gains on close-to-domain outdoor benchmarks, Cityscapes [12] and KITTI [19]. Later experiments show that DataDrop becomes negligible at larger scale, whereas regularization and crop selection remain the main components. The MAE baseline in this ablation uses $B = 512$, which gives stronger performance in our setup (*cf.* Appendix A.4).

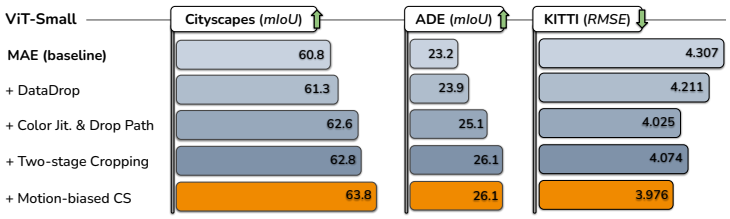

Figure 6: **Cumulative ablation study with ViT-S/16 pretrained on** WT++12h. Each row adds one component to the previous setting. DataDrop gives modest gains, while stronger regularization and crop selection account for most of the dense-transfer improvement. CS denotes crop selection.

5.2 Scaling and Analysis

We next analyze the scaling behavior of StreamMAE by varying encoder capacity and pretraining duration, and examine whether its design choices remain beneficial at larger scale.

Scaling encoder capacity. We scale the encoder to ViT-B/16 while keeping the pretraining stream fixed to WT++12h. As shown in Table 4, increasing model capacity alone does not close the gap between streaming baseline and standard i.i.d. MAE, particularly on dense prediction. Orthogonal-MAE provides modest gains over streaming MAE, but remains below StreamMAE. In contrast, StreamMAE scales effectively: it nearly matches standard i.i.d. MAE on ImageNet-1K and ADE20K, surpasses it on Cityscapes, and substantially improves depth estimation over streaming MAE. The ImageNet-1K i.i.d. MAE reference is trained for the same number of iterations as the WT++12h runs.

Scaling pretraining duration. We next scale the pretraining stream by sequentially appending additional walking-tour videos to WT++12h. As shown in Table 5, performance generally improves with longer streams for both ViT-S/16 and ViT-B/16. We additionally compare the WT++95h runs against standard i.i.d. MAE pretrained on ImageNet-1K under the same longer budget of 650k iterations, as reported in Appendix Table 21. With ViT-S/16, StreamMAE matches this longer reference on IN-1K and exceeds it on both segmentation benchmarks. With ViT-B/16, it remains competitive on IN-1K and Cityscapes but trails on ADE20K. Overall, these results show that StreamMAE benefits from additional streaming data while remaining close to the longer-budget, iteration-matched i.i.d. MAE reference on most benchmarks.

Table 4: **Increasing encoder size**. StreamMAE (ViT-B) improves over both StreamMAE (ViT-S) and the MAE baseline, while closing most of the gap to standard i.i.d. MAE. † marks our reproduction.

Pretraining Method	Encoder	Pretraining Data	IN-1K Acc @ 1 ↑	Cityscapes mIoU ↑	ADE20K mIoU ↑	NYUv2 RMSE ↓	KITTI RMSE ↓
Random Init	ViT-B	–	78.5	51.0	17.5	$0.881_{\pm 0.002}$	$5.818_{\pm 0.137}$
Standard I.I.D. Setup							
MAE – standard IID	ViT-B	WT++12h	81.2	67.8	32.9	$0.675_{\pm 0.004}$	$3.855_{\pm 0.048}$
MAE – standard IID	ViT-B	ImageNet-1K	81.5	72.3	35.9	$0.588_{\pm 0.003}$	$3.771_{\pm 0.039}$
Streaming Setup							
StreamMAE (ours)	ViT-S	WT++12h	77.5	63.8	26.1	$0.694_{\pm 0.003}$	$3.976_{\pm 0.012}$
MAE [28]	ViT-B	WT++12h	80.0	58.2	26.2	$0.753_{\pm 0.005}$	$4.543_{\pm 0.046}$
Orthogonal-MAE [23]†	ViT-B	WT++12h	80.0	60.6	27.3	$0.750_{\pm 0.002}$	$4.289_{\pm 0.011}$
StreamMAE (ours)	ViT-B	WT++12h	**81.1**	**69.0**	**32.7**	$\mathbf{0.655_{\pm 0.002}}$	$\mathbf{3.797_{\pm 0.006}}$
StreamMAE (ours)	ViT-B	WT++95h	**82.0**	**74.0**	**36.5**	$\mathbf{0.584_{\pm 0.002}}$	$\mathbf{3.496_{\pm 0.009}}$

Table 5: **Scaling pretraining video stream duration.**

	WT duration	IN-1K ↑	CS ↑	ADE ↑	NYUv2 ↓	KITTI ↓
ViT-S	WT++12h	77.5	63.8	26.1	$0.694_{\pm 0.003}$	$3.976_{\pm 0.012}$
	WT++25h	78.1	67.1	28.4	$0.659_{\pm 0.003}$	$\mathbf{3.747_{\pm 0.048}}$
	WT++50h	78.2	67.5	29.0	$0.656_{\pm 0.001}$	$3.895_{\pm 0.041}$
	WT++95h	**78.4**	**68.0**	**29.5**	$\mathbf{0.646_{\pm 0.003}}$	$3.765_{\pm 0.047}$
ViT-B	WT++12h	81.1	69.0	32.7	$0.655_{\pm 0.002}$	$3.797_{\pm 0.006}$
	WT++25h	81.6	72.5	35.0	$0.617_{\pm 0.003}$	$3.619_{\pm 0.009}$
	WT++50h	81.9	73.8	35.6	$0.600_{\pm 0.003}$	$3.620_{\pm 0.023}$
	WT++95h	**82.0**	**74.0**	**36.5**	$\mathbf{0.584_{\pm 0.002}}$	$\mathbf{3.496_{\pm 0.009}}$

Table 6: **Effect of DataDrop at scale.** Its contribution is negligible for ViT-B on longer pretraining videos.

	WT duration	DataDrop	IN-1K Acc@1 ↑	CS (mIoU) ↑	ADE (mIoU) ↑
ViT-S	WT++50h	✓	78.2	67.5	29.0
	WT++50h	✗	78.3	66.0	29.4
ViT-B	WT++50h	✓	81.9	73.8	35.6
	WT++50h	✗	81.7	73.6	36.4
	WT++95h	✓	82.0	74.0	36.5
	WT++95h	✗	81.9	74.3	36.3

We further analyze the role of DataDrop under this scaling regime in Table 6. While DataDrop provides modest gains in the default ViT-S/16 setting, its effect becomes negligible for ViT-B/16 when scaling pretraining to WT++50h or WT++95h. This suggests that DataDrop is useful at smaller scale (see also Figure 6), but is not the main driver of performance once model capacity and stream duration are increased. Here, the no-DataDrop setting uses $B = 512$, corresponding to roughly 136 seconds of video at 3.75 FPS. Additional batch-size sensitivity results are provided in Appendix A.3.

Checkpoint averaging for long streaming runs. Inspired by continual learning setups [35, 43], we additionally examine whether averaging checkpoints from a long streaming run can improve the final representation. For ViT-B/16 pretrained on WT++95h, we average weights from checkpoints saved at different stages of the same training trajectory. Percentages denote the fraction of training completed at each checkpoint. As shown in Table 7, this simple procedure improves dense prediction performance while leaving IN-1K accuracy essentially unchanged. These results suggest that checkpoints from different stages of pretraining may retain complementary information.

Table 7: **Checkpoint averaging** on the same ViT-B WT++95h trajectory improves dense transfer with negligible change in IN-1K accuracy.

Method	WT duration	IN-1K Acc@1 ↑	CS (mIoU) ↑	ADE (mIoU) ↑
StreamMAE	WT++95h	**82.0**	74.0	36.5
Avg. {40,60,80,100}%	WT++95h	81.9	**75.5**	37.0
Avg. {60,100}%	WT++95h	81.9	75.3	**37.3**

More updates versus new visual content. We next ask whether scaling gains come from more optimization steps or from new visual content. Table 8 shows that increasing the effective number of updates by using higher FPS does not improve transfer and can hurt dense prediction. Table 9 shows that StreamMAE can still benefit from additional epochs over the same stream, indicating that it does not require extreme data diversity to improve. However, streams that include new videos

Table 8: **Effect of temporal subsampling on WT++12h with ViT-B.** Denser sampling increases the effective number of updates, but does not improve performance.

Subsampling Factor (k)	IN-1K Acc@1 ↑	CS (mIoU) ↑	ADE (mIoU) ↑
$k = 4$ (15 FPS)	81.0	65.2	30.1
$k = 8$ (7.5 FPS)	81.0	66.8	31.4
$k = 16$ (3.75 FPS)	**81.1**	**69.0**	**32.7**

Table 9: **Effect of repeating the data.** Additional passes over WT++12h improve performance, but new videos yield larger gains.

WT++ data	IN-1K	CS	ADE
WT++12h	81.1	69.0	32.7
WT++12h×2	81.3	70.3	33.3
WT++25h	81.6	72.5	35.0
WT++12h×4	81.5	71.7	33.9
WT++50h	**81.9**	**73.8**	**35.6**

perform better than repeated epochs over WT++12h, suggesting that newly introduced scenes provide additional gains. These results suggest that longer-stream improvements are not explained by more updates alone. StreamMAE benefits from both continued optimization and increased visual diversity.

Attentive probing. We evaluate frozen representations on IN-1K by training only a single-query cross-attention pooling layer followed by a linear classifier. We follow the probing hyperparameters of Psomas et al. [39] and the augmentations of AIMv2 [18]. As shown in Table 10, StreamMAE improves over streaming MAE by 6.8 and 9.8 Acc@1 points for ViT-S/16 and ViT-B/16, respectively, while matching same-data i.i.d. MAE.

Table 10: ImageNet-1K attentive probing evaluation after pretraining on WT++12h.

Method	ViT-S/16		ViT-B/16	
	Acc@1 ↑	Acc@5 ↑	Acc@1 ↑	Acc@5 ↑
MAE – standard i.i.d.	40.0	63.0	46.4	69.0
MAE (streaming)	33.4	55.8	37.3	59.6
StreamMAE	**40.2**	**63.6**	**47.1**	**69.9**

5.3 Generalization Across Pretraining Domains

To examine whether StreamMAE extends beyond walking-tour video, we additionally pretrain ViT-B/16 on three diverse streams: HD-EPIC [38], an indoor egocentric kitchen stream, CROWD [3], a front-facing urban dashcam stream, and KrishnaCAM [47], a longitudinal egocentric daily-life stream. For each pretraining dataset, we compare streaming MAE, StreamMAE, and standard i.i.d. MAE using the same frames and number of pretraining iterations.

Table 11: **Generalization across pretraining domains using ViT-B/16**.

Pretraining Data	Method	CS (mIoU) ↑	ADE (mIoU) ↑	NYUv2 (RMSE) ↓	KITTI (RMSE) ↓
HD-EPIC [38]	MAE – standard i.i.d.	65.2	30.7	$0.681_{\pm 0.001}$	$4.184_{\pm 0.081}$
	MAE (streaming)	63.2	29.3	$0.711_{\pm 0.000}$	$4.206_{\pm 0.003}$
	StreamMAE (ours)	**65.6**	**32.0**	$\mathbf{0.652_{\pm 0.001}}$	$\mathbf{4.030_{\pm 0.049}}$
CROWD [3]	MAE – standard i.i.d.	68.1	32.3	$0.681_{\pm 0.000}$	$3.830_{\pm 0.026}$
	MAE (streaming)	65.7	29.1	$0.704_{\pm 0.004}$	$4.287_{\pm 0.147}$
	StreamMAE (ours)	**71.5**	**32.6**	$\mathbf{0.644_{\pm 0.001}}$	$\mathbf{3.756_{\pm 0.046}}$
KrishnaCAM [47]	MAE – standard i.i.d.	71.7	**34.6**	$\mathbf{0.616_{\pm 0.004}}$	$3.596_{\pm 0.003}$
	MAE (streaming)	67.9	32.9	$0.663_{\pm 0.009}$	$3.799_{\pm 0.010}$
	StreamMAE (ours)	**72.2**	34.5	$0.621_{\pm 0.001}$	$\mathbf{3.583_{\pm 0.012}}$

As shown in Table 11, StreamMAE consistently improves over streaming MAE across all four downstream tasks and all three domains, with gains of 1.6–5.8 mIoU on segmentation and reductions of 0.042–0.531 RMSE on depth estimation. On HD-EPIC and CROWD, StreamMAE matches or exceeds same-data i.i.d. MAE on every downstream task. On KrishnaCAM, it trails i.i.d. MAE by 0.1 mIoU on ADE20K and 0.005 RMSE on NYUv2, while outperforming it on Cityscapes and KITTI. Details of preprocessing and stream construction for all three datasets are provided in Appendix E.5.

Robustness to irregular camera motion. We further ablate motion-biased crop selection on KrishnaCAM [47], whose head-mounted viewpoint exhibits irregular camera motion across indoor and outdoor scenes. Relative to StreamMAE without motion-biased crop selection, the full method improves ADE20K and Cityscapes by 0.6 and 0.7 mIoU, respectively, and reduces KITTI RMSE by 0.063, while leaving NYUv2 unchanged (full results are provided in Appendix Table 20). Motion-biased crop selection therefore remains beneficial beyond the walking-tour domain.

6 Conclusion

We studied self-supervised learning from continuous video, where models are trained from scratch on temporally ordered frames without global reshuffling or long-term replay buffers. We showed that contrastive [10] and self-distillation [5] methods struggle in this setting, while masked reconstruction provides a more robust starting point. Our analysis suggests that the main challenge for MAE is not inter-batch similarity from fixed-order sliding-window consumption, but high intra-batch similarity caused by near-duplicate frames. Based on this insight, we introduced StreamMAE, which keeps the MAE objective while adapting the training pipeline to mitigate the effects of high intra-batch similarity and focus learning on more informative regions of the stream. StreamMAE outperforms streaming baselines, is competitive with standard i.i.d. MAE, and scales to longer video streams.

Acknowledgments and Disclosure of Funding

We thank Tengda Han for inspiring discussions and Ryousuke Yamada, Ivan Grubišić, Josip Šarić, Valentinos Pariza, and Ivan Sabolić for their feedback on the manuscript. This research has been supported by Croatian Science Foundation through grants DOK-NPOO-2023-10-2288, MOBDOK-2023-4880 and UIP-2026-7439 SERVER. The authors gratefully acknowledge the scientific support and HPC resources provided by the Erlangen National High Performance Computing Center (NHR@FAU) of the Friedrich-Alexander-Universität Erlangen-Nürnberg (FAU) under the BayernKI project `v115be`. BayernKI funding is provided by Bavarian state authorities.

References

[1] Amro Kamal Mohamed Abbas, Kushal Tirumala, Daniel Simig, Surya Ganguli, and Ari S Morcos. SemDeDup: Data-efficient learning at web-scale through semantic deduplication. In *ICLR 2023 Workshop on Mathematical and Empirical Understanding of Foundation Models*, 2023. 20

[2] Alireza Aghabagherloo, Aydin Abadi, Sumanta Sarkar, Vishnu Asutosh Dasu, and Bart Preneel. Impact of Data Duplication on Deep Neural Network-Based Image Classifiers: Robust vs. Standard Models. In *IEEE Security and Privacy Workshops (SPW)*, 2025. 20

[3] Md Shadab Alam, Olena Bazilinska, and Pavlo Bazilinskyy. A global dataset of continuous urban dashcam driving. *arXiv:2604.01044*, 2026. 10, 22

[4] Mahmoud Assran, Quentin Duval, Ishan Misra, Piotr Bojanowski, Pascal Vincent, Michael Rabbat, Yann LeCun, and Nicolas Ballas. Self-supervised learning from images with a joint-embedding predictive architecture. In *CVPR*, 2023. 2, 16

[5] Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging properties in self-supervised vision transformers. In *ICCV*, 2021. 1, 2, 4, 7, 8, 10, 21

[6] João Carreira, Michael King, Viorica Patraucean, Dilara Gokay, Catalin Ionescu, Yi Yang, Daniel Zoran, Joseph Heyward, Carl Doersch, Yusuf Aytar, et al. Learning from one continuous video stream. In *CVPR*, 2024. 2, 3, 16

[7] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *ICML*, 2020. 19

[8] Xinlei Chen and Kaiming He. Exploring simple siamese representation learning. In *CVPR*, 2021. 19

[9] Xinlei Chen, Haoqi Fan, Ross Girshick, and Kaiming He. Improved baselines with momentum contrastive learning. *arXiv preprint arXiv:2003.04297*, 2020. 2

[10] Xinlei Chen, Saining Xie, and Kaiming He. An empirical study of training self-supervised vision transformers. In *ICCV*, 2021. 2, 4, 7, 8, 10, 16, 21

[11] Rohan Choudhury, Guanglei Zhu, Sihan Liu, Koichiro Niinuma, Kris M. Kitani, and László A. Jeni. Don't Look Twice: Faster Video Transformers with Run-Length Tokenization. In *NeurIPS*, 2024. 6

[12] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The Cityscapes Dataset for Semantic Urban Scene Understanding. In *CVPR*, 2016. 7, 8, 29

[13] Victor Guilherme Turrisi da Costa, Enrico Fini, Moin Nabi, Nicu Sebe, and Elisa Ricci. solo-learn: A Library of Self-supervised Methods for Visual Representation Learning. *JMLR*, 2022. 7, 21, 29

[14] Timothée Darcet, Federico Baldassarre, Maxime Oquab, Julien Mairal, and Piotr Bojanowski. Cluster and Predict Latents Patches for Improved Masked Image Modeling. *TMLR*, 2025. 2

[15] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. ImageNet: A large-scale hierarchical image database. In *CVPR*, 2009. 1, 2, 3, 7, 8, 22, 29

[16] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. In *ICLR*, 2021. 7

[17] Enrico Fini, Victor G Turrisi Da Costa, Xavier Alameda-Pineda, Elisa Ricci, Karteek Alahari, and Julien Mairal. Self-supervised models are continual learners. In *CVPR*, 2022. 3

[18] Enrico Fini, Mustafa Shukor, Xiujun Li, Philipp Dufter, Michal Klein, David Haldimann, Sai Aitharaju, Victor G Turrisi da Costa, Louis Béthune, Zhe Gan, et al. Multimodal Autoregressive Pre-training of Large Vision Encoders. In *CVPR*, 2025. 10

[19] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for Autonomous Driving? The KITTI Vision Benchmark Suite. In *CVPR*, 2012. 7, 8, 29

[20] Agrim Gupta, Jiajun Wu, Jia Deng, and Fei-Fei Li. Siamese Masked Autoencoders. In *NeurIPS*, 2023. 3

[21] Tengda Han, Weidi Xie, and Andrew Zisserman. Video representation learning by dense predictive coding. In *Proceedings of the IEEE/CVF international conference on computer vision workshops*, 2019. 3

[22] Tengda Han, Sayna Ebrahimi, Dilara Gokay, Li Yang Ku, Maks Ovsjanikov, Iva Babukova, Daniel Zoran, Viorica Patraucean, Joao Carreira, Andrew Zisserman, and Dima Damen. Unique Lives, Shared World: Learning from Single-Life Videos. *arXiv:2512.04085*, 2025. 3

[23] Tengda Han, Dilara Gokay, Joseph Heyward, Chuhan Zhang, Daniel Zoran, Viorica Patraucean, Joao Carreira, Dima Damen, and Andrew Zisserman. Learning from streaming video with orthogonal gradients. In *CVPR*, 2025. 2, 3, 5, 7, 8, 9, 16, 21

[24] Tyler L Hayes, Nathan D Cahill, and Christopher Kanan. Memory efficient experience replay for streaming learning. In *ICRA*, 2019. 3

[25] Tyler L Hayes, Kushal Kafle, Robik Shrestha, Manoj Acharya, and Christopher Kanan. Remind your neural network to prevent catastrophic forgetting. In *ECCV*, 2020. 3

[26] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016. 19

[27] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning. In *CVPR*, 2020. 2

[28] Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. Masked autoencoders are scalable vision learners. In *CVPR*, pages 16000–16009, 2022. 1, 2, 7, 8, 9, 16, 21, 22

[29] Dapeng Hu, Shipeng Yan, Qizhengqiu Lu, Lanqing HONG, Hailin Hu, Yifan Zhang, Zhenguo Li, Xinchao Wang, and Jiashi Feng. How Well Does Self-Supervised Pre-Training Perform with Streaming Data? In *ICLR*, 2022. 3

[30] Gao Huang, Yu Sun, Zhuang Liu, Daniel Sedra, and Kilian Q. Weinberger. Deep Networks with Stochastic Depth. In *ECCV*, 2016. 6

[31] Tommie Kerssies, Daan De Geus, and Gijs Dubbelman. How to benchmark vision foundation models for semantic segmentation? In *CVPRW*, 2024. 7, 22

[32] Ilya Loshchilov and Frank Hutter. Decoupled Weight Decay Regularization. In *ICLR*, 2019. 7, 16, 23

[33] Scott C. Lowe, Anthony Fuller, Sageev Oore, Evan Shelhamer, and Graham W. Taylor. Self-Distillation of Hidden Layers for Self-Supervised Representation Learning. *arXiv:2603.15553*, 2026. 16

[34] Shivani Mall and Joao F Henriques. Cram: Large-scale video continual learning with bootstrapped compression. In *ICCV*, 2025. 3

[35] Daniel Marczak, Bartłomiej Twardowski, Tomasz Trzciński, and Sebastian Cygert. Magmax: Leveraging model merging for seamless continual learning. In *ECCV*, 2024. 9

[36] Ishan Misra, C Lawrence Zitnick, and Martial Hebert. Shuffle and learn: unsupervised learning using temporal order verification. In *ECCV*, 2016. 3

[37] Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy V. Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel HAZIZA, Francisco Massa, Alaaeldin El-Nouby, Mido Assran, Nicolas Ballas, Wojciech Galuba, Russell Howes, Po-Yao Huang, Shang-Wen Li, Ishan Misra, Michael Rabbat, Vasu Sharma, Gabriel Synnaeve, Hu Xu, Herve Jegou, Julien Mairal, Patrick Labatut, Armand Joulin, and Piotr Bojanowski. DINOv2: Learning robust visual features without supervision. *TMLR*, 2024. ISSN 2835-8856. 1, 2, 4, 20

[38] Toby Perrett, Ahmad Darkhalil, Saptarshi Sinha, Omar Emara, Sam Pollard, Kranti Kumar Parida, Kaiting Liu, Prajwal Gatti, Siddhant Bansal, Kevin Flanagan, Jacob Chalk, Zhifan Zhu, Rhodri Guerrier, Fahd Abdelazim, Bin Zhu, Davide Moltisanti, Michael Wray, Hazel Doughty, and Dima Damen. HD-EPIC: A Highly-Detailed Egocentric Video Dataset. In *CVPR*, 2025. 10, 22

[39] Bill Psomas, Dionysios Christopoulos, Eirini Baltzi, Ioannis Kakogeorgiou, Tilemachos Aravanis, Nikos Komodakis, Konstantinos Karantzalos, Yannis Avrithis, and Giorgos Tolias. Attention, please! Revisiting attentive probing through the lens of efficiency. In *ICLR*, 2026. 10

[40] Senthil Purushwalkam, Pedro Morgado, and Abhinav Gupta. The challenges of continuous self-supervised learning. In *ECCV*, 2022. 3

[41] René Ranftl, Alexey Bochkovskiy, and Vladlen Koltun. Vision transformers for dense prediction. In *ICCV*, 2021. 7, 23

[42] Tal Ridnik, Emanuel Ben-Baruch, Asaf Noy, and Lihi Zelnik. ImageNet-21K Pretraining for the Masses. In *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks*, 2021. 1

[43] Karsten Roth, Vishaal Udandarao, Sebastian Dziadzio, Ameya Prabhu, Mehdi Cherti, Oriol Vinyals, Olivier J Henaff, Samuel Albanie, Matthias Bethge, and Zeynep Akata. A Practitioner's Guide to Continual Multimodal Pretraining. In *NeurIPS 2024 Workshop on Scalable Continual Learning for Lifelong Foundation Models*, 2024. 9

[44] Mohammadreza Salehi, Efstratios Gavves, Cees GM Snoek, and Yuki M Asano. Time does tell: Self-supervised time-tuning of dense image representations. In *ICCV*, 2023. 2

[45] Mohammadreza Salehi, Shashanka Venkataramanan, Ioana Simion, Efstratios Gavves, Cees GM Snoek, and Yuki M Asano. MoSiC: Optimal-Transport Motion Trajectory for Dense Self-Supervised Learning. In *ICCV*, 2025. 2

[46] Nathan Silberman, Derek Hoiem, Pushmeet Kohli, and Rob Fergus. Indoor segmentation and support inference from rgbd images. In *ECCV*, 2012. 7, 8, 29

[47] Krishna Kumar Singh, Kayvon Fatahalian, and Alexei A Efros. KrishnaCam: Using a longitudinal, single-person, egocentric dataset for scene understanding tasks. In *WACV*, 2016. 10, 22

[48] Iva Sović, Ivan Martinović, and Marin Oršić. EAST: Early action prediction sampling strategy with token masking. In *ICLR*, 2026. 6

[49] Zhan Tong, Yibing Song, Jue Wang, and Limin Wang. VideoMAE: Masked autoencoders are data-efficient learners for self-supervised video pre-training. *NeurIPS*, 2022. 3

[50] Shashanka Venkataramanan, Mamshad Nayeem Rizve, Joao Carreira, Yuki M Asano, and Yannis Avrithis. Is ImageNet worth 1 video? Learning strong image encoders from 1 long unlabelled video. In *ICLR*, 2024. 2, 3, 4, 5, 19, 29

[51] Jeffrey S Vitter. Random sampling with a reservoir. *ACM Transactions on Mathematical Software (TOMS)*, 1985. 3

[52] Limin Wang, Bingkun Huang, Zhiyu Zhao, Zhan Tong, Yinan He, Yi Wang, Yali Wang, and Yu Qiao. VideoMAE v2: Scaling video masked autoencoders with dual masking. In *CVPR*, 2023. 3

[53] Renhao Wang, Yu Sun, Arnuv Tandon, Yossi Gandelsman, Xinlei Chen, Alexei A. Efros, and Xiaolong Wang. Test-Time Training on Video Streams. *JMLR*, 2025. 3

[54] Xudong Wang, Rohit Girdhar, Stella X Yu, and Ishan Misra. Cut and learn for unsupervised object detection and instance segmentation. In *CVPR*, 2023. 2

[55] Lihe Yang, Bingyi Kang, Zilong Huang, Zhen Zhao, Xiaogang Xu, Jiashi Feng, and Hengshuang Zhao. Depth anything v2. *NeurIPS*, 2024. 2

[56] Lihe Yang, Shang-Wen Li, Yang Li, Xinjie Lei, Dong Wang, Abdelrahman Mohamed, Hengshuang Zhao, and Hu Xu. In Pursuit of Pixel Supervision for Visual Pre-training. *arXiv:2512.15715*, 2025. 16, 23

[57] Yanlai Yang and Mengye Ren. Memory Storyboard: Leveraging Temporal Segmentation for Streaming Self-Supervised Learning from Egocentric Videos. In *Proceedings of The 4th Conference on Lifelong Learning Agents*, 2025. 2, 3, 7, 8, 19

[58] Bolei Zhou, Hang Zhao, Xavier Puig, Sanja Fidler, Adela Barriuso, and Antonio Torralba. Scene parsing through ade20k dataset. In *CVPR*, 2017. 7, 8, 29

[59] Chengxu Zhuang, Ziyu Xiang, Yoon Bai, Xiaoxuan Jia, Nicholas Turk-Browne, Kenneth Norman, James J DiCarlo, and Dan Yamins. How well do unsupervised learning algorithms model human real-time and life-long learning? *NeurIPS*, 2022. 3

A Additional Experiments

A.1 Intra- and Inter-Batch Similarity on `WT++12h`

In Table 2, we show that high inter-batch similarity does not explain why the streaming MAE baseline lags behind i.i.d.-trained MAE. A pre-shuffled ImageNet-1K stream remains close to standard i.i.d. MAE despite its high inter-batch similarity. We repeat this analysis using exactly the same `WT++12h` frames and evaluate both ViT-S/16 and ViT-B/16. We compare standard i.i.d. sampling, a pre-shuffled stream in which frames are shuffled once before applying the sliding-window protocol, and the original chronological stream.

Table 12: Intra-/inter-batch similarity and downstream performance for ViT-S/16 and ViT-B/16 pretrained on `WT++12h`.

Training regime	μ_{intra}	μ_{inter}	ViT-S/16		ViT-B/16	
			ADE20K (mIoU) ↑	Cityscapes (mIoU) ↑	ADE20K (mIoU) ↑	Cityscapes (mIoU) ↑
i.i.d.	0.171	0.738	25.9	63.5	**32.9**	67.8
Pre-shuffled stream	0.171	0.999	**26.0**	**63.9**	32.3	**68.8**
Chronological stream	0.665	1.000	23.9	61.3	26.2	58.2

The key comparison is between the two streaming regimes, which have nearly identical inter-batch similarities of 0.999 and 1.000. Pre-shuffling reduces intra-batch similarity from 0.665 to 0.171 and improves ADE20K and Cityscapes by 2.1 and 2.6 mIoU with ViT-S/16 and by 6.1 and 10.6 mIoU with ViT-B/16. Moreover, i.i.d. sampling and pre-shuffled streaming remain comparable despite their substantially different inter-batch similarities. Together with the ImageNet-1K control, these results support high intra-batch similarity as the main source of degradation in a streaming setting.

A.2 Downstream Evaluation Reliability

To assess downstream evaluation reliability for the MAE-based models in the main comparisons reported in Tables 3 and 4, we repeat ImageNet-1K fine-tuning with two seeds and ADE20K and Cityscapes fine-tuning with three seeds.

Table 13: Results across repeated downstream fine-tuning runs for fixed ViT-S/16 and ViT-B/16 checkpoints pretrained on `WT++12h`. We report mean and standard deviation over two seeds for ImageNet-1K and three seeds for ADE20K and Cityscapes.

Method	ViT-S/16			ViT-B/16		
	IN-1K Acc@1 ↑	ADE20K mIoU ↑	Cityscapes mIoU ↑	IN-1K Acc@1 ↑	ADE20K mIoU ↑	Cityscapes mIoU ↑
MAE – standard i.i.d.	$77.02_{\pm 0.02}$	$25.9_{\pm 0.1}$	$63.4_{\pm 0.7}$	$81.12_{\pm 0.10}$	$32.8_{\pm 0.1}$	$68.1_{\pm 0.4}$
MAE (streaming)	$77.02_{\pm 0.09}$	$24.1_{\pm 0.2}$	$60.6_{\pm 0.6}$	$80.05_{\pm 0.04}$	$26.6_{\pm 0.2}$	$58.9_{\pm 0.5}$
StreamMAE	$77.56_{\pm 0.05}$	$25.8_{\pm 0.3}$	$63.3_{\pm 0.6}$	$81.12_{\pm 0.00}$	$32.9_{\pm 0.6}$	$69.5_{\pm 0.4}$

As shown in Table 13, the improvements over the streaming MAE baseline are larger than the observed fine-tuning variation for both backbones. StreamMAE also matches or exceeds same-data i.i.d. MAE across the evaluated tasks. These evaluations preserve the conclusions of the main comparison.

A.3 Sensitivity to Batch Size

We test how far the temporal extent of each update can be reduced by decreasing the batch size without using DataDrop. ViT-B/16 pretrained on `WT++50h` with subsampling factor $k = 16$ (i.e., `3.75 FPS`), $B = 512$, 256, and 128 correspond to roughly 136, 68, and 34 seconds of video per batch. Table 14 shows that performance degrades gradually, but remains competitive even at $B = 128$, despite each update containing a very short and highly correlated segment of the stream.

Table 14: Effect of reducing batch duration without DataDrop. With ViT-B/16 pretrained on `WT++50h` at 3.75 FPS, smaller batches correspond to shorter and more temporally correlated video windows, yet performance degrades only gradually.

Method	Batch size	`DataDrop`	IN-1K Acc@1 ↑	CS (mIoU) ↑	ADE (mIoU) ↑
StreamMAE (ViT-B)	$B = 512$	✗	81.7	73.6	36.4
	$B = 256$	✗	81.8	73.3	35.4
	$B = 128$	✗	81.4	72.2	34.2

A.4 On Baselines and DataDrop

We use DataDrop for the streaming baselines reported in the main paper because it improves their downstream performance. In this setting, we form a larger local stream window with $B = 2048$, but backpropagate only through a randomly selected 25% subset, giving an effective batch size of 512. As shown in Table 15, using the full $B = 2048$ window without dropping often hurts performance, while DataDrop generally matches or improves the $B = 512$ setting. We therefore report baseline results with DataDrop in Figure 1 and Tables 3 and 4. For the gradient-similarity analysis in Figure 4, we instead use $B = 512$ without DataDrop for all methods, so that the measured gradients are not affected by example dropping.

A.5 Controlling for the Final Video

Since streaming runs use a constant learning rate, a possible concern is that gains from longer streams are driven mainly by the last video seen during training. To test this, we append the same final video, `WT++Budapest`, to shorter streams and compare against the full `WT++50h` stream, where `Budapest` already appears as the last segment.

Table 15: Effect of DataDrop on streaming baselines. We form a local window of $B = 2048$ samples and backpropagate through a randomly selected 25% subset. This generally improves over using the full window and matches or improves the $B = 512$ setting.

Method	Batch size	DataDrop (75%)	IN-1K Acc@1 ↑	CS (mIoU) ↑	ADE (mIoU) ↑
MOCOv3 [10] baseline (ViT-S)	$B = 2048$	✗	66.3	51.7	18.6
	$B = 2048$	✓	68.7	52.0	19.4
MAE [28] baseline w/ Orthogonal-AdamW [23] (ViT-S)	$B = 2048$	✗	75.2	55.3	20.6
	$B = 512$	✗	75.5	58.6	21.5
	$B = 2048$	✓	75.9	58.3	22.5
MAE [28] baseline (ViT-S)	$B = 2048$	✗	76.6	57.9	23.2
	$B = 512$	✗	76.7	60.8	23.2
	$B = 2048$	✓	77.1	61.3	23.9

Table 16: Controlling for the final-video effect. Appending the same final video to shorter streams does not match the full WT++50h stream.

Method	WT duration	IN-1K Acc@1 ↑	CS (mIoU) ↑	ADE (mIoU) ↑
StreamMAE (ViT-B)	WT++12h +Budapest	81.1	69.4	32.5
	WT++25h +Budapest	81.5	72.2	35.2
	WT++50h	81.9	73.8	35.6

As shown in Table 16, appending Budapest does not match training on the full WT++50h stream. This suggests that the gains from longer pretraining are not explained solely by the most recent video, but also depend on the preceding stream.

A.6 Masking Strategy

We also evaluate whether alternative masking strategies improve StreamMAE. Table 17 compares random uniform masking with block masking [56] and masking inspired by I-JEPA [4]/Bootleg [33]. We find that random uniform masking remains the strongest overall choice: block masking performs comparably on ADE20K, but underperforms on ImageNet-1K and Cityscapes, while the I-JEPA/Bootleg-style strategy is consistently worse. We therefore use standard random uniform masking in our experiments.

Table 17: Effect of masking strategy for StreamMAE with ViT-B/16. Standard random uniform masking remains the strongest overall choice.

Method	Masking Strategy	IN-1K Acc@1 ↑	CS (mIoU) ↑	ADE (mIoU) ↑
StreamMAE (ViT-B)	Block Masking [56] (75%)	80.9	68.1	32.8
	I-JEPA [4] / Bootleg [33]	80.4	66.9	31.7
	Random Uniform [28] (75%)	81.1	69.0	32.7

A.7 AdamW Momentum

Following Carreira et al. [6], we evaluate AdamW [32] without first-moment momentum by setting $\beta_1 = 0$. As shown in Table 18, this setting underperforms our default AdamW configuration with $\beta_1 = 0.9$ for StreamMAE with ViT-B. We therefore use $\beta_1 = 0.9$ throughout the paper.

Table 18: Effect of AdamW first-moment momentum for StreamMAE with ViT-B/16. Our standard setting $\beta_1 = 0.9$ performs better than deactivating momentum.

Method	AdamW β_1	IN-1K Acc@1 ↑	CS (mIoU) ↑	ADE (mIoU) ↑
StreamMAE (ViT-B)	0	81.0	67.1	31.8
	0.9	81.1	69.0	32.7

A.8 Additional Ablations

Table 19: Additional ablation of StreamMAE components with ViT-S/16 pretrained on `WT++12h`. The extra rows remove either DataDrop or regularization from the full method to isolate their roles.

Components				Downstream performance		
`DataDrop`	Color Jit. & Drop Path	Two-stage Cropping	Motion-biased Crop Selection	Cityscapes ↑	ADE20K ↑	KITTI ↓
–	–	–	–	60.8	23.2	$4.307_{\pm 0.046}$
✓	–	–	–	61.3	23.9	$4.211_{\pm 0.137}$
✓	✓	–	–	62.6	25.1	$4.025_{\pm 0.039}$
✓	✓	✓	–	62.8	26.1	$4.074_{\pm 0.048}$
✓	–	✓	✓	62.4	24.7	$4.022_{\pm 0.123}$
–	✓	✓	✓	63.1	25.4	$\mathbf{3.958_{\pm 0.087}}$
✓	✓	✓	✓	**63.8**	**26.1**	$3.976_{\pm 0.012}$

Table 19 extends the main ablation with two additional configurations that remove either DataDrop or the combination of color jitter and increased drop path from the full method, while keeping crop selection fixed. Removing color jitter and increased drop path lowers performance, confirming their importance under low within-batch diversity. Removing DataDrop remains competitive on semantic segmentation, suggesting that DataDrop is a useful complementary component but not the main driver of StreamMAE. This is consistent with our scaling experiments, where StreamMAE remains effective even without DataDrop when using larger models and longer streams.

Table 20: Ablation of motion-biased crop selection with ViT-B/16 pretrained on KrishnaCAM. Both variants use the same frames, pretraining budget, and remaining StreamMAE configuration.

Method	ADE (mIoU) ↑	CS (mIoU) ↑	NYUv2 (RMSE) ↓	KITTI (RMSE) ↓
StreamMAE w/o motion-biased selection	33.9	71.5	$\mathbf{0.621_{\pm 0.004}}$	$3.646_{\pm 0.053}$
StreamMAE (ours)	**34.5**	**72.2**	$\mathbf{0.621_{\pm 0.001}}$	$\mathbf{3.583_{\pm 0.012}}$

A.9 Longer i.i.d. ImageNet Reference

In the main benchmarking table (Table 3) the ImageNet-1K i.i.d. MAE reference is iteration-matched to the `WT++12h` streaming runs. Here, we provide an additional reference trained for the same number of iterations as longer `WT++95h` streaming runs. This corresponds to approximately 650k iterations, matching 95 hours of video at 3.75 FPS with stride $s = 2$. This comparison contextualizes the scaled StreamMAE results against a longer i.i.d. ImageNet pretraining budget.

Table 21: Longer i.i.d. ImageNet reference. Unlike the main benchmarking tables, where ImageNet-1K i.i.d. MAE is iteration-matched to `WT++12h`, this table matches the training length of the `WT++95h` streaming runs, approximately 650k iterations. StreamMAE trained on `WT++95h` remains competitive with this longer i.i.d. ImageNet reference, particularly for ViT-S and Cityscapes transfer.

Method	Pretraining data	IN-1K Acc@1 ↑	CS (mIoU) ↑	ADE (mIoU) ↑
Standard i.i.d. MAE (ViT-S)	ImageNet-1K	78.5	65.9	28.5
StreamMAE (ViT-S)	`WT++95h`	78.4	68.0	29.5
Standard i.i.d. MAE (ViT-B)	ImageNet-1K	82.5	75.2	39.8
StreamMAE (ViT-B)	`WT++95h`	82.0	74.0	36.5
StreamMAE (ViT-B, avg. {60,100}%; see Table 7)	`WT++95h`	81.9	75.3	37.3

A.10 Cosine Similarity of Gradients from Consecutive Batches

In Figure 7, we provide a more detailed visualization of the consecutive-gradient similarity analysis from the main paper. For clarity, we plot each MAE-based method separately and show a running average over 100 logged values. Gradient cosine similarity is logged every 10 training steps using the MLP parameters in the last transformer block. While StreamMAE has a mean similarity close to the standard i.i.d. MAE reference, the i.i.d. run exhibits lower variance.

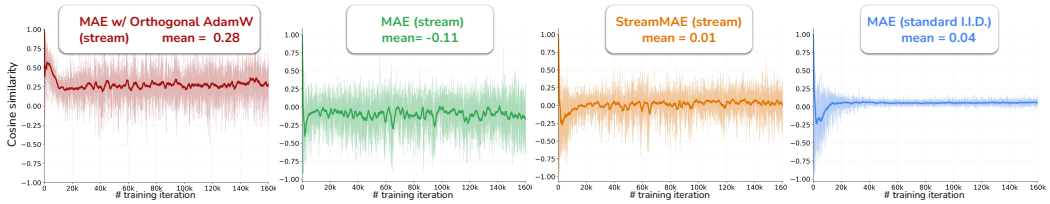

Figure 7: Running average of cosine similarity between gradients from consecutive batches, measured on last-block MLP parameters every 10 steps.

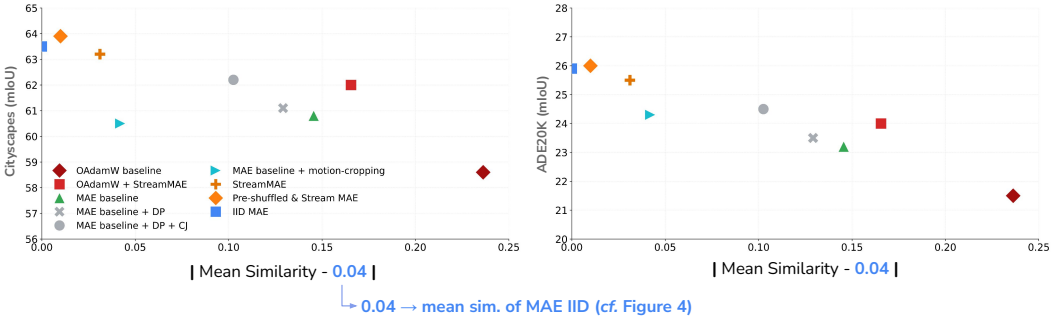

Figure 8: Downstream performance versus the distance from standard i.i.d. MAE in mean gradient cosine similarity between consecutive batches throughout training. Extending Figure 4, we include MAE trained on pre-shuffled & streaming `WT++12h` (Table 12), StreamMAE with Orthogonal-AdamW, and streaming MAE using only motion-biased crop selection. All models use a ViT-S/16 backbone and are pretrained on `WT++12h`. All streaming variants use $B = 512$ without DataDrop.

Figure 8 extends the analysis in Figure 4 with three additional runs: MAE trained on a pre-shuffled `WT++12h` stream (Table 12), StreamMAE with Orthogonal-AdamW, and streaming MAE using only motion-biased crop selection. All models use a ViT-S/16 backbone and are pretrained on `WT++12h`. The streaming variants use $B = 512$ without DataDrop. Across the evaluated configurations, a smaller distance from standard i.i.d. MAE in mean consecutive-batch gradient cosine similarity over training is generally associated with stronger downstream performance on Cityscapes and ADE20K. The only clear exception is streaming MAE with motion-biased crop selection on Cityscapes, which is closer to i.i.d. MAE in gradient similarity but performs similarly to the streaming MAE baseline. Notably, MAE trained on the pre-shuffled `WT++12h` stream is the closest streaming variant to i.i.d. MAE in gradient similarity and achieves comparable downstream performance. This further supports the findings in Tables 2 and 12 that high inter-batch similarity does not explain the degradation observed under streaming pretraining.

B Limitations and Open Challenges

We acknowledge several limitations of our study. First, except for motion-biased crop selection, StreamMAE does not explicitly model temporal dynamics. Extending StreamMAE to architectures or objectives that directly exploit temporal information remains an important direction for future work. Second, StreamMAE relies on fixed frame subsampling to reduce the effective frame rate, motivated by the observation in Table 8 that denser temporal sampling can degrade performance. However, fixed subsampling is likely suboptimal, since the rate of visual change can vary substantially throughout a video, as illustrated in Figure 3. Adaptive temporal sampling may therefore better match the local dynamics of the stream. Finally, we only consider checkpoint averaging at evaluation time, while incorporating long-timescale consolidation directly into streaming pretraining remains unexplored.

Table 22: Construction of the ordered WT++ streams used in scaling experiments. Each stream is a prefix of the next one.

Stream	# Videos	Duration	
WT++12h	1	12.0h	London stream
WT++25h	10	24.8h	WT++12h + 9 videos from original WalkingTours [50]
WT++50h	26	50.0h	WT++25h + 16 appended videos
WT++95h	58	94.5h	Full WT++ collection

C WT++ dataset

We provide additional details on the construction of WT++. The dataset contains 58 urban walking-tour videos with a total duration of 94.5 hours. Videos are originally recorded at 60 FPS, similar to the released WalkingTours dataset. In our experiments, we temporally subsample each stream before training; unless stated otherwise, we use subsampling factor $k = 16$, corresponding to an effective frame rate of 3.75 FPS. For scaling experiments, we construct ordered streams of increasing duration by sequentially appending videos, yielding WT++12h, WT++25h, WT++50h, and WT++95h. Each stream extends the preceding one with new videos while retaining all earlier videos in the same order. Thus, scaling pretraining duration introduces new visual content while preserving the earlier part of the stream.

Table 23: Construction of the ordered WT++ streams. Each row lists the new videos introduced at that stage, in training order (all videos from earlier stages are retained).

Stream stage	# New videos	Cities / videos in training order
WT++12h	1	London
WT++25h	9	Venice, Amsterdam, Singapore, Istanbul, Bangkok, Stockholm, Kuala Lumpur, Zurich, Chiang Mai
WT++50h	16	Frankfurt, Phnom Penh, Sevilla, Hoi An, Prague, Chemnitz, Marmaris, Dubrovnik, Mostar, Gibraltar, Barcelona, Valencia, Rhodes, Helsinki, Copenhagen, Budapest
WT++95h	32	Vilnius, Danang, Klaipeda, Liege, Marbella, Monschau, Naples, Nessebar, Port de Soller, Sarajevo, Arcadia, Timisoara, Warsaw, Malacca, Florence, Cardiff, Georgetown, Luxembourg, Alicante, Ipoh, Belfast, Kaliningrad, Maastricht, Valletta, Vienna, Saigon, Cadiz, Kyiv, Oxford, Sihanoukville, Chongqing, Tokyo

C.1 Dataset release

All videos used in WT++ are publicly accessible online. We will release the dataset package, including metadata, source links, and preprocessing scripts, under CC BY 4.0. For videos already licensed under CC BY 4.0 we plan to redistribute processed copies under their original CC BY 4.0 terms with attribution to the original creators. For the videos currently under the Standard YouTube License, we will release only metadata and source links, or replace them with compatible alternatives.

D *MemoryStoryboard* [57] Results

For the *MemoryStoryboard* baseline, we use the official codebase and replace the default ResNet-50 [26] encoder with ViT-S/16 to match the architecture used in our main comparisons. The encoder is initialized from scratch, and we use the final [CLS] token as the image representation. We train on the WT++London stream with 224×224 crops, a short FIFO memory of 2048 samples, and a long reservoir memory of 16384 samples. Each batch contains 64 current-stream samples and 448 replay samples, giving an effective batch size of 512. We additionally swept the stream stride and reservoir size and evaluated input resolutions of 112×112, as used in the original setup, and 224×224. The reported configuration uses stride $s = 2$, which performed better than $s = 1$.

As *MemoryStoryboard* uses SimSiam [8] and SimCLR [7] objectives, we evaluate both, and tune the learning rate over $\{8 \times 10^{-5}, 10^{-4}, 3 \times 10^{-4}\}$. The best result is obtained with SimCLR and learning rate 10^{-4}, which is the configuration reported in Figure 1 and Table 3. We use AdamW with weight decay 0.05 and enable the default label-merging procedure. We also evaluated the original ResNet-50 recipe on WT++12h using only the most recent 2048 frames, without long-term replay. Under this constrained memory setting, the learned representations showed signs of collapse, so we do not report downstream results for this configuration.

Since *MemoryStoryboard* was originally developed with ResNet-50 backbones and SimSiam/SimCLR objectives, adapting it to ViT-S/16 may require additional architecture-specific tuning. We therefore view this result as a controlled reproduction under our ViT-S/16 streaming evaluation protocol rather than as an exhaustive optimization of the method. In particular, the original *MemoryStoryboard* setup uses larger strides between consecutive batches, which may be an additional direction for optimizing its performance in our setting.

E Additional Training Details

E.1 Inter- and Intra-Batch Similarity Computation

We quantify inter- and intra-batch similarity using frozen DINOv2-L [37] features. All similarities are computed on raw images or frames, without training augmentations. Thus, $\mathbf{X}^{(t)}$ denotes the batch/window of underlying samples from the stream, not the augmented views used during SSL training. For each image or frame $\mathbf{x}_i^{(t)} \in \mathbf{X}^{(t)} = \{\mathbf{x}_1^{(t)}, \ldots, \mathbf{x}_B^{(t)}\}$, we extract the `[CLS]` token representation and ℓ_2-normalize it:

$$\mathbf{z}_i^{(t)} = \frac{f_{\mathrm{DINOv2}}(\mathbf{x}_i^{(t)})}{\|f_{\mathrm{DINOv2}}(\mathbf{x}_i^{(t)})\|_2}.$$

We define *intra-batch similarity* as the mean off-diagonal cosine similarity within a batch:

$$\mu_{\mathrm{intra}}(\mathbf{X}^{(t)}) = \frac{1}{B(B-1)} \sum_{i \neq j} (\mathbf{z}_i^{(t)})^\top \mathbf{z}_j^{(t)} = \frac{2}{B(B-1)} \sum_{i<j} (\mathbf{z}_i^{(t)})^\top \mathbf{z}_j^{(t)}.$$

This measures how similar examples are within a single optimization batch. To measure similarity between consecutive batches, we use a symmetric nearest-neighbor score, since duplicate or near-duplicate samples are known to degrade representation learning [1, 2]. This metric directly estimates whether each example has a highly similar counterpart within the immediately following batch. For two consecutive batches $\mathbf{X}^{(t)}$ and $\mathbf{X}^{(t+1)}$, we first compute the directed similarities:

$$\mu_{t \to t+1} = \frac{1}{B} \sum_{i=1}^{B} \max_{1 \leq j \leq B} (\mathbf{z}_i^{(t)})^\top \mathbf{z}_j^{(t+1)},$$

and

$$\mu_{t+1 \to t} = \frac{1}{B} \sum_{j=1}^{B} \max_{1 \leq i \leq B} (\mathbf{z}_j^{(t+1)})^\top \mathbf{z}_i^{(t)}.$$

The *inter-batch similarity* is the average of the two directions:

$$\mu_{\mathrm{inter}}(\mathbf{X}^{(t)}, \mathbf{X}^{(t+1)}) = \frac{1}{2} \left(\mu_{t \to t+1} + \mu_{t+1 \to t} \right).$$

This captures whether examples in one batch have close matches in the next batch.

For the reported values, we average μ_{intra} over sampled batches and μ_{inter} over consecutive batch pairs. For pre-shuffled and streaming ImageNet-1K, we use batches of size $B = 512$ with stride $s = 8$, matching the corresponding experiments, and report the mean over three stream starting points, each containing 100 consecutive batches. For `WT++London`, we use $B = 512$ and stride $s = 1$, matching our streaming setting, and again average over three random starting points with 100 consecutive batches each. For standard i.i.d. ImageNet-1K, each batch is sampled independently from the full training set, and here we report the mean over 50 batches, again using batch size $B = 512$.

E.2 Two-Stage Cropping and Motion-Biased Crop Selection

Videos are originally resized to 1280×720 resolution. For two-stage cropping, the first-stage crop size is 592×336, approximately preserving the 16:9 aspect ratio while remaining aligned with the ViT patch grid. The final MAE random resized crop is then applied within this first-stage region. For motion-biased crop selection, we cache frame-difference scores at the ViT patch level. Since the ViT patch size is 16, each 1280×720 frame yields a compact 80×45 patch-level difference map.

This cache is inexpensive to store and avoids recomputing frame differences or reloading frames when selecting motion-biased crops. First-stage crop candidates are constrained to be aligned with the ViT patch grid, so their motion scores can be computed directly from the cached patch-level differences. When DataDrop is used, patch-level differences are still computed for the local stream window before example dropping. Thus, crop selection can use the same cached difference maps, while only the retained examples contribute to the training loss.

E.3 Training i.i.d. Baselines

The i.i.d. MAE baselines on ImageNet-1K and `WT++London`, reported in Tables 2 to 4, are trained with an iteration-matched protocol. That is, each i.i.d. baseline is trained for the same number of optimization steps as the corresponding streaming run on `WT++London`. These baselines are therefore not intended to reproduce fully optimized MAE pretraining schedules. They provide controlled references at the same training budget as our streaming experiments.

By i.i.d. training, we mean that each optimization step samples a batch uniformly at random from the full dataset, without preserving temporal order or using sliding-window batches. We train these i.i.d. baselines with the standard MAE cosine learning-rate schedule, whereas streaming runs use a constant learning rate with linear warm-up of 5% of total number of iterations. Unless stated otherwise, the batch size is 512. For IN-1K ViT-B/16 i.i.d. experiments, we use base learning rate 8×10^{-5} with linear batch-size scaling. For ViT-S/16, we use base learning rate 3.2×10^{-4}, which performed better.

The fixed-order pre-shuffled ImageNet-1K experiment in Table 2 is also iteration-matched. We pre-shuffle ImageNet-1K once, consume the resulting sequence as a fixed stream with batch size $B = 512$ and stride $s = 8$, and train for approximately the same number of steps as the corresponding i.i.d. baseline.

E.4 SSL Baselines

We reproduce three representative SSL baselines in our streaming video setup: DINO [5], MAE [28], and MoCo v3 [10]. All methods are implemented in solo-learn [13] and trained from scratch on the `WT++12h` stream using the same single-traversal sliding window protocol. Unless stated otherwise, batches are formed in temporal order with stride $s = 1$, and baselines use a local stream window of size $B = 2048$ with DataDrop, yielding an effective batch size of 512. Training uses mixed precision, synchronized batch normalization when required by the method, and a constant learning rate with linear warm-up.

DINO [5]. We train DINO with a ViT-S/16 backbone, AdamW, learning rate 10^{-4}, weight decay 0.04, and teacher momentum increased from 0.996 to 0.9996 with a cosine schedule. We use 8192 prototypes, projection hidden dimension 2048, projection output dimension 256, student temperature 0.1, and teacher temperature 0.04. The augmentation pipeline follows the multi-crop recipe with two global 224×224 crops and four local 96×96 crops. Increasing the number of prototypes or local crops did not lead to consistent improvements.

MAE [28]. We train MAE with a ViT-S/16 backbone, AdamW, learning rate 8×10^{-5}, weight decay 0.05, and AdamW betas $(0.9, 0.95)$. We use the standard masking ratio of 0.75, normalized pixel loss, and a decoder with embedding dimension 512, depth 8, and 16 attention heads. The MAE baseline uses a single 224×224 reconstruction crop with random resized cropping and horizontal flipping, without color jitter or blur. We also evaluate MAE with Orthogonal-AdamW [23] using the same configuration.

MoCo v3 [10]. We train MoCo v3 with a ViT-S/16 backbone, AdamW, learning rate 10^{-4}, weight decay 0.1, and teacher momentum increased from 0.996 to 0.9996 with a cosine schedule. The projection and prediction hidden dimensions are 4096, the projection output dimension is 256, and the contrastive temperature is 0.2. We use the standard asymmetric two-crop MoCo v3 augmentation pipeline with two 224×224 crops, color jitter, grayscale augmentation, Gaussian blur, and solarization. Increasing the effective batch size (i.e., removing DataDrop) did not improve MoCo v3 in our streaming setup (please see Table 15). For StreamMAE, we follow the standard MoCo v3 color-jitter setting: brightness 0.4, contrast 0.4, saturation 0.2, and hue 0.1, and apply it with probability 0.5.

Streaming-specific adaptations. We did not assume a priori that MAE would be the strongest objective. We swept the baselines over learning rate, batch size, teacher momentum, crop configuration, and prototype count where applicable. We also tested several adaptations designed specifically for high temporal redundancy.

For **MoCo v3**, we tested multiple temporal positives, temporally soft targets, slow and fast EMA teachers with different momentum values, and periodic checkpoint merging and reinitialization. None of these variants resolved the optimization instability or prevented downstream transfer from degrading as pretraining progressed.

For **DINO**, we tested slow and fast EMA teachers, slow- and fast-evolving prototypes, updates restricted to hard examples, and Sinkhorn-Knopp assignment after observing low prototype utilization. None of these variants consistently improved downstream transfer.

We also tested Orthogonal-AdamW with MAE, MoCo v3, and DINO without consistent gains. DataDrop is applied to all streaming baselines in the main comparison because it matches or improves their corresponding no-drop configurations (see Table 15).

E.5 Additional Pretraining Datasets

We construct three additional ordered pretraining streams from HD-EPIC [38], CROWD [3], and KrishnaCAM [47] (see Tab. 11). For each dataset, we preserve the temporal order within individual videos and concatenate the selected videos into a single fixed stream.

HD-EPIC [38] contains egocentric video recorded entirely in kitchens. We use recordings from four kitchens, P01, P02, P06, and P08, preserving the provided chronological order. The constituent videos average approximately 15 minutes and form an 18-hour stream. We temporally sample the stream at 2.5 FPS, yielding approximately 160k pretraining iterations.

CROWD [3] contains front-facing urban dashcam video. We use 16 videos recorded in Edinburgh, Salt Lake City, Jinan, Da Nang, Quito, and Gold Coast. The videos average approximately 53.5 minutes and form a 14.5-hour stream. We temporally sample the stream at 4 FPS, yielding approximately 203k pretraining iterations.

KrishnaCAM [47] is a 70-hour longitudinal egocentric dataset covering indoor and outdoor daily life. Its constituent videos average approximately 9.5 minutes and are concatenated chronologically. We temporally sample the resulting stream at 1 FPS, yielding approximately 250k pretraining iterations. For KrishnaCAM, StreamMAE uses $K = 2$ candidate crops to reduce repeated selection of the same regions during stationary activities.

F Additional Evaluation Details

F.1 Image Classification

For ImageNet-1K [15] classification, we follow the MAE [28] fine-tuning protocol. We fine-tune ViT-S/16 and ViT-B/16 end-to-end at resolution 224×224, initializing the backbone from the pretrained checkpoint and training a new linear classifier. For MAE-based models, we use global average pooling over patch tokens followed by normalization and a linear classifier; for DINO and MoCo v3, we use the final `[CLS]` token representation. We fine-tune for 100 epochs with AdamW, cosine learning-rate decay, 5 warmup epochs, weight decay 0.05, layer-wise learning-rate decay 0.65, drop-path rate 0.1, mixup 0.8, cutmix 1.0, and random erasing probability 0.25. We use the learning-rate scaling rule $\mathrm{lr} = \mathrm{blr} \cdot B_{\mathrm{eff}}/256$, with $\mathrm{blr} = 5 \times 10^{-4}$ and global batch size $B_{\mathrm{eff}} = 1024$. At evaluation time, images are resized to 256 pixels on the shorter side and center-cropped to 224×224. We report top-1 accuracy on the ImageNet-1K validation set.

F.2 Semantic Segmentation

For semantic segmentation, we follow the protocol of Kerssies et al. [31]. We evaluate ViT-S/16 and ViT-B/16 on ADE20K and Cityscapes by attaching a linear per-patch prediction head to the ViT encoder and fine-tuning the full model end-to-end. For pretrained checkpoints, only the encoder is initialized from the self-supervised weights; the segmentation head is initialized randomly.

We use AdamW [32] with polynomial learning-rate decay with power 0.9. The segmentation head uses learning rate 10^{-4}. For pretrained models, the encoder learning rate is scaled by 0.5, while randomly initialized models use the same learning rate for encoder and head. We train for 40k steps on ADE20K and 20k steps on Cityscapes with batch size 16 and mixed precision.

Training augmentations include random horizontal flipping, color jitter, scale jitter in $[0.5, 2.0]$, padding when needed, and random cropping. Crop sizes are 512×512 for ADE20K and 1024×1024 for Cityscapes. Positional embeddings are interpolated to the target patch grid. At evaluation time, we use sliding-window inference with the same crop sizes and average logits in overlapping regions. We report validation mIoU.

F.3 Depth estimation

For depth estimation, we follow the evaluation protocol used by Yang et al. [56], but perform full-finetuning instead of evaluating frozen encoders. Hence, we set the encoder learning rate to half that of the DPT [41] depth head. The prediction head takes normalized patch tokens and CLS tokens from four intermediate encoder layers as input, concatenated along the channel dimension. We set the minimum depth to 0.001 for both datasets and the maximum depth to 10 and 80 for NYUv2 and KITTI, respectively. We train using the scale-invariant logarithmic loss (SiLog loss) for 60 epochs with a batch size of 32, a learning rate of 1×10^{-4} for the depth head, and a weight decay of 0.01. We use linear warmup for the learning rate in the first 10% of training. During training, we use a crop size of 256×256. During validation, we employ sliding-window inference with a stride of $\frac{2}{3}$ of the crop size, corresponding to 170 pixels, and average predictions in overlapping regions. For depth estimation, the reported metrics are mean $\pm$ standard deviation over two fine-tuning runs with different seeds.

G Compute Resources

We report approximate compute requirements in Table 24. All runs were performed on NVIDIA H100 GPUs with mixed-precision training. Wall-clock times for pretraining exclude downstream fine-tuning. The full research project required additional compute for preliminary experiments, hyperparameter sweeps, and failed runs that are not included in the table.

For StreamMAE, a single H100 is sufficient, but we typically used $2\times$H100 for `WT++12h` and $4\times$H100 for longer streams to reduce turnaround time. On `WT++12h`, StreamMAE takes approximately 7 hours with ViT-S/16 and 8 hours with ViT-B/16 on $2\times$H100, and scales roughly with pretraining stream duration. MoCo v3 and DINO on `WT++12h` take approximately 18 and 25 hours, respectively, on $2\times$H100. For these runs, DataDrop is enabled with $B = 2048$, retaining 25% of samples.

Downstream full fine-tuning on ImageNet-1K takes approximately 4 hours for ViT-S/16 and 5 hours for ViT-B/16 on $4\times$H100. Semantic segmentation fine-tuning takes approximately 2 hours for ViT-S/16 and 2.5 hours for ViT-B/16 on a single H100. Depth fine-tuning takes approximately 3 hours on KITTI and 1.5 hours on NYUv2; reported depth metrics are mean $\pm$ standard deviation over two fine-tuning runs with different seeds.

Table 24: Approximate compute requirements. Pretraining times are reported for `WT++12h` and exclude downstream fine-tuning.

Experiment	Backbone	GPUs	Wall-clock time
StreamMAE pretraining on `WT++12h`	ViT-S/16	$2\times$H100	$\sim$7h
StreamMAE pretraining on `WT++12h`	ViT-B/16	$2\times$H100	$\sim$8h
MoCo v3 pretraining on `WT++12h`	ViT-S/16	$2\times$H100	$\sim$18h
DINO pretraining on `WT++12h`	ViT-S/16	$2\times$H100	$\sim$25h
ImageNet-1K fine-tuning	ViT-S/16	$4\times$H100	$\sim$4h
ImageNet-1K fine-tuning	ViT-B/16	$4\times$H100	$\sim$5h
Semantic segmentation fine-tuning	ViT-S/16	$1\times$H100	$\sim$2h
Semantic segmentation fine-tuning	ViT-B/16	$1\times$H100	$\sim$2.5h
KITTI depth fine-tuning	ViT-S/B	$1\times$H100	$\sim$3h
NYUv2 depth fine-tuning	ViT-S/B	$1\times$H100	$\sim$1.5h

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and introduction state the paper's main scope and claims: we study from-scratch SSL on ordered video streams without global shuffling, multi-epoch replay, or large replay buffers; introduce `WT++` (Section 3.2); analyze why streaming MAE lags behind standard i.i.d. MAE (Section 4); and propose StreamMAE. The claims are supported by the experiments: StreamMAE improves over streaming baselines (Table 3), narrows the gap to standard i.i.d. MAE on the same video data, and scales with longer streams (Table 5). We also qualify the analysis by showing that inter-batch similarity, i.e., fixed-order sliding-window consumption across batches, is not the main issue by itself; rather, high intra-batch similarity, i.e., low within-batch diversity in video batches, is the main challenge.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: We discuss limitations and future directions in Appendix B, including the lack of explicit temporal modeling, the use of fixed temporal subsampling, and the limited exploration of long-timescale model consolidation during streaming pretraining.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.

- While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [N/A].

   Justification: Paper does not include theoretical results.

   Guidelines:

   - The answer [N/A] means that the paper does not include theoretical results.
   - All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
   - All assumptions should be clearly stated or referenced in the statement of any theorems.
   - The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
   - Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
   - Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: We disclose the main implementation details in Section 5, including architectures, optimizer settings, batch construction, stream stride, DataDrop, masking, and crop-selection details. Downstream evaluation protocols are described in Appendix F, and additional baseline and training details are provided in Appendix E. We will also release source code, configuration files, preprocessing scripts, and checkpoints for training and evaluation.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.

(b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
(c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
(d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: We will release the source code. The release will include training and evaluation scripts, configuration files, environment instructions, dataset metadata, preprocessing scripts, and stream definitions needed to reproduce the main results. Dataset licensing and release details are provided in Section C.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: We provide the main experimental setup and training details in Section 5. Additional details on `WT++` construction and splits, baseline reproduction, pretraining hyperparameters, and downstream evaluation protocols are provided in Appendices C, D, E, and F.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [No]

   Justification: The experiments are computationally expensive because each result requires self-supervised pretraining followed by full fine-tuning. For classification and semantic segmentation, we therefore primarily report single fine-tuning runs. For the main MAE-based comparisons, Appendix A.2 additionally reports means and standard deviations over two classification and three segmentation fine-tuning seeds. For depth estimation, we report means and standard deviations over two fine-tuning seeds.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).
   - It should be clear whether the error bar is the standard deviation or the standard error of the mean.
   - It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
   - For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
   - If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: We report compute resources in Appendix G, including GPU type, number of GPUs, and approximate wall-clock time for the main pretraining and downstream fine-tuning runs. We also state that the full research project required additional compute for preliminary experiments, hyperparameter sweeps, and failed runs not included in the reported tables.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

Answer: [Yes]

Justification: We have reviewed the NeurIPS Code of Ethics and believe the research conforms to it. The work is foundational research on self-supervised visual representation learning. Dataset construction, licensing, and release details are documented in Section C. We release only assets whose licenses permit redistribution or for which permission is obtained, and otherwise provide metadata/source links or replace the videos.

Guidelines:

- The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
- If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
- The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

Answer: [Yes]

Justification: The work is foundational research on self-supervised visual representation learning and is not tied to a deployed application. Potential positive impacts include reducing reliance on manually curated datasets and supporting more efficient learning from naturally collected video streams. As with other visual representation learning methods, downstream use should consider privacy and responsible deployment norms, especially for video data. We mitigate dataset-release concerns by documenting licenses and releasing only permitted assets or metadata/source links as described in Section C.

Guidelines:

- The answer [N/A] means that there is no societal impact of the work performed.
- If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

Answer: [N/A]

Justification: The paper does not release high-risk generative models. WT++ is released for research on representation learning, with licensing and release details documented in Section C.

Guidelines:

- The answer [N/A] means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

Answer: [Yes]

Justification: We cite the original papers and codebases for all existing assets used in the paper and follow their respective licenses and terms of use. This includes Cityscapes [12] under its custom non-commercial research license; ADE20K [58], whose images follow non-commercial terms and whose annotations/code are released under BSD 3-Clause; NYUv2 [46], used for academic research; KITTI [19] under CC BY-NC-SA 3.0; and ImageNet-1K [15] under custom non-commercial research terms, with images retaining third-party copyrights. We also use the released WalkingTours dataset [50] under the license specified in its original release, and solo-learn [13] under the MIT license. For WT++, license details and release terms are documented in Section C.

Guidelines:

- The answer [N/A] means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

Answer: [Yes]

Justification: Please see Section C for details on WT++ construction, release plan, and licensing. We document the video sources, stream definitions, preprocessing procedure, and license status. We will release the dataset package, source code, and checkpoints.

Guidelines:

- The answer [N/A] means that the paper does not release new assets.

- Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
- The paper should discuss whether and how consent was obtained from people whose asset is used.
- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: The paper does not involve crowdsourcing or research with human subjects. Some source videos may contain pedestrians in public spaces, but we do not collect participant data, interact with individuals, or conduct human-subject experiments.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: The paper does not involve crowdsourcing or research with human subjects, so IRB approval is not applicable. Some source videos may contain pedestrians in public spaces, but we do not collect participant data, interact with individuals, or conduct human-subject experiments.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [N/A]

Justification: the core method development in this research does not involve LLMs as any important, original, or non-standard components.

Guidelines:

- The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.