# Forget, Anticipate and Adapt: Test Time Training for Long Videos

Rajat Modi[1], Sebastian Noel[1], Xin Liang[1], and Yogesh S. Rawat[1]

Institute of Artificial Intelligence, University of Central Florida
rajatmodi62@gmail.com, yogesh@crcv.ucf.edu
https://rajatmodi62.github.io/2025/07/10/ffn/

**Abstract.** Test Time Training (TTT) is a mechanism in which a model adapts to an incoming test-sample by performing some self-supervised (SSL) task and updating its weights even during inference. This procedure does not require labels at test-time. This paper focuses on TTT for long-videos. A major concern with existing approaches is: 1) they perform TTT updates using a sliding window containing frames in the past, whose compute increases linearly with the size of window. This becomes computationally intractable when the videos are hours long. 2) TTT is performed even when temporally close frames look similar, thereby consuming a lot of compute.

We present the Frame Forgetting Network (FFN) that: 1) operates on only three frames within the sliding window, namely the frame that exits, the current frame and the frame after that. The model still manages to retain temporal context and work for hours long-videos; 2) mathematically define a 'surprise' metric: how much 'new information' the incoming frame contains with respect to the past seen frame. This facilitates determining how to modify the effective window size during TTT and constitutes the core mechanism of an adaptive windowing algorithm. Additionally, we curate a dataset EpicTours containing up to 3 hour long videos of walking city-tours, whereas earlier datasets on this problem were only 5 min long. We demonstrate FFN's empirical effectiveness on dense-segmentation, video classification tasks, generalization to depth-estimation, and multi-hour long videos. The project page can be found at https://github.com/rajatmodi62/ffn.

## 1 Introduction

Typically, machine learning models *only* update their weights during training, but *freeze* them during inference [1, 2]. However, as humans, we possess the ability to *continuously* learn and *adapt* to our ever-changing environment. Test Time Training (TTT) shows the promise of bringing a similar adaptive capability to artificial intelligence, allowing a model to continuously learn and update its weights *even while testing*, entirely without the need for ground-truth labels.

However, applying TTT to *video processing* presents a unique set of challenges. Existing online distillation methods are based on a student-teacher setup [3], with the teacher usually deployed on a remote server and the student on a local device. However, this is a bottleneck in *offline* real-world scenarios, such as *disaster-prone regions*, where server connectivity may be limited.

Consequently, the key challenge lies in determining how to perform TTT on videos locally, on-device, and computationally cheaply. This is especially critical for dense computer vision tasks, such as segmentation, where each pixel matters, and should ideally be achieved utilizing a *single* model.

Historically, there have been two approaches towards adaptation: either (i) train a *dedicated* video model specifically for the task, or (ii) start from a pre-trained image model and adapt it accordingly. Unfortunately, large-scale pre-training datasets for video remain *limited*, and efforts to repurpose image models are inherently constrained by the absence of explicit temporal information. Furthermore, existing TTT methods for videos rely on a *sliding window approach* [4]. A sliding window serves as a *fixed-size temporal buffer* that holds a defined number of preceding frames and moves forward *step by step* as the video advances. Because the model re-evaluates all frames currently inside this window, it performs a *duplicate*, highly expensive computation at every *single* timestep. This *redundant* computation is far *too prohibitive* for long videos, thereby limiting practical deployment

To overcome these limitations, we propose the Frame Forgetting Network (FFN). Our approach is built on the core insight that as the sliding window progresses, only one new frame enters, and one frame exits. Thus, a model *only needs to pay* the computational cost of processing these crucial frames *instead of* recalculating the entire window for every time-step.

Our FFN consists of two core components: a Memory Restoration Mechanism (MRM) and an Adaptive Window Algorithm (AWA). The MRM allows the model to actively 'forget' its adaptation on past frames *exiting* the window and 'restore' the model's original predicted features using a temporal module. Next, the AWA algorithm can dynamically 'anticipate' when adaptation is required, improving the management of computational load. Through extensive experiments across 11 datasets, we validate FFN's *competitive performance* on dense segmentation tasks, *strong generalization* on depth estimation, and ability to learn *stable representations* during recurrent video-processing.

## 2 Frame Forgetting Network

First, we shall talk about our problem statement, cover preliminaries on TTT, and then discuss our Frame Forgetting Network (FFN).

### 2.1 Problem statement

Given a *streaming* long-video $\mathcal{V} = \{x_1, x_2, x_t, ..., x_n\}$, a model can access only the past frames $x_1, x_2, ..., x_{t-1}$, the current frame $x_t$, but *not* the future. The problem is focused on adaptation: for $x_t$, how does the model *decide* when to 'adapt' or not. This adaptation should help improve downstream performance and be cheaper to compute. In this work, we focus on adaptation using TTT, an instantiation that allows a model to even update its weights during inference.

### 2.2 Preliminaries

TTT involves two phases, a) Training Time Training and b) Test Time Training.

**Training Time Training**: Fig1(i) illustrates how a TTT setup works. We consider a model containing backbone $f$, a downstream head $h$ and a self-supervised (SSL) head $g$ specializing in some task that does not require external labels (e.g., image reconstruction). First, we train $(f, g, h)$ *jointly* in a train set, which consists of densely-annotated images. This first phase is known as *training-time-training* [4].

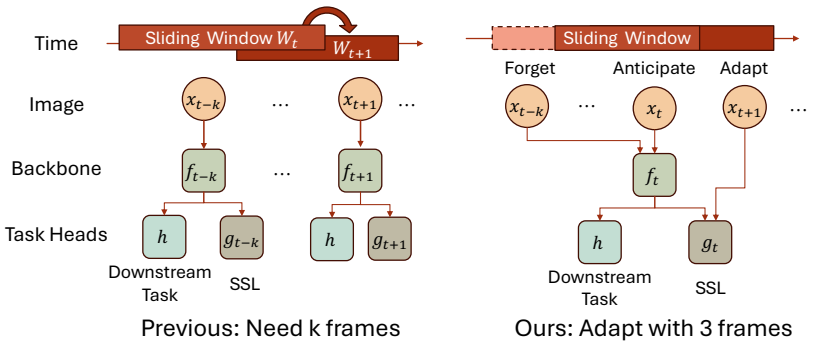


**Fig. 1:** (i) Test Time Training setup: Here $f$ is image-backbone, $g$ is SSL head, $h$ downstream head. Learning is self-supervised, where input test sample $x_t$ is transformed into $x_{t'}$ and compared again with $x_t$. (ii) TTT on videos relies on sliding windows, we denote two such windows $W_t$, $W_{t+1}$ . Notice that they contain a lot of overlapping frames, requiring N operations everytime. (iii) (Ours) We can perform TTT by gradually operating on only 3 frames via a 'forget, anticipate, adapt' procedure.

**Test Time Training**: Here, the model is given access to a test-video consisting of several frames. The model looks at an incoming frame $x_t$ and tries to reconstruct it. As in Fig1, this involves feed-forwarding $x_t$ through the backbone $f$, then through the SSL-head $g$, thus computing the RGB reconstruction $x'_t$. Next, we estimate the reconstruction loss $\ell_s(x'_t, x_t)$, where $x'_t = g \circ f(x_t)$. This may be used to update $f, g$ over several iterations. The downstream head $h$ is *kept frozen* throughout. This is known as *test-time-training*.

### 2.3 The Principle of locality


**Fig. 2: The Principle of Locality**: The first 3 frames (outdoors, marked in red) may not be directly relevant to last 3 frames (indoors, marked in green), therefore during TTT, our model neglects them. Best viewed in color.

Adaptation in videos follows a 'principle of locality' [4]. Intuitively, a frame at $t = 1$ (say indoors) *may not* be relevant to the frame at $t = 5k$ (say outdoors). Consider a current frame $x_t$. Instead of looking at the *entire past timesteps*, existing TTT methods use a sliding window of size $k$, which we denote by $W_t$. More formally, the TTT update rule at $x_t$ is governed by:

$$f_t, g_t = \arg\min_{f,g} \frac{1}{k} \sum_{t'=t-k+1}^{t} \ell_s(g \circ f(x_t), x_t), \tag{1}$$

where $k$ is the size of a window containing frames encountered in the past (illustrated by window $W_t$ in Fig1). This creates two issues:

1. The window $W_t$ sums up the $k$ past frames for *each* timestep. Inductively, $W_t$ slides over time (e.g., $W_{t+1}$) with the model's weights being reset after every update. The *arg min* operator requires backpropagation through $f, g$ every timestep $t$. Thus, TTT quickly becomes computationally intractable, with a $2hr$ video (7200 frames) taking up to $8hrs$.
2. The model wastes a lot of compute; for example, it shall perform TTT even when two successive frames are pixel-wise *identical*.

Note that $f, g$ can change their weights over time as the model undergoes adaptation. We denote this by the subscript $t$, namely $f_t$, and $g_t$.

### 2.4 Method

**The sliding window-invariant**: Consider a sliding window $W_t = [x_{t-k}, x_{t-k+1}, ..., x_{t-1}, x_t]$, and the next window $W_{t+1} = [x_{t-k+1}, ..., x_t, x_{t+1}]$. Mathematically, this reveals that *both* have *the same* number of frames *except* for two: frame numbered $x_{t-k}$ that *exits* the window $W_{t+1}$ and frame numbered $x_{t+1}$ that *enters* the window $W_{t+1}$.

The intuition that operating on sliding windows requires a mechanism to update a running state and *only* handle elements which come in/go out is well-known in data structures like arrays (e.g., Kadane's Algorithm [5]). However, inducing

such behavior is *non-trivial* in neural nets, which in-principle may be treated as forms of parallel distributed memories [6].

This reveals that while making a transition from $W_k \rightarrow W_{k+1}$, we *don't* need to process $k$ frames. Rather, we can operate on *only* 3 frames by defining three mechanisms:

1. *Forget:* for frame $x_{t-k}$, 'forget' the model's features *after* adaptation, and 'restore' the model's predicted features to those *prior* to TTT.
2. *Anticipate:* The current frame $x_t$ under processing can be used to 'anticipate' what shall come next at $x_{t+1}$. For brevity, we call this prediction $x'_{t+1}$
3. *Adapt:* Compare $x'_{t+1}$ with the actual frame $x_{t+1}$. If the difference is greater than some threshold $\tau$, the model chooses to adapt on $x_t$. Otherwise, it can perform simple inference on $x_t$ and move on. Note that inference is equivalent to the feed-forward $f \circ h(x_t)$ through the model.

An initial approach might be to hard-code $\tau$ to some value (say 0.5). However, this makes the model's behavior *static*: there *might be* some consecutive frames who have large pixel-difference, but do not deserve adaptation (for e.g., slight rotation of images of the same object). Thus, we need a mechanism that could govern this $\tau$ dynamically.

Inspired by this, we present the Frame Forgetting Network (FFN). It consists of two components: 1) Memory Restoration Mechanism (MRM) 2) Adaptive Window Algorithm (AWA). MRM allows the model to 'forget' the frame that moves out of the window ($x_{t-k}$), while AWA allows the model to *anticipate* if $x_t$ deserves adaptation, by dynamically estimating $\tau$. Next, we explain these blocks.

### 2.5 The Memory Restoration Mechanism (MRM)

To forget the model's adaptation on $x_{t-k}$, we can first *cache* its features before adaptation; i.e., we estimate $f_{t-k-1}(x_{t-k})$. The idea is that the *current* weights of the model $f_t$ should be adjusted so that it can predict the earlier features $f_{t-k-1}(x_{t-k})$. This suggests that the model $f$ requires two things: i) input $x_{t-k}$, and ii) a sense of time $t-k$. We propose instilling this via a **temporal module**.

We implement a temporal module as a three layered MLP[1]. First, the timestep $t-k$ corresponding to $x_{t-k}$ is injected into the temporal module. Instead of conditioning the temporal module by a single integer, we implement $t$ as a one-dimensional positional encoding, similar to the one used in transformers [8]. Note that injecting time $t$ in an image-based backbone $f$ may be cheaper than training a video-backbone from scratch. The output of temporal module is then used to condition the backbone $f_{t-k-1}$, resulting in two input arguments $x_{t-k}, t-k$. Mathematically, we can minimize the following:

[1] This is because neural fields work well with coordinate information [7]

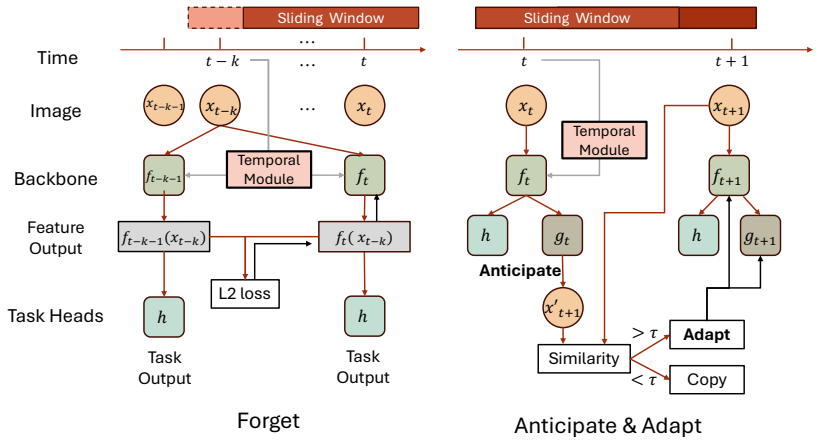


**Fig. 3: Frame Forgetting Network**(i) Forget step: Backbone $f_t$ takes the frame $x_{t-k}$ to forget as input, along with timestep $t-k$, to get the feature encoding, $f_t(x_{t-k})$. To *forget* its adaptation on $x_{t-k}$, backbone is trained with pre-adapted features $f_{t-k-1}(x_{t-k})$ via L2 loss. (ii) Anticipate and Adapt step: Given frame $x_t$ model predicts the next frame $x'_{t+1}$. This is compared with actual frame $x_{t+1}$ to make estimate whether to adapt or not.

$$\ell = f_t(x_{t-k}, t-k) - f_{t-k-1}(x_{t-k}, t-k) \tag{2}$$

This can be used to make a single gradient step through backpropagation and adjust the weights of the backbone $f$.

### 2.6 Adaptive Window Algorithm (AWA)

**An Anticipative Head:** Given the current frame $x_t$, we need to *anticipate* whether adaptation is required. The idea is to allow the model to take $x_t$ as input, predict the next frame $x'_{t+1}$, and compare with the real frame $x_{t+1}$. The average of pixel-wise difference can then serve as an indicator if adaptation is required[2]. Thus, we define the visual space difference as

$$v_{visual}(t) = \frac{2}{\sqrt{h \times w}} \sum_{h,w} \frac{\|x_{t+1} - x'_{t+1}\|_2}{255} \tag{3}$$

One concern about equation 3 is that *if* an incoming frame $x_{t+1}$ just 'rotates' slightly over the current frame $x_t$, $v_{visual}$ shall be very high owing to the fluctuations in the *pixel* space (for e.g., capturing a video over phone). Intuitively, however, these two frames still look almost the same. This means that *low-level* information *might* not be a reliable indicator.

[2] This anticipative-head may be learned with as few as 1 long-video during the training phase of TTT. We also experimented with the *k-step* future-frame predictor [9, 10], but found the *next* timestep predictor to work quite well (similar to classical auto-regression in language models)

Thus, we make use of the following observation: *higher* layers of a model $f$ remain *invariant* to minor-changes in input frames. Inspired by this, we define a memory buffer $B = [v_{latent}(t_1), v_{latent}(t_2), ..., v_{latent}(t_{last})]$, where $v_{latent}(t_i) = f_{t_i}(x_{t_i})$. Note that different $t_i$ here denote the timesteps when the model actually 'adapted' in the past, and *need not* be consecutive. The idea is that if the *latent* feature of the *most recently adapted* past frame ($t_{last}$) is very similar to the latent feature of the current frame $x_t$, we must *not* adapt. Please note that $t_{last}$ need *not* be equal to $t-1$. We define an anticipation matrix A as:

$$A = \frac{v_{latent}(t_i) \cdot v_{latent}(t_{last})}{\|v_{latent}(t_i)\|_2 \|v_{latent}(t_{last})\|_2} \tag{4}$$

**The surprise metric**: Both equations 3 and 4 can be unified together to define a metric that measures how much new information (surprise $S$) a frame $x_t$ provides to the model. We define $S$ as:

$$S_t = [\log(1 + v_{\text{visual}}(t))] \times [1 - A] \tag{5}$$

Next, we consider three cases: (a) when video contains static frames (e.g., CCTV camera feed): Here $v_{visual} \approx 0, A \approx 1$, so surprise $S_t \approx 0$; (b) when video contains a shaking camera (e.g., person captures a video with a head-mounted display): Here the scene remains the same, but frames change a lot. Therefore $v_{visual} \uparrow$, $A \approx$ *constant*, so surprise $S_t$ *stays low* ; (c) jump cuts, where the scene changes suddenly (say a transition from indoors to outdoors). There $v_{visual} \uparrow$, $A \uparrow$, however $v_{visual}$ increases *far more* than $A$[3], so surprise $S_t$ takes on a *very high* value.

Both cases b) and c) suggest that if $S_t$ exceeds or equals to *some adaptive threshold* $\tau_t$, the model should perform TTT. Otherwise, it may perform inference on the frame $x_t$ (i.e., a single forward pass $h(f_{t-1}(x_t))$) and move on to the next frame $x_{t+1}$. In this case, $f_t = f_{t-1}$, since it is *not* updated. Next, we derive $\tau_t$.

**Deriving the adaptive threshold $\tau_t$:** Recall that the buffer $B$ contains latent features of all frames on which TTT was performed. We assume that the *size* of this buffer is $N$. Each element in the buffer contains the latent feature $v_{latent}(t_i)$, and the surprise computed at $t_i$. We define $\tau_t$ as :

$$\mu_t = \frac{1}{W} \sum_{i=t-W}^{t-1} S_i \quad ; \quad \sigma_t = \sqrt{\frac{1}{W} \sum_{i=t-W}^{t-1} (S_i - \mu_t)^2} \tag{6}$$

$$\tau_t = \mu_t + \sigma_t \tag{7}$$

[3] there are more numbers of pixels in $v_{visual}$ than the dimensions in $A$, so we multiply with the factor of $\frac{2}{\sqrt{N}}$ in Eq3 to compensate.

**Fig. 4: EpicTours dataset**: Our dataset consists of up to 3 hour long videos of walking tours across cities spanning the globe. We provide manual annotations at semantic/instance level for studying TTT on videos. Dataset shall be made publicly available.

Intuitively, $\mu_t$ tracks how the mean of surprise flows over time [11, 12], $\sigma_t$ estimates variance, $\tau_t$ is one standard deviation away from the mean[4]. Note that while the size of buffer $N$ is fixed, the threshold $\tau_t$ itself is dynamic. Therefore, unlike equation 1 where TTT was done for *each* iteration, the model now dynamically decides when to do TTT. Also, we are processing three frames in a timestep and *not* N.

**Revisiting the cold-start problem:** Initially, our model does not see any frame, hence the buffer $B$ is empty. Therefore, it is difficult to make a reliable estimate of whether to adapt or not. This issue also appears as a cold-start problem in recommendation systems [13]. We attempt to resolve this by 'adapting' $B$ initial-frames until the buffer is full. Since $W <<<$ video-length, this approach worked quite well, albeit at the expense of some initial-lag in the system[5].

## 3 Experiments on FFN

Here, we experiment with FFN across a wide range of benchmarks, for example, (i) dense tasks like semantic, instance, and panoptic segmentation; (ii) generalization across video depth-estimation; (iii) action-classification; (iv) segmentation on multi-hour long videos.

**Datasets:** We report video segmentation results in COCO-Videos [4], KITTI-STEP [14]. For action classification, we report UCF101 [15], Something-Something v2 [16]. Similarly, for depth-estimation, we create a similar setup as Video-DepthAnything (CVPR'2025) [17], and show results on 6 datasets (KITTI [18], Scannet [19], Bonn [20], NYUv2 [21], Sintel [22]). As shown in Table 2, most of these datasets contain only videos up to $5mins$ long.

To address this, we propose a new video segmentation dataset, **EpicTours Dataset**, which contains multi-hour-long videos of people exploring cities across

[4] Statistically, the central limit theorem says that the majority Gaussian probability mass lies in one standard deviation, which informed this choice.

[5] Buffer $B$ should contain 60 frames initially. Each frame takes $0.7sec$ for TTT, which incurs about 42 seconds of lag as $B$ gets initialized.

**Table 1: TTT Results for segmentation.** Time is in seconds per frame (A100 GPU). s.w: sliding window holding $k$ frames for TTT. no s.w.: TTT is done non-overlapping sliding windows. FFN offers competitive performance with lower time. †: effects of individual components analyzed later.

| Setting | Method | COCO Videos | | KITTI-STEP | | |
|---|---|---|---|---|---|---|
| | | *Inst.*↑ | *Pan.*↑ | *Val.*↑ | *Test*↑ | *Time*↓ |
| Independent frames | Main Task Only | 16.7 | 13.9 | 53.8 | 52.5 | 1.8 |
| | MAE Joint Training | 16.5 | 13.5 | 53.5 | 52.5 | 1.8 |
| | TTT-MAE No Memory | 35.4 | 20.1 | 53.6 | 52.5 | 3.8 |
| Full Video | Offline TTT-MAE All Frames | 33.6 | 19.6 | 53.2 | 51.2 | 1.8 |
| Stream | LN Adapt | 16.5 | 14.7 | 53.8 | 52.5 | 2.0 |
| | Tent | 16.6 | 14.6 | 53.8 | 52.2 | 2.8 |
| | Tent w/ Class Bal. | 16.7 | 14.8 | 53.8 | 52.5 | 3.7 |
| | Self-Train | - | - | 54.7 | 54.0 | 6.6 |
| | Self-Train w/ Class Bal. | - | - | 54.1 | 53.6 | 6.9 |
| | Online TTT-MAE (n.o. s.w.) | 35.3 | 20.8 | 48.1 | 51.7 | 0.4 |
| | Online TTT-MAE (s.w.) | 37.6 | 21.7 | 55.4 | 54.3 | 4.1 |
| | FFN (Ours)† | 45.1 | 29.6 | 57.3 | 59.5 | 0.7 |

the globe, representing geographical diversity. Our videos are densely annotated with the help of SAM 3 inference [23], manually filtered/refined by expert human annotators. Notably, our EpicTours dataset contains videos up to 3 hours long, with 30 classes (subset of COCO). This enables us to study the applicability of our FFN on real-world scenarios, for example, videos containing multiple-people. Please refer to the supplementary for more details.

**Metrics**: For instance, panoptic and semantic segmentation we report Average Precision (AP), Panoptic Quality (PQ), and mIoU respectively. Similarly, for video-depth estimation, we report AbsRel:Absolute Relative Error, and $\delta_1$ which denotes % of pixels where the ratio between the prediction and the ground truth falls below a threshold of 1.25. For action-classification, we report top-1 accuracy.

**Implementation details:** For segmentation (Tab1, Tab5), we initialize the backbone $f$ with Mask2former, SSL-task head $g$ from scratch. We train $g$ jointly during the training-time phase on both image datasets / a single long video. Similarly, $h$ the task-specific head (segmentation) utilizes Mask2former's pre-trained weights. For depth-estimation (Tab3), we leverage the Video-Depth Anything architecture. For video-classification (Tab4), we adopt the self-supervised backbone (DINOv3) [28] backbone (a very-recent foundational model). We set the memory buffer size $W = 50$, perform a *single* gradient step per incoming-frame, and use the Adam optimizer. Our SSL-head $g$ is trained with next-frame anticipation instead of current-frame reconstruction. We perform experiments with 3 seeds and report their mean accu-

**Table 2: EpicTours Dataset.** Our dataset contains videos averaging 1.5 hours with dense annotations. len means average length in seconds.

| Video Dataset | Len (s) ↑ | Frames↑ |
|---|---|---|
| CityScapes [24] | 1.8 | 3k |
| DAVIS [25] | 3.5 | 3.4k |
| YouTube [26] | 4.5 | 123k |
| KITTI-S [27] | 40.0 | 8k |
| COCO-V | 309.0 | 30k |
| EpicTours | 5300.4 | 2.49M |

**Table 3: Generalization of TTT to video-depth estimation.** Table shows zero-shot baselines containing both single image [31] and video depth estimation models [32–35]. For TTT baselines, we benchmark the strongest baseline (TTT-Online). Results may indicate FFN's competitiveness with current state of the art. (s.w.): denotes sliding window. ($^{\dagger}$): contains 50 frames. ($^{\ddagger}$): contains 170 frames. [31, 34].

| Method / Metrics | KITTI [18] | | Scannet [19] | | Bonn [20] | | NYUv2 [21] | | Sintel [22] $^{\dagger}$ | | Scannet $^{\ddagger}$ |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | AbsRel (↓) | $\delta_1$ (↑) | AbsRel (↓) | $\delta_1$ (↑) | AbsRel (↓) | $\delta_1$ (↑) | AbsRel (↓) | $\delta_1$ (↑) | AbsRel (↓) | $\delta_1$ (↑) | TAE (↓) |
| *Zero-shot baselines* | | | | | | | | | | | |
| DAv2-L [31] | 0.137 | 0.815 | 0.150 | 0.768 | 0.127 | 0.864 | 0.094 | 0.928 | 0.390 | 0.541 | 1.140 |
| NVDS [32] | 0.233 | 0.614 | 0.207 | 0.628 | 0.199 | 0.674 | 0.217 | 0.598 | 0.408 | 0.464 | 2.176 |
| NVDS + DAv2-L | 0.227 | 0.617 | 0.194 | 0.658 | 0.191 | 0.700 | 0.184 | 0.679 | 0.449 | 0.503 | 2.536 |
| ChronoDepth [33] | 0.243 | 0.576 | 0.199 | 0.665 | 0.199 | 0.665 | 0.173 | 0.771 | 0.192 | 0.673 | 1.022 |
| DepthCrafter [34] | 0.164 | 0.753 | 0.169 | 0.730 | 0.153 | 0.803 | 0.141 | 0.822 | 0.299 | 0.695 | 0.639 |
| DepthAnyVideo [35] | - | - | - | - | - | - | - | - | 0.405 | 0.659 | 0.967 |
| Video Depth-Anything [36] | 0.083 | 0.946 | 0.087 | 0.933 | 0.070 | 0.961 | 0.064 | 0.967 | 0.300 | 0.633 | 0.570 |
| *TTT baselines* | | | | | | | | | | | |
| Online TTT-MAE (s.w.) | 0.071 | 0.958 | 0.076 | 0.949 | 0.064 | 0.970 | 0.058 | 0.974 | 0.265 | 0.710 | 0.495 |
| FFN (Ours) | 0.059 | 0.972 | 0.062 | 0.965 | 0.051 | 0.982 | 0.049 | 0.988 | 0.230 | 0.762 | 0.380 |

racy. All experiments are performed on a single ampere of 80 GB.

**Baselines:** Inspired by [4], we consider multiple baselines that *span* image-models, offline video processing, test-time-adaptation (TTA), test-time-training. In 1, we categorize our baselines as (i) independent frames: the main task only means zero-shot per-frame inference on a model trained on image-segmentation. MAE joint-training *jointly* pretrains MAE [29] for reconstruction/segmentation. Similarly, TTT-MAE memory takes the *previous* MAE-joint training baseline, performs TTT on *each* frame *independently* (ii) full-video: an ideal *offline-oracle*, allowed to access the *entire* video (iii) stream: a tougher setup where a model only encounters frames one-by-one, 'without' peeking in the future.

Our TTA baselines include the pioneering TENT approach [30], layer-norm adaptation, self-training with confident pseudo-labels (to rule out whether improvements are due to self-supervision or TTT on videos). Next, we consider the TTT-Online baseline [4] across two setups (a) without sliding window: we perform TTT on a window of $w$ frames and then move onto a *non-overlapping* window; (b) with overlapping sliding window: the original setup in [4]. Finally, we consider zero-shot *foundational* models like DinoV3, evaluating feature robustness on general representation-learning.

**FFN can surpass TTT-online on segmentation:** In Tab 1 for the COCO-Videos dataset, our FFN obtains 45.1(+7.5 ↑) for instance segmentation, 29.6 (+7.9 ↑) for panoptic segmentation. Similarly, on the KITTI-STEP validation set, FFN gets 57.3(+1.9 ↑), 59.5(+5.2 ↑) on the testing split. One takeaway is that the *streaming* baselines (including our FFN) can *surpass* the offline TTT-MAE oracle, which *can access* entire video, thereby validating the principle-of-locality. Similarly, FFN performs competitively with TTA baselines like Tent.

**Table 4: TTT results for coarse-classification** on video-action datasets. We report top 1 accuracy.

| Method | UCF101↑ | SSv2↑ |
|---|---|---|
| *TTA baselines* | | |
| DINOv2 | 93.8 | 68.4 |
| V-JEPA 2 | 93.8 | 75.4 |
| Web-DINO | 94.1 | 68.1 |
| DINOv3 | 93.5 | 70.8 |
| *TTT-baselines* | | |
| Online TTT | 94.2 | 73.4 |
| FFN (Ours) | 95.0 | 74.1 |

**Table 5: TTT results on hours long videos**: evaluated on proposed EpicTours dataset.

| Method | Sem.↑ | Inst.↑ |
|---|---|---|
| *TTA/TTT baselines* | | |
| LN-Adapt | 32.1 | 26.8 |
| Tent | 35.6 | 27.7 |
| Tent w/ Bal. | 35.9 | 29.3 |
| Self-Train | 37.3 | – |
| Self-Train w/ Bal. | 39.1 | – |
| TTT-Online (s.w.) | 42.8 | 31.4 |
| FFN (Ours) | 49.3 | 36.7 |

**FFN for performing TTT with better compute-accuracy tradeoffs:** One concern about TTT is that it involves backpropagation through both backbone $f$, SSL-head $g$, for *every* incoming frame, making it *slow* in practice. For example, as shown in Tab1, TTT-Online takes $4.1sec$ *per* frame. Fortunately, FFN processes only 3 frames per window and consumes $0.7sec$ per timestep. It also gets a higher result (45.1 vs 37.6) showing a *better* compute-accuracy tradeoff. We acknowledge the potential to make TTT *even* faster.

**FFN can generalize to video depth-estimation:** We test the generalization ability of FFN on a challenging video depth-estimation task (Table 3). In particular, FFN performs well across the board on all 6 datasets.

**FFN can classify coarse-actions in videos:** While segmentation only tests per-frame fine-grained understanding, we also study how well FFN understands the *temporal* aspect. Thus, we subject FFN to action-classification in UCF101/ Something-Somethingv2. We notice gains of 0.8 ↑, 0.7 ↑, respectively. Note that SSv2 is especially challenging; it requires FFN to reason about *direction* of time: e.g., moving something up vs down.

**FFN's effectiveness on *realistic hours* long videos:** In Tab5, we subject our FFN to an even tougher task: how well can it adapt to hours long videos captured in the real-world, often on *low-resolution* devices like cellphones. In the EpicTours dataset, FFN shows promising gains of 6.5% ↑ for semantic segmentation, 5.3% ↑ for instance segmentation. A use-case *may be* to deploy FFN on drones for disaster-relief efforts or food-delivery.

## 4 Ablations on FFN

Here, we discuss ablations on the Frame Forgetting Network presented in Tab1.

**Rope encoding performs best:** Recall that FFN injects temporal information about an incoming frame by conditioning the backbone $f$ with time (Fig 3). Table 6 shows that the the rope-based time encoding outperforms both relative

**Table 6:** Impact of different choices of positional encodings.

| Method | COCO | | KITTI | |
|---|---|---|---|---|
| | *Inst.*↑ | *Pan.*↑ | *Val.*↑ | *Test*↑ |
| + relative | 40.5 | 24.1 | 53.9 | 54.8 |
| + absolute† | 45.1 | 29.6 | 57.3 | 59.5 |
| + rope | 46.8 | 31.2 | 58.9 | 60.7 |

**Table 7:** Impact of various choices on memory buffer $F$.

| Memory Buffer $F$ | COCO | | KITTI | |
|---|---|---|---|---|
| | *Inst.*↑ | *Pan.*↑ | *Val.*↑ | *Test*↑ |
| FIFO | 43.9 | 28.1 | 56.4 | 57.8 |
| MBO† | 45.1 | 29.6 | 57.3 | 59.5 |
| FIFO+MBO | 45.7 | 31.9 | 59.1 | 61.0 |

**Table 8:** Impact of different losses for computing $f_{latent}$.

| Method | COCO | | KITTI | |
|---|---|---|---|---|
| | *Inst.*↑ | *Pan.*↑ | *Val.*↑ | *Test*↑ |
| + $L_1$ | 43.2 | 27.8 | 56.1 | 57.4 |
| + $L_2$ | 44.5 | 28.9 | 56.8 | 58.6 |
| + Cosine† | 45.1 | 29.6 | 57.3 | 59.5 |

and absolute time encodings. Intuitively, rope can rotate both query/key matrices in latent space, and encode both relative/absolute positions *simultaneously*.

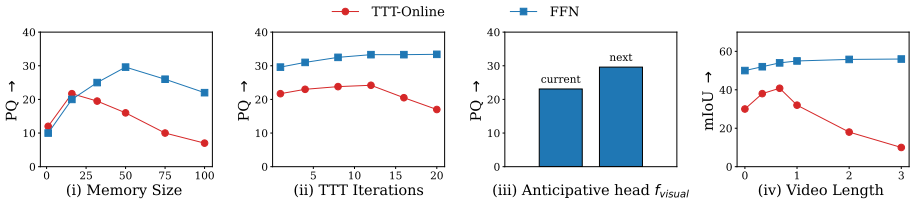


**Fig. 5:** First three plots show panoptic segmentation on COCO-Videos whereas last plot shows semantic segmentation on our EpicTours dataset. (i) Effect of increasing the size of buffer $B$ (ii) Increasing number of iterations on each test sample during TTT (iii) Effect of training SSL head with current-frame reconstruction/ vs next-frame. (iv) FFN's performance remains stable even when subjected to 3 hour long videos, whereas TTT-Online degrades rapidly.

**Temporal conditioning of the backbone $f$ helps:** We perform an ablation where we remove $t$ as a condition of the backbone $f$ and find that performance *drops* from 45.1 to 42.7 on COCO-Videos instance segmentation. This shows that instilling a notion of time in $f$ helps.

**Memory buffer B can take inspiration from both FIFO/MBC policies:** In Table 7, we implement two variations of the memory buffer $B$. MBC means that an incoming frame is merged with the 'most-similar' frame. MBC performs better than the FIFO queue [37]. Alternatively, one may also 'merge' an incoming frame (MBC) and consider it as 'most recent' element in a FIFO-queue[6], achieving even *better* results.

**Cosine loss performs best on $v_{latent}$:** In Tab8, we experiment with different types of loss to compute $v_{latent}$ (in eq4), and find that cosine works the best.

[6] After merging, we swap the merged position with *last position* in the FIFO queue (similar to the quicksort algorithm). This prevents the bottleneck of shifting *all elements* one by one to achieve sorting. The former operation is a single swap, whereas the latter is computationally expensive (and CPU becomes the bottleneck instead of GPU).

Intuitively, cosine projects all features to a unit-hypersphere, cancels out their magnitudes, and only measures angle difference, leading to a better estimate.

**Increasing buffer length helps, but *only* up to a limit:** In Fig5(i), we see that FFN performs well as buffer-size increases up to 50 frames, then drops. This helps validate the insight that keeping only *some* past-frames is important, and keeping everything *worsens* the performance. Intuitively, the capacity of the model $f$ is *finite*, trying to remember everything leads to instability during learning.

**Increasing TTT iterations improves performance:** Fig5(ii) shows that increasing TTT iterations improves performance. Note that FFN remains *almost constant*, which means that it manages to learn a lot even with just one gradient-step, whereas TTT-online needs several steps to reach its peak.

**Anticipating the 'next' frame is better than 'current' frame prediction in $f_{visual}$:** Fig5(iii) shows that it is better to predict what shall come one step in the future, rather than just predicting the current frame, thereby showing that 'anticipation' can provide a unique inductive bias to a neural net.

**FFN retains stable performance for long videos:** In Fig5(iv), we plot FFN's average performance as it continues to perform TTT over 3 hour long videos on our EpicTour dataset. Note that TTT-online degrades beyond 50 mins, whereas FFN improves/retains performance over time, indicating drift is less of an issue as compared to TTT-online. We refer the reader to supplementary material for additional ablations.

## 5 Related Work

There are several works operating within the regime of 'online model distillation' [3]: here the teacher is generally kept on the Internet, while a student sitting on an edge-device makes *decision* when to adapt or not. In FFN, we aim to adapt a *single model* on-device and don't require a teacher. Moreover, we focus on improving the quality of generated predictions and not explicitly matching the real-time performance of detectors, which remains an open challenge [4].

The idea of 'principle of locality' can be traced back to Vapnik et al [38, 39]. This also found applications in [40, 40–42]. FFN's difference lies in how it *implements* this locality: methods like TTT-Online [4] 'reset' weights after certain set of frames is processed. However, FFN's rely on the 'forgetting' principle: restore a model's baseline prior to adaptation. Although both aim to achieve identical goals (locality), the FFN's mechanism is different. While ideas on forgetting are typically used to unlearn 'bad' representations in safety-alignment [43], we show-

case the application of this principle for *representational-restoration*.

Similarly, the computer vision community has used the idea of test-time-training for several applications [44–47], especially depth estimation [48–52]. Other works such as [53–56] explore online-learning and their extensions to videos [57]. Although the videos considered in such papers are mostly synthetic/very short, we also show results on long-realistic videos.

Finally, video compression algorithms [58] also implement a mechanism to measure surprise: they measure how much change a new frame offers relative to a previous frame, which in turn is used to inform compression. The key idea is that most redundant frames are assigned short length codes [59]. However, this process is typically *not* learned, whereas FFN *learns* via an anticipative-head. Furthermore, in video-compression, the size of the file *grows* with the number of frames encountered, whereas in FFN all the knowledge of a video is squeezed into a *finite* set of weights [60].

## 6 Additional Discussions

We shall now discuss some directions that might help *inspire* further research on FFN. Please note that while these are relevant to FFN, their precise implementation remains beyond the scope of this work. Although TTT requires only *one* gradient step, it *remains slow* in practice. One reason is that it still relies on backpropagation. It creates a subtle 'layer-lock' [61]: the first layer has to 'wait' for the gradients from the last layer to come back, thereby wasting compute cycles. Alternatives include local-learning algorithms like forward-forward [60], target propagation [62] or no-prop [63], which can update these weights during the feed-forward phase only. A key challenge remains that these algorithms still do not surpass backpropagation's performance on large scale datasets.

TTT relies on the assumption that the 'useful' video frames keep 'streaming' continuously. However, there might be cases where there are *no changes* in the incoming frames (e.g., stationary cameras). During that time, learning does not happen, and FFN remains *idle*. An alternate mechanism was discussed by [64, 65]: make FFN enter a "sleeping" phase, sample the data from within FFN itself, and perform a *few* iterations of gradient descent. A key challenge remains how to "sample" efficiently [66], which was later *partially* resolved by score-matching networks relying on Langevin dynamics [67]. Further, we showed that anticipating the next-frame works *better* than the current-frame prediction. This lends itself to the question: Is it *possible* to meta-learn the SSL task itself. Recent works like TTT-MLP have just started exploring this interesting direction [68].

## 7 Conclusion

In this work, we introduced the Frame Forgetting Network (FFN): for performing test-time-training on videos which may be hours long. That is, a memory restoration mechanism allows the FFN to restore the representation of a model before adaptation. Our key contribution is the logic of handling sliding temporal-windows by operating selectively on exiting and entering frames, rather than doing duplicate processing at each time-step.

Next, we discuss an adaptive anticipative head that decides when to do TTT on an incoming frame. Finally, we introduced the EpicTours dataset, which contains *hours long videos* to effectively study this important problem. Our empirical results and ablations may help validate further merits of such an approach. In the future, we hope to explore FFN for hours long video generation [69].

## References

1. Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, et al. Dinov2: Learning robust visual features without supervision. *arXiv preprint arXiv:2304.07193*, 2023.
2. 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.
3. Ravi Teja Mullapudi, Steven Chen, Keyi Zhang, Deva Ramanan, and Kayvon Fatahalian. Online model distillation for efficient video inference. *arXiv preprint arXiv:1812.02699*, 2018.
4. Renhao Wang, Yu Sun, Arnuv Tandon, Yossi Gandelsman, Xinlei Chen, Alexei A Efros, and Xiaolong Wang. Test-time training on video streams. *Journal of Machine Learning Research*, 26(9):1–29, 2025.
5. Jon Bentley. Programming pearls: algorithm design techniques. *Communications of the ACM*, 27(9):865–873, 1984.
6. Geoffrey E Hinton. Learning distributed representations of concepts. In *Proceedings of the Annual Meeting of the Cognitive Science Society*, volume 8, 1986.
7. Ben Mildenhall, Pratul P Srinivasan, Matthew Tancik, Jonathan T Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. *Communications of the ACM*, 65(1):99–106, 2021.
8. Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.
9. Michael Zhang, James Lucas, Jimmy Ba, and Geoffrey E Hinton. Lookahead optimizer: k steps forward, 1 step back. *Advances in neural information processing systems*, 32, 2019.
10. Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*, 2018.
11. Donglai Wei, Joseph J Lim, Andrew Zisserman, and William T Freeman. Learning and using the arrow of time. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 8052–8060, 2018.

12. David Layzer. The arrow of time. *Scientific American*, 233(6):56–69, 1975.
13. Jian Wei, Jianhua He, Kai Chen, Yi Zhou, and Zuoyin Tang. Collaborative filtering and deep learning based recommendation system for cold start items. *Expert systems with applications*, 69:29–39, 2017.
14. Mark Weber, Jun Xie, Maxwell Collins, Yukun Zhu, Paul Voigtlaender, Hartwig Adam, Bradley Green, Andreas Geiger, Bastian Leibe, Daniel Cremers, et al. Step: Segmenting and tracking every pixel. *arXiv preprint arXiv:2102.11859*, 2021.
15. Khurram Soomro, Amir Roshan Zamir, and Mubarak Shah. Ucf101. In *arXiv preprint arXiv:1212.0402*, 2012.
16. Raghav Goyal, Samira Ebrahimi Kahou, Vincent Michalski, Joanna Materzynska, Susanne Westphal, Heuna Kim, Valentin Haenel, Ingo Fruend, Peter Yianilos, Moritz Mueller-Freitag, et al. The" something something" video database for learning and evaluating visual common sense. In *Proceedings of the IEEE international conference on computer vision*, pages 5842–5850, 2017.
17. Lihe Yang, Bingyi Kang, Zilong Huang, Zhen Zhao, Xiaogang Xu, Jiashi Feng, and Hengshuang Zhao. Depth anything v2. *Advances in Neural Information Processing Systems*, 37:21875–21911, 2024.
18. Andreas Geiger, Philip Lenz, Christoph Stiller, and Raquel Urtasun. Vision meets robotics: The kitti dataset. *The International Journal of Robotics Research*, 32(11):1231–1237, 2013.
19. Angela Dai, Angel X Chang, Manolis Savva, Maciej Halber, Thomas Funkhouser, and Matthias Nießner. Scannet: Richly-annotated 3d reconstructions of indoor scenes. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5828–5839, 2017.
20. E. Palazzolo, J. Behley, P. Lottes, P. Giguère, and C. Stachniss. ReFusion: 3D Reconstruction in Dynamic Environments for RGB-D Cameras Exploiting Residuals. 2019.
21. Pushmeet Kohli Nathan Silberman, Derek Hoiem and Rob Fergus. Indoor segmentation and support inference from rgbd images. In *ECCV*, 2012.
22. Daniel J. Butler, Jonas Wulff, Garrett B. Stanley, and Michael J. Black. *A Naturalistic Open Source Movie for Optical Flow Evaluation*, page 611–625. Jan 2012.
23. Nicolas Carion, Laura Gustafson, Yuan-Ting Hu, Shoubhik Debnath, Ronghang Hu, Didac Suris, Chaitanya Ryali, Kalyan Vasudev Alwala, Haitham Khedr, Andrew Huang, et al. Sam 3: Segment anything with concepts. *arXiv preprint arXiv:2511.16719*, 2025.
24. 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 *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3213–3223, 2016.
25. F. Perazzi, J. Pont-Tuset, B. McWilliams, L. Van Gool, M. Gross, and A. Sorkine-Hornung. A benchmark dataset and evaluation methodology for video object segmentation. In *Computer Vision and Pattern Recognition*, 2016.
26. Ning Xu, Linjie Yang, Yuchen Fan, Jianchao Yang, Dingcheng Yue, Yuchen Liang, Brian Price, Scott Cohen, and Thomas Huang. Youtube-vos: Sequence-to-sequence video object segmentation. In *Proceedings of the European conference on computer vision (ECCV)*, pages 585–601, 2018.
27. Lingdong Kong, Shaoyuan Xie, Hanjiang Hu, Lai Xing Ng, Benoit Cottereau, and Wei Tsang Ooi. Robodepth: Robust out-of-distribution depth estimation under corruptions. *Advances in Neural Information Processing Systems*, 36:21298–21342, 2023.

28. Oriane Siméoni, Huy V Vo, Maximilian Seitzer, Federico Baldassarre, Maxime Oquab, Cijo Jose, Vasil Khalidov, Marc Szafraniec, Seungeun Yi, Michaël Ramamonjisoa, et al. Dinov3. *arXiv preprint arXiv:2508.10104*, 2025.
29. Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross B. Girshick. Masked autoencoders are scalable vision learners. *CoRR*, abs/2111.06377, 2021.
30. Dequan Wang, Evan Shelhamer, Shaoteng Liu, Bruno Olshausen, and Trevor Darrell. Tent: Fully test-time adaptation by entropy minimization. *arXiv preprint arXiv:2006.10726*, 2020.
31. Lihe Yang, Bingyi Kang, Zilong Huang, Zhen Zhao, Xiaogang Xu, Jiashi Feng, and Hengshuang Zhao. Depth anything v2. *arXiv:2406.09414*, 2024.
32. Yiran Wang, Min Shi, Jiaqi Li, Zihao Huang, Zhiguo Cao, Jianming Zhang, Ke Xian, and Guosheng Lin. Neural video depth stabilizer. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9466–9476, 2023.
33. Jiahao Shao, Yuanbo Yang, Hongyu Zhou, Youmin Zhang, Yujun Shen, Matteo Poggi, and Yiyi Liao. Learning temporally consistent video depth from video diffusion priors. *arXiv preprint arXiv:2406.01493*, 2024.
34. Wenbo Hu, Xiangjun Gao, Xiaoyu Li, Sijie Zhao, Xiaodong Cun, Yong Zhang, Long Quan, and Ying Shan. Depthcrafter: Generating consistent long depth sequences for open-world videos. *arXiv preprint arXiv:2409.02095*, 2024.
35. Honghui Yang, Di Huang, Wei Yin, Chunhua Shen, Haifeng Liu, Xiaofei He, Binbin Lin, Wanli Ouyang, and Tong He. Depth any video with scalable synthetic data. *arXiv preprint arXiv:2410.10815*, 2024.
36. Sili Chen, Hengkai Guo, Shengnan Zhu, Feihu Zhang, Zilong Huang, Jiashi Feng, and Bingyi Kang. Video depth anything: Consistent depth estimation for super-long videos. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 22831–22840, 2025.
37. Bo He, Hengduo Li, Young Kyun Jang, Menglin Jia, Xuefei Cao, Ashish Shah, Abhinav Shrivastava, and Ser-Nam Lim. Ma-lmm: Memory-augmented large multimodal model for long-term video understanding. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 13504–13514, 2024.
38. A. Gammerman, V. Vovk, and V. Vapnik. Learning by transduction. In *In Uncertainty in Artificial Intelligence*, pages 148–155. Morgan Kaufmann, 1998.
39. Vladimir Vapnik and S. Kotz. *Estimation of Dependences Based on Empirical Data: Empirical Inference Science (Information Science and Statistics)*. Springer-Verlag, Berlin, Heidelberg, 2006.
40. Léon Bottou and Vladimir Vapnik. Local learning algorithms. *Neural computation*, 4(6):888–900, 1992.
41. Hao Zhang, Alexander C Berg, Michael Maire, and Jitendra Malik. Svm-knn: Discriminative nearest neighbor classification for visual category recognition. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, volume 2, pages 2126–2136. IEEE, 2006.
42. Moritz Hardt and Yu Sun. Test-time training on nearest neighbors for large language models. *arXiv preprint arXiv:2305.18466*, 2023.
43. Xiangyu Qi, Ashwinee Panda, Kaifeng Lyu, Xiao Ma, Subhrajit Roy, Ahmad Beirami, Prateek Mittal, and Peter Henderson. Safety alignment should be made more than just a few tokens deep. *arXiv preprint arXiv:2406.05946*, 2024.
44. Vidit Jain and Erik Learned-Miller. Online domain adaptation of a pre-trained cascade of classifiers. In *CVPR 2011*, pages 577–584. IEEE, 2011.

45. Assaf Shocher, Nadav Cohen, and Michal Irani. "zero-shot" super-resolution using deep internal learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3118–3126, 2018.
46. Yotam Nitzan, Kfir Aberman, Qiurui He, Orly Liba, Michal Yarom, Yossi Gandelsman, Inbar Mosseri, Yael Pritch, and Daniel Cohen-Or. Mystyle: A personalized generative prior. *arXiv preprint arXiv:2203.17272*, 2022.
47. Binhui Xie, Shuang Li, Mingjia Li, Chi Harold Liu, Gao Huang, and Guoren Wang. Sepico: Semantic-guided pixel contrast for domain adaptive semantic segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2023.
48. Alessio Tonioni, Oscar Rahnama, Thomas Joy, Luigi Di Stefano, Thalaiyasingam Ajanthan, and Philip HS Torr. Learning to adapt for stereo. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9661–9670, 2019.
49. Alessio Tonioni, Fabio Tosi, Matteo Poggi, Stefano Mattoccia, and Luigi Di Stefano. Real-time self-adaptive deep stereo. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 195–204, 2019.
50. Zhenyu Zhang, Stephane Lathuiliere, Elisa Ricci, Nicu Sebe, Yan Yan, and Jian Yang. Online depth learning against forgetting in monocular videos. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4494–4503, 2020.
51. Yiran Zhong, Hongdong Li, and Yuchao Dai. Open-world stereo video matching with deep rnn. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 101–116, 2018.
52. Xuan Luo, Jia-Bin Huang, Richard Szeliski, Kevin Matzen, and Johannes Kopf. Consistent video depth estimation. *ACM Transactions on Graphics (ToG)*, 39(4):71–1, 2020.
53. Nicklas Hansen, Rishabh Jangir, Yu Sun, Guillem Alenyà, Pieter Abbeel, Alexei A Efros, Lerrel Pinto, and Xiaolong Wang. Self-supervised policy adaptation during deployment. *arXiv preprint arXiv:2007.04309*, 2020.
54. Yu Sun, Wyatt L Ubellacker, Wen-Loong Ma, Xiang Zhang, Changhao Wang, Noel V Csomay-Shanklin, Masayoshi Tomizuka, Koushil Sreenath, and Aaron D Ames. Online learning of unknown dynamics for model-based controllers in legged locomotion. *IEEE Robotics and Automation Letters*, 6(4):8442–8449, 2021.
55. Yuejiang Liu, Parth Kothari, Bastien van Delft, Baptiste Bellot-Gurlet, Taylor Mordan, and Alexandre Alahi. Ttt++: When does self-supervised test-time training fail or thrive? *Advances in Neural Information Processing Systems*, 34, 2021.
56. Longhui Yuan, Binhui Xie, and Shuang Li. Robust test-time adaptation in dynamic scenarios. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 15922–15932, 2023.
57. Riccardo Volpi, Pau De Jorge, Diane Larlus, and Gabriela Csurka. On the road to online adaptation for semantic image segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 19184–19195, 2022.
58. Thomas Wiegand, Gary J Sullivan, Gisle Bjontegaard, and Ajay Luthra. Overview of the h. 264/avc video coding standard. *IEEE Transactions on circuits and systems for video technology*, 13(7):560–576, 2003.
59. Richard S Sutton. Generalization in reinforcement learning: Successful examples using sparse coarse coding. *Advances in neural information processing systems*, 8, 1995.
60. Geoffrey Hinton. The forward-forward algorithm: Some preliminary investigations. *arXiv preprint arXiv:2212.13345*, 2(3):5, 2022.

61. Sindy Löwe, Peter O'Connor, and Bastiaan Veeling. Putting an end to end-to-end: Gradient-isolated learning of representations. *Advances in neural information processing systems*, 32, 2019.
62. Dong-Hyun Lee, Saizheng Zhang, Asja Fischer, and Yoshua Bengio. Difference target propagation. In *Joint european conference on machine learning and knowledge discovery in databases*, pages 498–515. Springer, 2015.
63. Qinyu Li, Yee Whye Teh, and Razvan Pascanu. Noprop: Training neural networks without full back-propagation or full forward-propagation. *arXiv preprint arXiv:2503.24322*, 2025.
64. Cade Metz. Geoffrey hinton—the 'godfather' of ai and neural networks. *MIT Technology Review*, 2021.
65. Geoffrey E Hinton, Peter Dayan, Brendan J Frey, and Radford M Neal. The" wake-sleep" algorithm for unsupervised neural networks. *Science*, 268(5214):1158–1161, 1995.
66. Geoffrey E Hinton, Terrence J Sejnowski, and David H Ackley. *Boltzmann machines: Constraint satisfaction networks that learn*. Carnegie-Mellon University, Department of Computer Science Pittsburgh, PA, 1984.
67. Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. *arXiv preprint arXiv:2011.13456*, 2020.
68. Yu Sun, Xinhao Li, Karan Dalal, Jiarui Xu, Arjun Vikram, Genghan Zhang, Yann Dubois, Xinlei Chen, Xiaolong Wang, Sanmi Koyejo, et al. Learning to (learn at test time): Rnns with expressive hidden states. *arXiv preprint arXiv:2407.04620*, 2024.
69. Karan Dalal, Daniel Koceja, Gashon Hussein, Jiarui Xu, Yue Zhao, Youjin Song, Shihao Han, Ka Chun Cheung, Jan Kautz, Carlos Guestrin, et al. One-minute video generation with test-time training. *arXiv preprint arXiv:2504.05298*, 2025.
70. Bowen Cheng, Ishan Misra, Alexander G Schwing, Alexander Kirillov, and Rohit Girdhar. Masked-attention Mask Transformer for Universal Image Segmentation. In *CVPR*, 2022.
71. Khurram Soomro, Amir Roshan Zamir, and Mubarak Shah. Ucf101: A dataset of 101 human actions classes from videos in the wild. *arXiv preprint arXiv:1212.0402*, 2012.
72. 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 *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3213–3223, 2016.
73. Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. Masked autoencoders are scalable vision learners. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 16000–16009, 2022.
74. Yossi Gandelsman, Yu Sun, Xinlei Chen, and Alexei Efros. Test-time training with masked autoencoders. *Advances in Neural Information Processing Systems*, 35:29374–29385, 2022.