

AgedNet-Motion: Training-Free Selective Task Refresh for Resource-Aware Streaming Multi-Task Perception

Sungjun Jang

Kyung Hee University

Taegeun Oh

tgoh@du.ac.kr

DongSeoul University

Research Article

Keywords: streaming multi-task perception, selective execution, temporal caching, resource-aware inference, training-free scheduling, video analytics

Posted Date: October 19th, 2026

DOI: <https://doi.org/10.21203/rs.3.rs-10804096/v1>

License: © ⓘ This work is licensed under a Creative Commons Attribution 4.0 International License.

[Read Full License](#)

Additional Declarations: No competing interests reported.

AgedNet-Motion: Training-Free Selective Task Refresh for Resource-Aware Streaming Multi-Task Perception

Sungjun Jang¹ and Taegeun Oh^{2*}

¹Department of Computer Science and Engineering, Kyung Hee University, Yongin, Republic of Korea.

^{2*}Department of Electronic Engineering, Dong Seoul University, Seongnam-si, Gyeonggi-do, 13117, Republic of Korea.

*Corresponding author(s). E-mail(s): tgoh@du.ac.kr;

Abstract

Purpose: Conventional multi-task perception models execute all task-specific branches on every video frame, even when adjacent frames contain redundant information. This study investigates whether task-level selective refresh can reduce streaming inference latency while retaining useful output quality without training an additional scheduler.

Methods: AgedNet-Motion executes the common feature-extraction path on every frame, refreshes one selected task branch, and reuses the remaining outputs from task-specific temporal caches. Its refresh score combines a calibrated task-specific age-regret lookup table with cumulative grayscale frame difference. The lookup table and one scalar motion weight are selected on a calibration partition without gradient optimization. The system is evaluated with YOLOP, A-YOLOM, and YOLOPX on BDD100K video streams.

Results: Relative to full refresh, AgedNet-Motion reduces mean model and post-processing latency by 42.19%, 51.54%, and 33.24% for YOLOP, A-YOLOM, and YOLOPX, respectively. On YOLOP, it improves mean ground-truth quality over Age-Only by 0.453 percentage points with a video-cluster bootstrap confidence interval above zero. For A-YOLOM, calibration selects a zero motion weight, reproducing Age-Only while outperforming Tuned Fixed. For YOLOPX, it outperforms Tuned Fixed and remains statistically indistinguishable from Age-Only. ONNX Runtime deployment preserves comparable relative latency reductions for YOLOP and A-YOLOM.

Conclusion: Task-level temporal caching provides a practical resource-quality control mechanism for existing streaming multi-task models. Calibration can

exploit motion when beneficial and disable it when the time-only policy is sufficient, without modifying model weights or training a separate predictor.

Keywords: streaming multi-task perception, selective execution, temporal caching, resource-aware inference, training-free scheduling, video analytics

1 Introduction

Visual perception tasks, including object detection and region segmentation, are essential for understanding the surroundings of autonomous vehicles and mobile robots. A driving system must locate vehicles and pedestrians while simultaneously identifying drivable areas and lane markings. Reflecting these requirements, datasets such as BDD100K provide annotations for heterogeneous visual tasks in driving scenes [29]. Multi-task learning (MTL) models such as YOLOP [26] and A-YOLOM [23] combine a shared feature-extraction path with task-specific heads to produce multiple perception outputs in a single model. This structure avoids the redundant feature extraction incurred by running an independent model for each task. In a resource-constrained streaming multimedia system, however, sharing features within a frame does not by itself eliminate repeated task-output computation across temporally redundant frames.

Nevertheless, conventional MTL inference executes every task-specific branch on every frame, even for a continuous video stream. Adjacent video frames are often similar, and an output computed for one frame may remain useful for a short period. Recomputing every head without considering scene changes or output freshness therefore wastes computation. Conversely, reusing a task output for too long can fail to capture object motion, boundary changes, or newly appearing objects, thereby degrading perception quality. Streaming multi-task perception consequently requires not only caching previous outputs but also deciding how to allocate a limited execution budget among task refreshes.

Converting an existing MTL model into a selective-execution system raises two challenges. First, separating the shared path from the task-specific branches and executing only selected branches must preserve the outputs of the original model while reducing actual latency. Second, the system must select which task to refresh online using only the current scene and cache state. A predefined fixed schedule (Tuned Fixed, once its cycle is calibration-selected) cannot react to input changes. An Age-Only policy, which considers only the time elapsed since each output was last refreshed, cannot distinguish different scene dynamics for the same task and output age. A learned scheduler could model these dynamics, but it would require additional training data, target definitions, and model-specific training and validation. To retain the original MTL model without introducing another learned component, the scheduler should be lightweight, causal, and training-free.

This work proposes *AgedNet-Motion*, a training-free method for task-level selective refresh. AgedNet-Motion executes the common feature-extraction path on every frame, refreshes only a limited number of task-specific branches, and reuses the remaining

outputs from a temporal cache. It computes refresh priorities by combining a task-specific age-regret lookup table obtained from calibration data with the cumulative grayscale frame difference observed since each task was last refreshed. The motion weight is selected solely on the calibration partition from a finite, predefined candidate set, without training a gradient-based predictor. A zero candidate can remove the motion multiplier; in the A-YOLOM evaluation reported here, this choice produced the same schedule as Age-Only.

This work evaluates AgedNet-Motion on YOLOP, A-YOLOM, and YOLOPX using original BDD100K video streams. Relative to full refresh, AgedNet-Motion reduces the mean model and post-processing latency by 42.19% on YOLOP, 51.54% on A-YOLOM, and 33.24% on YOLOPX. On YOLOP, it improves mean ground-truth quality over Age-Only by 0.453 percentage points, with a video-cluster bootstrap 95% confidence interval above zero. It also achieves a higher point estimate than Tuned Fixed, although the corresponding confidence interval includes zero. On A-YOLOM, calibration selects a motion weight of zero, causing AgedNet-Motion to produce exactly the same schedule as Age-Only while achieving a mean ground-truth quality 0.922 percentage points above Tuned Fixed. On YOLOPX, calibration selects an interior motion weight, yielding a mean ground-truth quality 0.940 percentage points above Tuned Fixed with a confidence interval above zero, while the small difference from Age-Only is not statistically confirmed in either direction. Thus, AgedNet-Motion is never confirmed worse than the strongest non-learned baseline on any of the three models and improves upon the time-only policy when motion is beneficial on YOLOP.

Beyond this native-inference research pipeline, this work further deploys the resulting selective-execution system through a production inference runtime (ONNX Runtime) on GPU. On a held-out subset of BDD100K videos, the deployed system closely reproduces the ground-truth quality and preserves comparable relative latency reduction observed with native inference, indicating that AgedNet-Motion’s benefits are not an artifact of the research implementation.

This work makes the following contributions:

- This work implements a resource-aware, temporal-cache-based streaming architecture that separates the shared path and task-specific branches of existing MTL models, refreshes only selected branches, and reuses the remaining outputs without modifying the trained weights or output structures.
- This work proposes AgedNet-Motion, a training-free scheduler that combines task-specific output age with input motion accumulated since the last refresh. It uses only calibration lookup tables and a single scalar weight, without an additional predictor or gradient training.
- This work evaluates the resulting accuracy-latency trade-off on YOLOP, A-YOLOM, and YOLOPX under the same BDD100K video split and evaluation protocol. YOLOP and A-YOLOM are architecturally distinct MTL codebases; YOLOPX shares YOLOP’s backbone and neck implementation but replaces its detection head with an anchor-free design. This work shows that calibration can remove the motion multiplier when it provides no benefit; on A-YOLOM, this yielded the same schedule as Age-Only, and on YOLOPX calibration selected an interior motion weight with only a marginal effect over the time-only policy.

- This work deploys AgedNet-Motion through a production inference runtime (ONNX Runtime) on GPU and confirms on held-out real video that the deployed system closely reproduces the accuracy and latency characteristics measured with native inference.

2 Background and Motivation

2.1 Shared Computation and Static Execution in Multi-Task Models

Multi-task learning (MTL) produces multiple task outputs from a single input. Using an independent model for each task repeatedly performs similar low- and mid-level feature extraction. Running these models concurrently requires high instantaneous compute and memory capacity, whereas running them sequentially increases the end-to-end response time. Conventional MTL models reduce this redundancy by combining a shared backbone with multiple task-specific heads.

This work targets MTL architectures in which distinct task heads follow a shared feature-extraction path. YOLOP and A-YOLOM are representative models of this architecture for driving-scene perception, jointly performing object detection, drivable-area segmentation, and lane segmentation [23, 26]. The objective of this work is not to redesign their perception architectures, but to change how their task heads are executed over continuous video, as detailed in Section 4.

Figure 1 compares MTL inference with concurrent and sequential execution of three independent models. MTL removes redundant feature extraction by executing its shared backbone only once. Conventional MTL inference, however, still executes every task head for every input. It therefore shares computation within an input but repeats the same static execution pattern across a video stream, regardless of whether each task output needs to be refreshed.

2.2 Reusing Task Outputs in Continuous Inputs

Unlike an unordered collection of still images, adjacent frames in a video stream contain substantial temporal redundancy. When the input changes only slightly, detection and segmentation outputs from the preceding frame may remain useful for a short period. Prior video recognition systems have exploited this redundancy through a variety of temporal-reuse mechanisms; Section 3 discusses specific instances of this literature. These approaches demonstrate that full computation need not be repeated for every input.

For an MTL model with a shared backbone and multiple heads, temporal reuse can be applied at the granularity of task outputs. After computing shared features for the current frame, the system can execute a subset of the heads and obtain the outputs of skipped heads from a temporal cache. The *age* of a task output denotes the number of frames elapsed since that head was last executed. An output with age zero has been refreshed for the current input, whereas an output with a larger age was produced from an earlier input and is being reused.

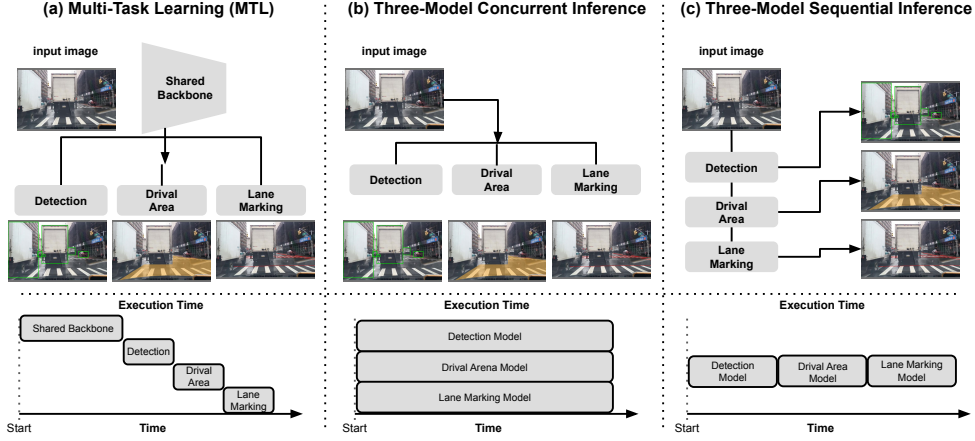


Fig. 1 Execution structures for MTL, concurrent multi-model, and sequential multi-model inference over three tasks. MTL shares its feature extraction path, but conventional inference still executes every task head for every input.

Task-level reuse reduces head computation, but an old output may fail to reflect the current input. Moreover, outputs for different tasks do not become stale at a uniform rate. Their representations and quality metrics differ, so the quality loss at a given age can vary by task. Even for the same task and age, the reuse error can depend on the magnitude of the input change. Selective execution therefore requires not only a maximum cache-age constraint but also a scheduler that allocates a limited head-execution budget among tasks.

2.3 Limitations of Time-Based Selection Policies

The simplest selection policy is a classical round-robin schedule: it repeatedly executes task heads in a fixed, predefined order. Tuned only in its task order via calibration (*Tuned Fixed* in Section 5), it gives every task an execution opportunity and has a predictable cost, but it cannot distinguish a nearly static input from a rapidly changing one. It may consequently spend computation on an output that does not require an update or fail to refresh a change-sensitive output in time.

An *Age-Only* policy adapts the Age of Information (AoI) framework from networked status-update systems [10, 28] to task-branch refresh in an MTL model: in place of AoI’s analytic staleness penalty, it builds an empirical lookup table of the mean reuse regret observed for each task and output age on calibration data, then greedily selects the task with the largest expected regret at its current age. This policy captures task-dependent temporal sensitivity and is more adaptive than a fixed schedule; Section 3 discusses how this AoI perspective has been extended elsewhere and why this work calibrates a separate regret curve per task rather than assume one global staleness curve. Neither Age-Only nor Tuned Fixed is a contribution of this work; both are baselines adapted from existing scheduling concepts to the MTL task-refresh setting studied here.

Age-Only nevertheless assigns the same priority to every occurrence of a given task and age. It cannot determine whether the input has remained nearly unchanged or substantial motion has accumulated since the last refresh. A task with low mean regret on the calibration partition may also receive consistently lower priority than the other tasks and consequently be refreshed infrequently. Fixed scheduling ignores the input content, while Age-Only uses only elapsed time. A scheduler that responds to continuous inputs therefore needs a runtime-observable change signal in addition to task age.

2.4 Training-Free Motion-Aware Selective Execution

One option is to train a neural network or regression model to predict the quality loss caused by input changes. Such an approach, however, requires additional scheduler training data and target definitions, and may require retraining or revalidation when the underlying MTL model changes. The goal of this work is not to train another prediction model, but to retain the weights of an existing MTL model while replacing static task execution with selective execution. The scheduler should therefore have low computational overhead, operate causally, and require no gradient training.

Frame difference is a representative low-cost signal for measuring changes in a video stream; Section 3 discusses specific systems that use such signals for frame- or layer-level decisions. This observation motivates combining elapsed time with observed input motion when managing the freshness of MTL task outputs.

Based on this motivation, this work proposes AgedNet-Motion, a training-free task-level scheduler. It combines the temporal sensitivity represented by task-specific output age with the grayscale frame difference accumulated since each task was last refreshed. When motion is small, the scheduler largely retains the age-based priority; as input changes accumulate, it places greater emphasis on refreshing stale outputs. This process requires neither a separate accuracy predictor nor gradient training.

The central question of this work is whether task outputs can be selectively refreshed over continuous inputs without modifying the weights or task definitions of an existing MTL model, and whether combining a low-cost motion signal with output age can respond more effectively to input changes than fixed or time-only policies. The following section presents the selective execution architecture, the calibration of the age-regret lookup tables and motion weight, and the online scheduling rule of AgedNet-Motion.

3 Related Work

3.1 Static Execution in Multi-Task Models

Multi-task learning (MTL) produces outputs for multiple tasks from a single input by combining a shared backbone with multiple task-specific heads (Section 2). Although this structure reduces redundant computation across models, conventional inference executes every task head regardless of temporal changes in the input or the need to refresh individual outputs. The problem addressed in this work is therefore not tied

to a particular application environment; it is the static task-execution scheme of MTL models in general.

This work uses YOLOP, A-YOLOM, and YOLOPX as representative models for evaluating this MTL structure. YOLOP shares features extracted by a single encoder across object detection, drivable-area segmentation, and lane segmentation [26]. A-YOLOM performs the same three tasks using a unified segmentation structure and lightweight task heads [23]. YOLOPX [9] retains YOLOP’s shared backbone and neck but replaces its detection head with an anchor-free design, providing a same-lineage architectural variant alongside the fully independent A-YOLOM codebase. BDD100K provides heterogeneous visual-task annotations for evaluating these models [29]. These models and the dataset serve as the evaluation platform rather than defining the scope of this work’s contribution. Other panoptic driving-perception networks such as YOLOPv2 [7] follow a similar shared-backbone, multi-branch design and are natural candidates for future extension of this framework beyond the three models evaluated here. AgedNet-Motion retains the trained weights and output structures, separates the task paths after the shared computation, refreshes only the selected head for the current input, and reuses the remaining outputs from a temporal cache. It thereby converts the static execution scheme of an MTL model into selective execution.

3.2 Efficient Multi-Task Inference

Prior work has reduced the inference cost of MTL models by adapting their execution paths to the input or a computational budget. AdaMTL attaches a task-aware policy network to a shared encoder and selects which blocks and tokens to activate according to the input and target complexity [15]. Efficient Controllable Multi-Task Architectures trains a supernet with slimmable shared encoders and task-specific decoders, from which a subnetwork is selected at deployment according to the compute budget and task preferences [1]. M³ViT activates only sparse expert paths associated with the requested task and jointly optimizes their execution with the underlying hardware [5]. These approaches adjust the accuracy–cost trade-off by selecting computation paths based on the input or user requirements.

VPEngine shares features extracted by a foundation-model backbone across multiple task heads and allows the inference frequency of each task to be dynamically adjusted according to application demands [13]. Its separation of shared feature extraction from task-specific execution is closely related to this work. However, prior efficient MTL approaches require a separately learned policy, supernet training, or a new model structure, or determine task frequencies from application-level requirements. In contrast, AgedNet-Motion does not modify the trained weights of YOLOP, A-YOLOM, or YOLOPX. It uses the age of each task output and observed image motion to select the head to refresh in the current frame. Thus, it converts a static MTL execution scheme into task-level conditional execution without training an additional conditional-computation module.

3.3 Conditional and Dynamic Computation

A broader line of work introduces input-dependent computation paths into otherwise static networks, generally by training an auxiliary policy or gating mechanism. Skip-Net learns a gating policy that skips residual blocks per input using reinforcement learning [24]. BlockDrop trains a policy network to choose which residual blocks to execute at test time, trading accuracy for latency [27]. BranchyNet attaches early-exit branches to intermediate layers and terminates inference once a confidence threshold is met [22]. AR-Net dynamically selects the input frame resolution for action recognition according to a learned policy [14], and Skip-Convolutions learn to skip convolutional computation on temporally redundant regions of a video [6]. Han et al. survey this broader family of dynamic neural networks, most of which require training a dedicated gating, routing, or exit-decision module alongside or on top of the base network [8]. AgedNet-Motion shares the goal of input-dependent computation but differs in mechanism: rather than training a gate, router, or exit policy, it reuses an empirically calibrated, closed-form age-motion score to decide which existing task branch to execute, requiring no additional trainable parameters.

3.4 Temporal Reuse and Freshness-Aware Scheduling

Temporal redundancy in video has been widely exploited to avoid repeated computation on every frame. Clockwork ConvNets observes that semantic content changes more slowly than pixels and executes different network layers at fixed or adaptive rates [19]. Deep Feature Flow runs an expensive convolutional network only on sparse key frames and propagates their features to other frames through a flow field [30]. Low-Latency Video Semantic Segmentation combines temporal feature propagation with an accuracy-prediction-based adaptive scheduler to allocate computation to the current frame [11]. These methods reduce per-frame segmentation cost by controlling the update frequency of network layers or intermediate features.

Video analytics systems have also used inexpensive visual-change signals to select which frames to process. Reducto relates low-level frame-difference features, such as pixel and edge differences, to video content and dynamically selects filtering thresholds. It then omits transmission and processing for frames expected to produce similar analytics results [12]. A separate line of work formalizes staleness itself as Age of Information (AoI), originally proposed for networked status-update systems [10, 28] and since applied to scheduling decisions for downstream learning and inference tasks [18, 21], including evidence that the relationship between age and prediction error need not follow a single shared function [20]. The Age-Only baseline in this work (Section 2) is a direct instantiation of this AoI perspective at task-branch granularity, and the Tuned Fixed baseline is a classical round-robin schedule with its task order selected by calibration; neither is a contribution of this work. RALF applies an analogous accuracy-aware staleness criterion to decide when to refresh derived features in a machine-learning feature store [25], indicating that freshness-aware scheduling is a recognized problem beyond video perception as well.

Existing temporal-reuse methods primarily reuse network layers or intermediate features for a single vision task, or select entire frames as the unit of processing.

In contrast, AgedNet-Motion executes the common feature-extraction path on every frame while independently caching the outputs of three semantically distinct task branches. It combines the age of each task output with the cumulative grayscale frame difference observed since that task was last refreshed and uses the resulting priority to allocate a limited head-execution budget. Thus, AgedNet-Motion manages output freshness at task granularity in an existing MTL model without introducing feature warping or a learned accuracy predictor.

4 Resource-Aware Selective-Refresh System

4.1 AgedNet-Motion Overview

This work considers a pretrained MTL model composed of a common feature-extraction path and multiple task-specific branches. Let $\mathcal{T} = \{1, \dots, K\}$ denote an arbitrary set of K tasks and I_t the input at frame t . This work uses F for the maximal common path that is executed regardless of the selected task, B_i for the branch used only by task i , and P_i for its deterministic post-processing. A fresh task output is therefore $y_i^t = P_i(B_i(F(I_t)))$. Depending on the underlying model, B_i may include a private neck or decoder in addition to the final prediction head. This decomposition requires every B_i to be independently executable once $F(I_t)$ is available; models with irreducible cross-task or cascaded dependencies require a different partition and are outside the present scope. AgedNet-Motion executes $F(I_t)$ on every frame but only one B_i per frame after initialization. The post-processed outputs of branches that are not executed are obtained from task-specific temporal caches. The formulation does not depend on a particular dataset or output type; each task only needs a quality measure for offline calibration.

For each task i , the scheduler maintains a cached output c_i , its last refresh frame r_i , and its current output age $a_i(t) = t - r_i$. At the first frame, $t = 0$, all task branches are executed to initialize the K caches. At each subsequent frame, one task is selected, and only its branch and cache are updated; the caches of the other $K - 1$ tasks remain unchanged. The execution budget after initialization is therefore one task-branch invocation per frame, without assuming that all branches have equal latency. Fresh and reused outputs are assembled to provide all K task outputs at every frame.

Figure 2 presents the online execution flow of AgedNet-Motion. The current frame is sent to both the common path and a lightweight motion monitor. The scheduler selects a task branch using task ages, motion accumulated since each task’s last refresh, and an age-regret lookup table and motion weight fixed during offline calibration. The selected branch replaces its cached output, while the other caches are reused with a hold-last policy. This process uses neither future frames, online ground truth, nor a separately learned predictor.

4.2 Selective Task-Branch Execution and Temporal Caching

AgedNet-Motion retains the layers and trained parameters of the original MTL model and separates only its forward path into common and task-specific computation. The common feature $z_t = F(I_t)$ is computed once regardless of the selected task. If task

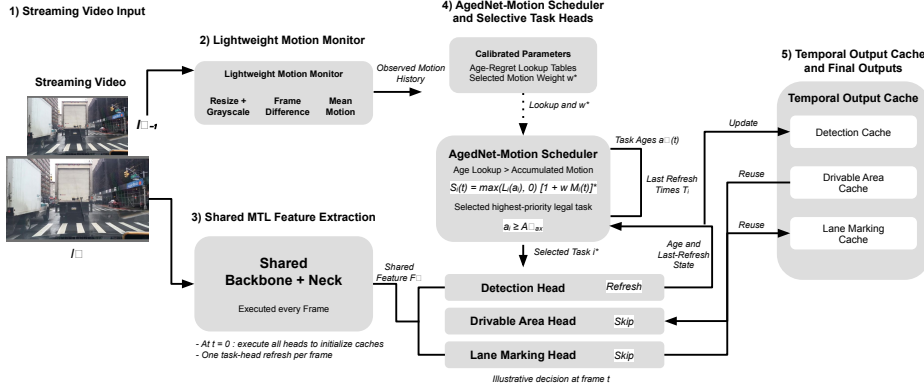


Fig. 2 Selective-refresh architecture of AgedNet-Motion. The common feature-extraction path runs on every frame. The scheduler uses each task’s output age and motion accumulated since its last refresh to select one task-specific branch. The selected output updates its temporal cache, while the outputs of the remaining tasks are reused. A three-task instantiation is shown for illustration, but the formulation supports an arbitrary number of task branches.

i^* is selected, only $B_{i^*}(z_t)$ and P_{i^*} are executed; the computation graphs of the other task-specific branches are not entered. The separation boundary is chosen such that selecting all branches performs the same computation as the original full forward pass. Selective execution therefore changes neither the model weights nor the task-output formats.

At the first frame, all post-processed outputs y_i^0 are computed from z_0 , and the state is initialized as $c_i(0) = y_i^0$ and $r_i = 0$. For $t > 0$, if task i^* is selected, the cache is updated as

$$c_i(t) = \begin{cases} P_i(B_i(z_t)), & i = i^*, \\ c_i(t-1), & i \neq i^*. \end{cases} \quad (1)$$

The selected task is assigned $r_{i^*} = t$, making its age zero immediately after the refresh. The ages of unselected tasks increase by one at the next frame.

Cached outputs are reused directly without optical-flow or ego-motion warping. This work also employs an overdue-candidate guard to prevent a task from being ignored indefinitely. Once a task reaches the threshold $A_{\max}$, overdue tasks compete before tasks with younger caches. If several tasks reach the threshold simultaneously, only one branch is executed in the current frame, and the remaining overdue tasks stay in the prioritized candidate set at the next decision. Thus, $A_{\max}$ is a forced-candidate threshold rather than a hard upper bound that guarantees every task age remains below it. The candidate-set construction and selection rule are defined in Section 4.4.

4.3 Training-Free Calibration

Before online inference, AgedNet-Motion constructs two artifacts on a disjoint calibration partition. The first is an age-regret lookup table $L_i(a)$ indexed by task and integer age; the second is a scalar motion weight w^* . Both artifacts are frozen before held-out online inference. Ground truth is used only to construct the lookup and select the motion weight; it is never available to the online scheduler.

Let s denote a calibration sample formed from a temporal window, let τ_s be an annotated reference frame in that window, and let $g_{i,s}^{\tau_s}$ be the corresponding ground truth for task i . A dataset with dense annotations may provide several reference frames per sequence, whereas a sparsely annotated dataset may provide only one. Given a task-specific quality function q_i , the reuse regret between a fresh output and an age- a hold-last output is

$$R_{i,s}(a) = q_i(y_{i,s}^{\tau_s}, g_{i,s}^{\tau_s}) - q_i(y_{i,s}^{\tau_s-a}, g_{i,s}^{\tau_s}). \quad (2)$$

This quantity measures the task-quality decrease caused by reusing an output from an earlier frame rather than computing a fresh output at the annotated reference frame. Intermediate frames need not be annotated because both the fresh and reused predictions are evaluated against the same reference-frame ground truth. The quality function may differ across tasks and output types, but it must be expressed on the common range $[0, 1]$ and oriented so that a larger value indicates better quality. The task-specific functions used in this work’s experimental instantiation are given in Section 5.

For each task and age, the lookup table averages regret over eligible calibration samples:

$$L_i(a) = \frac{1}{|\mathcal{S}_{i,a}|} \sum_{s \in \mathcal{S}_{i,a}} R_{i,s}(a), \quad 1 \leq a \leq A_{\max}, \quad (3)$$

where $\mathcal{S}_{i,a}$ is the set of annotated calibration samples for which regret at task i and age a can be computed. The lookup is a discrete table of empirical means indexed by task and integer age, not a fitted regression curve. This work chooses $A_{\max}$ within the common calibration coverage so that $\mathcal{S}_{i,a}$ is nonempty for every retained task-age pair. Here, *training-free* does not mean that calibration data or ground truth is absent; it means that no new scheduler predictor is trained and no gradient optimization is performed. The table $L_i(a)$ itself is estimated through ground-truth-supervised calibration.

The motion weight is selected from the predefined finite set $\mathcal{W} = \{0, 1, 2, 4, 8, 16\}$. For each $w \in \mathcal{W}$, this work rolls out the calibration windows under predeclared initialization settings using the same lookup, one-branch-per-frame budget, and overdue-candidate guard. Let $\bar{Q}_i^{\text{cal}}(w) \in [0, 1]$ denote the aggregate calibration quality for task i , computed with a task-appropriate evaluation metric. The cross-task calibration utility is the equally weighted mean

$$\Phi_{\text{cal}}(w) = \frac{1}{K} \sum_{i \in \mathcal{T}} \bar{Q}_i^{\text{cal}}(w). \quad (4)$$

The final weight is

Algorithm 1: Training-free calibration of AgedNet-Motion

Input: $\mathcal{D}_{\text{cal}}$, F , and $\{B_i, P_i\}_{i \in \mathcal{T}}$
Input: A_{max} with valid coverage, rollout settings $\mathcal{P}$
Input: Finite weight set $\mathcal{W}$
Output: Age-regret lookup L and selected motion weight w^*

```

1  foreach  $i \in \mathcal{T}$  do
2    for  $a \leftarrow 1$  to  $A_{\text{max}}$  do
3       $\mathcal{R}_{i,a} \leftarrow \emptyset$ ;
4    end
5  end
6  foreach  $s \in \mathcal{D}_{\text{cal}}$  do
7    Read  $\tau_s$  and available task ground truth;
8    Compute full outputs for the frames required by each valid age;
9    foreach  $i \in \mathcal{T}$  do
10     foreach valid  $a \in \{1, \dots, A_{\text{max}}\}$  for  $(i, s)$  do
11       Compute  $R_{i,s}(a)$  using Equation 2;
12        $\mathcal{R}_{i,a} \leftarrow \mathcal{R}_{i,a} \cup \{R_{i,s}(a)\}$ ;
13     end
14   end
15 end
16 foreach  $i \in \mathcal{T}$  do
17   for  $a \leftarrow 1$  to  $A_{\text{max}}$  do
18      $L_i(a) \leftarrow \text{mean}(\mathcal{R}_{i,a})$ ;
19   end
20 end
21 foreach  $w \in \mathcal{W}$  do
22   Roll out all  $(s, p) \in \mathcal{D}_{\text{cal}} \times \mathcal{P}$  under the fixed budget and guard;
23   Compute  $\Phi_{\text{cal}}(w)$  using Equation 4;
24 end
25  $w^* \leftarrow \min(\arg \max_{w \in \mathcal{W}} \Phi_{\text{cal}}(w))$ ;
26 return  $L, w^*$ ;

```

$$w^* = \arg \max_{w \in \mathcal{W}} \Phi_{\text{cal}}(w). \quad (5)$$

The lookup proxy q_i and aggregate rollout metric $\bar{Q}_i^{\text{cal}}$ need not be identical: the former estimates reference-frame reuse regret, whereas the latter evaluates the complete rollout. For aggregation, every rollout metric is expressed on the common range $[0, 1]$. The concrete metrics and rollout settings used in this work’s experiments are reported in Section 5. This work chooses the smaller w in the event of a tie. Setting $w = 0$ removes the motion multiplier, but it does not generally guarantee equivalence to the Age-Only baseline that uses the raw lookup because Equation 8 clips negative lookup values. Whether the two policies produce the same schedule is examined separately in Section 5. The selected $L_i(a)$ and w^* are frozen for the corresponding MTL model and are neither refitted per held-out video nor updated online. Algorithm 1 summarizes the complete procedure.

4.4 Online Age–Motion Scheduling

Online frame difference is computed directly from the input video. This work resizes I_t to a grayscale image G_t of size $H_m \times W_m$ and computes the normalized mean absolute difference between consecutive frames:

$$m_t = \frac{1}{255H_mW_m} \sum_{x=1}^{W_m} \sum_{y=1}^{H_m} |G_t(x, y) - G_{t-1}(x, y)|. \quad (6)$$

This causal scalar uses only inputs observed through the current frame and requires neither an optical-flow network nor an external motion sensor. This work’s implementation uses a resolution of 96×54 for G_t .

Because tasks have different last-refresh times, the same frame-level motion is accumulated over a different interval for each task. The accumulated motion of task i at frame t is

$$M_i(t) = \sum_{k=r_i+1}^t m_k. \quad (7)$$

At each frame, $a_i(t) = t - r_i$ and $M_i(t)$ denote the state immediately before branch selection. After the selected branch is refreshed, its stored last-refresh time becomes t and its post-update age is zero; this updated state is used at frame $t + 1$.

This work defines $\tilde{a}_i(t) = \min(a_i(t), A_{\max})$ to handle ages outside the lookup range. The AgedNet-Motion refresh-priority score is

$$S_i(t) = \max(L_i(\tilde{a}_i(t)), 0) (1 + w^* M_i(t)). \quad (8)$$

The first factor represents the task-specific temporal sensitivity measured during calibration, while the second accounts for input changes accumulated since the last refresh. This work clips the lookup value at zero so that a negative mean regret cannot reverse the refresh priority.

To apply the overdue-candidate guard, this work first computes $\mathcal{F}_t = \{i \in \mathcal{T} : a_i(t) \geq A_{\max}\}$ and defines the legal candidate set as

$$\mathcal{C}_t = \begin{cases} \mathcal{F}_t, & \mathcal{F}_t \neq \emptyset, \\ \mathcal{T}, & \mathcal{F}_t = \emptyset. \end{cases} \quad (9)$$

The selected task is

$$i^* = \arg \max_{i \in \mathcal{C}_t} S_i(t). \quad (10)$$

For equal scores, this work chooses the first task in a fixed deterministic total order $\prec_{\mathcal{T}}$ defined for the task set. After selection, this work computes $P_{i^*}(B_{i^*}(z_t))$ and updates c_{i^*} and r_{i^*} . The other caches and last-refresh states remain unchanged, so the ages and accumulated motion at the next frame depend only on the observed history.

Algorithm 2 gives the complete process from cache initialization through repeated online decisions. The online scheduler reads only the current and past frames, cache state, and frozen L and w^* . Ground truth, future frames, and current outputs from unselected branches are unavailable before the decision, and no parameters are fitted per video.

In summary, AgedNet-Motion constructs only a discrete age-regret lookup and a single model-specific motion weight through ground-truth-supervised calibration. During online inference, it makes deterministic decisions using these two frozen artifacts

Algorithm 2: Causal online selective refresh with AgedNet-Motion

Input: Stream $I_0, I_1, \dots, F$, and $\{B_i, P_i\}_{i \in \mathcal{T}}$
Input: Frozen L, w^* , and threshold $A_{\max}$
Output: Cached outputs $\{c_i(t)\}_{i \in \mathcal{T}}$

```
1  $z_0 \leftarrow F(I_0); G_0 \leftarrow \text{GrayResize}(I_0);$ 
2 foreach  $i \in \mathcal{T}$  do
3    $c_i(0) \leftarrow P_i(B_i(z_0)); r_i \leftarrow 0; M_i(0) \leftarrow 0;$ 
4 end
5 emit  $\{c_i(0)\}_{i \in \mathcal{T}};$ 
6 for  $t \leftarrow 1, 2, \dots$  do
7    $z_t \leftarrow F(I_t); G_t \leftarrow \text{GrayResize}(I_t);$ 
8   Compute  $m_t$  using Equation 6;
9   foreach  $i \in \mathcal{T}$  do
10     $a_i(t) \leftarrow t - r_i; M_i(t) \leftarrow M_i(t-1) + m_t;$ 
11  end
12   $\mathcal{F}_t \leftarrow \{i : a_i(t) \geq A_{\max}\};$ 
13  if  $\mathcal{F}_t \neq \emptyset$  then
14     $\mathcal{C}_t \leftarrow \mathcal{F}_t;$ 
15  else
16     $\mathcal{C}_t \leftarrow \mathcal{T};$ 
17  end
18  foreach  $i \in \mathcal{C}_t$  do
19    Compute  $S_i(t)$  using Equation 8;
20  end
21   $i^* \leftarrow \arg \max_{i \in \mathcal{C}_t} S_i(t);$ 
22  Break ties by the fixed order  $\prec_{\mathcal{T}};$ 
23   $c_{i^*}(t) \leftarrow P_{i^*}(B_{i^*}(z_t));$ 
24   $r_{i^*} \leftarrow t; M_{i^*}(t) \leftarrow 0;$ 
25  foreach  $i \in \mathcal{T} \setminus \{i^*\}$  do
26     $c_i(t) \leftarrow c_i(t-1);$ 
27  end
28  emit  $\{c_i(t)\}_{i \in \mathcal{T}};$ 
29 end
```

together with only the frames and cache state observed so far. After cache initialization, it executes the common path and one task-specific branch per frame while continuously providing all K task outputs through hold-last caching. Every online decision is causal and uses no future frame, ground truth, or learned predictor.

5 Evaluation

5.1 Experimental Setup

This work evaluates AgedNet-Motion on YOLOP, A-YOLOM, and YOLOPX [9] using the same original BDD100K video streams and split. YOLOPX retains YOLOP’s ELANNet backbone and PaFPNELAN neck but replaces its detection head with an anchor-free, YOLOX-style design; it is therefore a weaker instance of architectural distinctness than A-YOLOM, whose codebase is fully independent of YOLOP. From a manifest of 300 original BDD100K videos, 100 videos form the calibration partition used to build $L_i(a)$ and select w^* (Section 4.3), and the remaining 200 videos form the held-out partition used only for reporting; this split is identical across all three models. Because each video contains a single annotated keyframe, this work evaluates three cold-start window lengths per video (31, 32, and 33 frames, corresponding to phase offsets 0, 1, and 2 relative to the keyframe) and pools the results; the three phase-shifted samples of the same video are treated as one cluster in the bootstrap

so that no video contributes statistically independent evidence more than once. This work sets $A_{\max} = 6$.

Two quality functions serve different purposes, as anticipated in Section 4.3. The lookup proxy q_i used to construct $L_i(a)$ is a fast reference-frame metric: a single-threshold detection F1 score for object detection and foreground intersection-over-union for drivable-area and lane segmentation. The aggregate rollout metric $\bar{Q}_i^{\text{cal}}$ used to select w^* , and the mean GT quality reported in this section, instead use detection mAP50 and the two foreground intersection-over-union scores. Mean GT quality is their equally weighted arithmetic mean. Detection mAP50–95, computed from the precision–recall curve over confidence thresholds, is reported separately as a stricter detection metric and is not included in mean GT quality. All reported policy comparisons use a video-cluster bootstrap with 1000 resamples.

Native-inference latency is measured on an RTX 3060 GPU. For the deployment validation in Section 5.4, which covers YOLOP and A-YOLOM only, the same common path and task branches are additionally exported to ONNX and executed with ONNX Runtime 1.20.2 [16] using the CUDA execution provider, on the same GPU; YOLOPX is evaluated with native inference only (Section 5.5).

5.2 Calibration Results

Calibration selects a different tuned fixed cycle and motion weight for each model. For YOLOP, the tuned cycle is lane→od→drivable and calibration selects $w^* = 16$, the largest candidate in $\mathcal{W}$. For A-YOLOM, the tuned cycle is od→lane→drivable and calibration selects $w^* = 0$, the smallest candidate. For YOLOPX, the tuned cycle is od→lane→drivable and calibration selects $w^* = 8$, an interior candidate.

Figure 3 shows the calibration mean GT quality for every candidate in $\mathcal{W} = \{0, 1, 2, 4, 8, 16\}$. The three models exhibit three distinct trends: YOLOP’s calibration quality increases monotonically with w , indicating that the motion signal keeps helping up to the largest candidate evaluated; A-YOLOM’s calibration quality decreases monotonically, indicating that adding motion sensitivity does not help this model and calibration safely falls back to $w^* = 0$, which reproduces Age-Only exactly by Equation 8; and YOLOPX’s calibration quality is flat across $w \in \{0, 1, 2, 4\}$, rises marginally at $w = 8$ (by 0.016 percentage points over $w = 0$), and falls at $w = 16$, so calibration selects the interior peak but the motion signal’s effect on this model is small in either direction.

5.3 Main Results

Table 1 reports mean GT quality, per-task metrics, and native-inference latency on the 200 held-out videos, phase-pooled, for six policies:

- **Full refresh:** every task branch is executed on every frame; no selective execution is applied. This is the conventional MTL inference behavior described in Section 2 and the reference point for all latency-saving and quality-loss figures reported below.
- **Tuned Fixed:** the classical round-robin schedule of Section 2, adapted here only by selecting its task order via calibration in Section 5.2; it is not a contribution of this work.

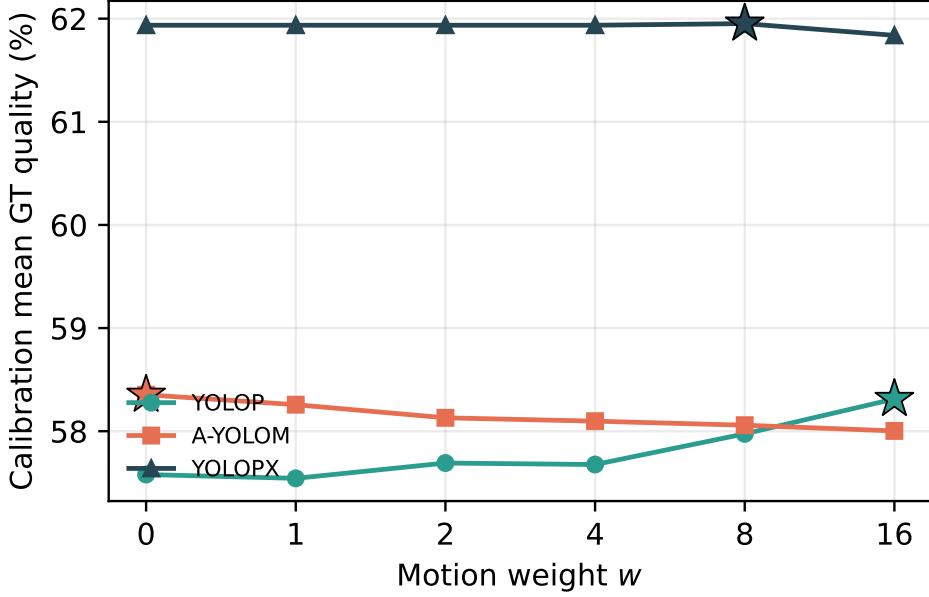


Fig. 3 Calibration-only grid search over the motion weight $w \in \{0, 1, 2, 4, 8, 16\}$. YOLOP’s calibration quality increases monotonically, selecting $w^* = 16$; A-YOLOM’s decreases monotonically, selecting $w^* = 0$; YOLOPX is flat and then peaks at an interior $w^* = 8$. Stars mark the selected weight for each model.

- **Age-Only:** the Age-of-Information-derived, time-only policy of Section 2 (also not a contribution of this work), ranking candidates by the same raw age-regret lookup $L_i(a)$, without clipping or the motion term. As noted in Section 4.3, this need not coincide exactly with AgedNet-Motion at $w = 0$ because Equation 8 clips negative lookup values; the two happen to coincide exactly on A-YOLOM, as Table 1 shows.
- **AgedNet-Motion:** the training-free policy proposed in this work (Section 4).
- **Learned Freshness:** a ridge-regression scheduler fit on calibration features to predict reuse regret directly, framed as an instance of the learning-to-rank problem of ordering task branches by refresh priority [2, 3]. This is a gradient-trained alternative from an earlier stage of this project’s development, reported here only as a reference point; it did not beat the non-learned baselines on any of the three models and is not a contribution of this work.
- **Terminal GT Oracle:** a non-causal upper bound that, given access to ground truth at evaluation time, always selects whichever task would have been best to refresh at each frame under the same one-branch-per-frame budget. Because it requires information unavailable to any online scheduler, *it is excluded from the policy comparison* and reported only to show the remaining headroom below this same-budget, non-causal optimum.

Table 1 Main results on 200 held-out BDD100K videos, phase-pooled. Latency is native-inference, RTX 3060. Bold marks AgedNet-Motion, the policy proposed in this work; it is not necessarily the column maximum (see the YOLOPX row, where Age-Only is marginally higher on four columns).

Model	Policy	mAP50	mAP50-95	Drivable IoU	Lane IoU	Mean GT Quality	Latency	Saving vs. Full
YOLOP	Full refresh	76.46%	43.74%	80.52%	23.85%	60.28%	8.00 ms	–
	Tuned Fixed	71.77%	37.29%	80.16%	22.59%	58.18%	4.65 ms	41.90%
	Age-Only	73.56%	39.85%	79.89%	19.97%	57.81%	4.64 ms	41.97%
	AgedNet-Motion	73.82%	40.17%	79.63%	21.32%	58.26%	4.62 ms	42.19%
	Learned Freshness	73.63%	40.16%	79.46%	21.29%	58.13%	4.63 ms	42.10%
	Terminal GT Oracle	76.84%	39.01%	81.19%	23.95%	60.66%	–	–
A-YOLOM	Full refresh	77.58%	46.37%	81.84%	21.62%	60.35%	7.78 ms	–
	Tuned Fixed	71.98%	39.96%	81.47%	19.88%	57.78%	3.79 ms	51.27%
	Age-Only	75.75%	44.08%	80.43%	19.99%	58.73%	3.78 ms	51.42%
	AgedNet-Motion	75.75%	44.08%	80.43%	19.99%	58.73%	3.77 ms	51.54%
	Learned Freshness	74.19%	42.49%	80.48%	18.70%	57.79%	3.79 ms	51.35%
	Terminal GT Oracle	77.98%	41.66%	82.24%	21.88%	60.70%	–	–
YOLOPX	Full refresh	82.74%	48.00%	82.94%	25.13%	63.60%	16.26 ms	–
	Tuned Fixed	76.08%	40.85%	82.63%	24.08%	60.93%	10.91 ms	32.89%
	Age-Only	80.91%	45.59%	81.41%	23.39%	61.90%	10.85 ms	33.30%
	AgedNet-Motion	80.76%	45.50%	81.49%	23.36%	61.87%	10.86 ms	33.24%
	Learned Freshness	79.66%	43.84%	81.76%	23.11%	61.51%	10.84 ms	33.33%
	Terminal GT Oracle	82.94%	43.41%	83.65%	25.21%	63.93%	–	–

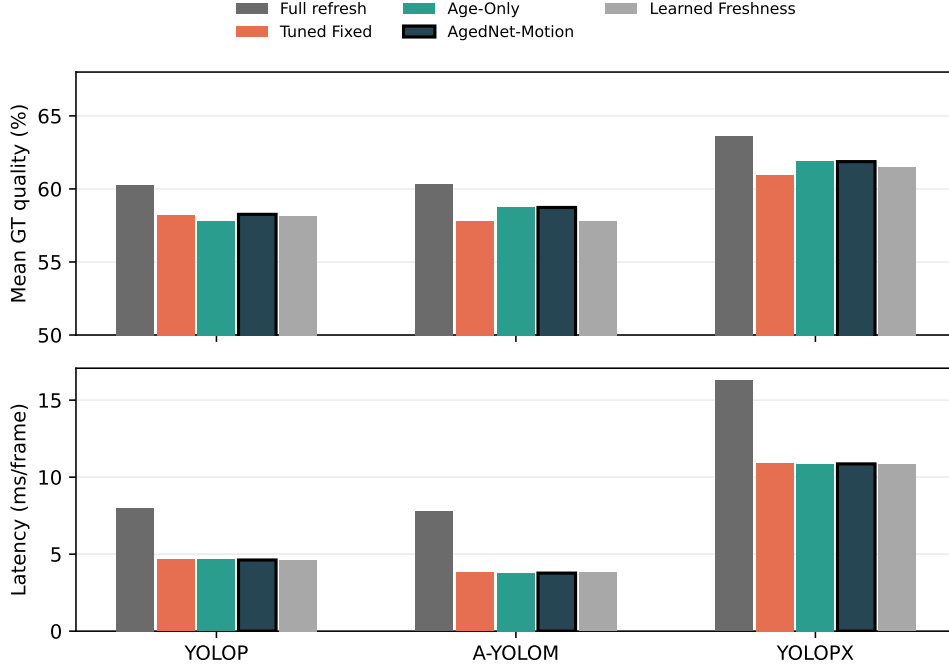


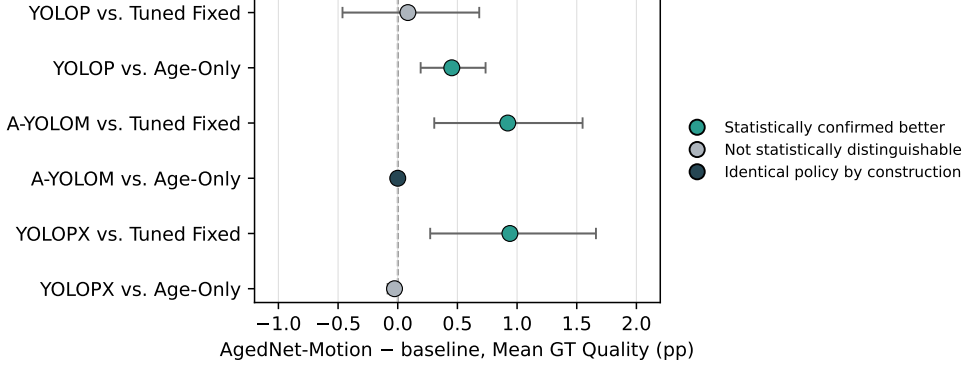
Fig. 4 Mean GT quality (top) and native-inference latency (bottom) for Full refresh, Tuned Fixed, Age-Only, AgedNet-Motion (outlined), and Learned Freshness, on all three evaluated models. Every selective policy cuts latency by a similar amount within a given model; AgedNet-Motion is never confirmed worse than the stronger of Tuned Fixed and Age-Only on any model, and is statistically confirmed better than the weaker of the two on every model (Table 2), without requiring a per-model choice of which simpler baseline to trust.

Table 2 reports the video-cluster bootstrap comparison between AgedNet-Motion and each non-learned baseline; Figure 5 visualizes the same six comparisons directly. Across the three models, the better of Tuned Fixed and Age-Only is not the same baseline every time, and calibration is what tells AgedNet-Motion which behavior to adopt without this being specified by hand: it recovers Age-Only when motion information does not help (A-YOLOM, by construction at $w^* = 0$), leans toward the motion signal when it does (YOLOP), and lands close to both when neither baseline has an edge (YOLOPX). On YOLOP, AgedNet-Motion’s improvement over Age-Only is statistically confirmed (95% CI strictly above zero); its higher point estimate over Tuned Fixed is not. On A-YOLOM, AgedNet-Motion is identical to Age-Only by construction ($w^* = 0$) and its improvement over Tuned Fixed is statistically confirmed. On YOLOPX, AgedNet-Motion’s improvement over Tuned Fixed is statistically confirmed, while its small, slightly negative point-estimate difference from Age-Only is not statistically distinguishable from zero in either direction – the two policies are empirically indistinguishable on this model.

A task-level breakdown clarifies why AgedNet-Motion helps on YOLOP. Relative to Full refresh, Age-Only sacrifices lane quality by -3.88 percentage points, reflecting

Table 2 Video-cluster bootstrap comparison (1000 resamples).

Model	Comparison	Mean Diff	95% CI	Verdict
YOLOP	AgedNet-Motion – Tuned Fixed	+0.085pp	[−0.463, +0.683]	not confirmed
YOLOP	AgedNet-Motion – Age-Only	+0.453pp	[+0.192, +0.736]	confirmed
A-YOLOM	AgedNet-Motion – Tuned Fixed	+0.922pp	[+0.306, +1.549]	confirmed
A-YOLOM	AgedNet-Motion – Age-Only	0.000pp	[0, 0]	identical policy
YOLOPX	AgedNet-Motion – Tuned Fixed	+0.940pp	[+0.272, +1.661]	confirmed
YOLOPX	AgedNet-Motion – Age-Only	−0.027pp	[−0.085, +0.015]	not confirmed

**Fig. 5** Forest plot of the six video-cluster bootstrap comparisons in Table 2 (Mean GT Quality, percentage points; error bars are 95% CIs). No comparison shows AgedNet-Motion statistically confirmed worse than a baseline: three of six are statistically confirmed better, one is identical to Age-Only by construction ($w^* = 0$ on A-YOLOM), and two are not statistically distinguishable in either direction. This is the property emphasized throughout this section: AgedNet-Motion never needs to be told in advance which of Tuned Fixed or Age-Only is the stronger baseline for a given model.

the structurally low lane priority discussed in Section 2; AgedNet-Motion reduces this loss to -2.53 percentage points while simultaneously staying closer to Full on detection than Tuned Fixed does (-2.64 versus -4.70 percentage points). On A-YOLOM, Age-Only’s lane loss relative to Full (-1.63 percentage points) is already close to Tuned Fixed’s (-1.74 percentage points), so lane starvation is less severe to begin with; adding motion sensitivity does not improve calibration quality in this regime, and calibration correctly disables it. On YOLOPX, Age-Only’s lane loss relative to Full (-1.74 percentage points) and AgedNet-Motion’s (-1.76 percentage points) are nearly identical, and the two policies’ per-task refresh shares differ by at most 0.3 percentage points (Table 1 raw data); motion information simply does not change which task the scheduler prioritizes often enough to move the aggregate result, consistent with the small calibration gap between $w = 0$ and $w^* = 8$ in Figure 3. Headroom relative to Terminal GT Oracle remains at $+2.40$ percentage points on YOLOP, $+1.98$ percentage points on A-YOLOM, and $+2.06$ percentage points on YOLOPX, indicating that none of the three models’ selective-execution policies are close to exhausting the attainable budget-constrained quality. The scheduler’s own computation cost is negligible relative to model inference: 0.0053 ms/frame on YOLOP and 0.0054 ms/frame on

A-YOLOM, versus 4.6–4.8 ms/frame on YOLOP and 3.6–3.9 ms/frame on A-YOLOM for a single common path plus one task branch; the YOLOPX driver script does not separately instrument this cost, but the scheduling rule performs the same constant-time arithmetic per task regardless of model, so no different order of magnitude is expected.

5.4 Deployment Validation

The results above are obtained with native PyTorch inference. To confirm that AgedNet-Motion’s behavior is not an artifact of the research implementation, this work deploys the same selective-execution system through a production inference runtime and re-evaluates it on real, held-out video. Production inference-serving systems typically optimize aggressively for per-request latency under real-time constraints [4, 17]; this experiment checks that AgedNet-Motion’s savings survive that kind of optimized deployment path rather than only the native research execution graph used above.

Setup. This work exports the common path F and each task branch B_i, P_i to separate ONNX graphs and executes them with ONNX Runtime 1.20.2 using the CUDA execution provider on the same RTX 3060 GPU used for native inference. Each ONNX head session declares only the common-path outputs it actually consumes, since the exporter prunes unused graph inputs. Five warm-up iterations are run before every timing measurement to exclude one-time initialization cost. This work observed a small numeric discrepancy between native PyTorch and ONNX Runtime outputs on GPU, traced it to TensorFloat-32 (TF32) being enabled by default for PyTorch matrix multiplications and convolutions on Ampere GPUs but disabled by default in the ONNX Runtime CUDA execution provider, and verified experimentally that disabling TF32 removes the discrepancy. This confirms the gap is a numerical-precision artifact of the runtime default, not an export or scheduling error.

Protocol. The deployed scheduler must be the same frozen artifact evaluated in Section 5.3, not a separately fitted one. This work therefore recomputes $L_i(a)$, the tuned fixed cycle, and w^* from the same 100-video calibration partition used in Section 5.1; since this computation is deterministic, the recomputed cycle and w^* are bit-identical to Section 5.2 for both models, confirming that the deployed system implements the same scheduler as the one reported above. This work then evaluates on 20 videos drawn from the 200-video held-out partition. This smaller sample is sufficient to check runtime-deployment fidelity, which is the purpose of this experiment; it is not intended to re-establish statistical significance, which Section 5.3 already reports over the full held-out set.

Results. Table 3 compares native PyTorch and ONNX Runtime outputs for Full refresh, Tuned Fixed, Age-Only, and AgedNet-Motion on both models. Figure 6 visualizes the same comparison. The largest quality discrepancy between the two runtimes is 0.0083 percentage points, consistent with the TF32 rounding difference described above and far smaller than any of the differences between scheduling policies. Latency savings relative to Full refresh are preserved after deployment: AgedNet-Motion reduces ONNX Runtime latency by 43.2% on YOLOP and 52.3% on A-YOLOM, close to the native-inference savings in Table 1.

Table 3 Deployment validation: native PyTorch vs. ONNX Runtime (GPU), $N = 20$ held-out videos, using the same frozen scheduler as Table 1.

Model	Policy	PT mAP50	PT mAP50-95	PT Drv. IoU	PT Lane IoU	PT GT Q.	ONNX GT Q.	Diff	PT ms	ONNX ms
YOLOP	Full refresh	80.61%	46.99%	81.24%	23.75%	61.87%	61.87%	+0.0061	9.291	10.685
	Tuned Fixed	78.63%	41.53%	80.91%	23.22%	60.92%	60.92%	-0.0016	5.170	6.330
	Age-Only	76.67%	41.36%	81.24%	18.24%	58.72%	58.71%	-0.0069	4.745	6.008
	AgedNet-Motion	80.00%	43.61%	81.23%	19.01%	60.08%	60.07%	-0.0083	4.830	6.072
A-YOLOM	Full refresh	83.94%	49.87%	85.74%	21.48%	63.72%	63.72%	-0.0014	8.119	12.228
	Tuned Fixed	76.48%	43.24%	84.43%	20.72%	60.54%	60.54%	+0.0007	3.753	6.143
	Age-Only	83.94%	49.87%	84.22%	17.63%	61.93%	61.93%	+0.0025	3.583	5.809
	AgedNet-Motion	83.94%	49.87%	84.22%	17.63%	61.93%	61.93%	+0.0025	3.603	5.833

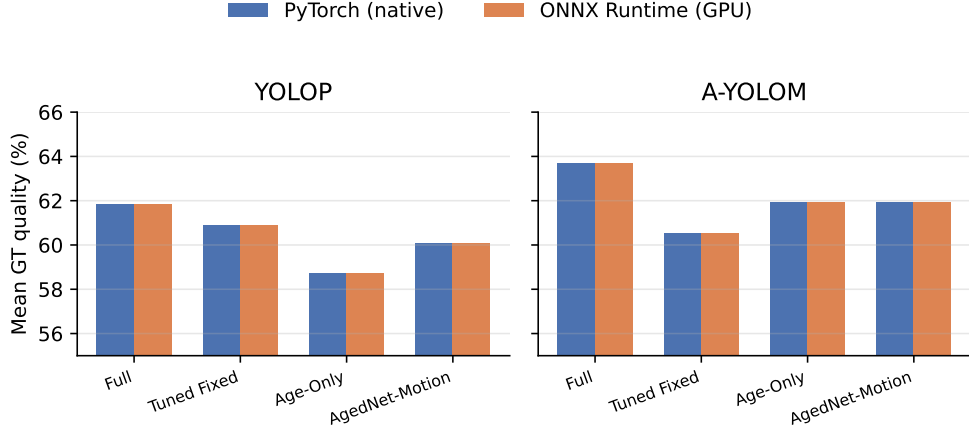


Fig. 6 Mean GT quality under native PyTorch inference versus ONNX Runtime (GPU) deployment, for Full refresh, Tuned Fixed, Age-Only, and AgedNet-Motion, on YOLOP and A-YOLOM. The two runtimes are visually indistinguishable for every policy.

In summary, AgedNet-Motion reduces native-inference latency by 42.19% on YOLOP, 51.54% on A-YOLOM, and 33.24% on YOLOPX relative to Full refresh. On YOLOP and A-YOLOM it never falls below the stronger of Tuned Fixed and Age-Only in point estimate; on YOLOPX its point estimate is 0.03 percentage points below Age-Only, a difference the bootstrap does not distinguish from zero, so it is never *confirmed* worse than the stronger baseline on any of the three models. Its improvement over Age-Only on YOLOP and its improvement over Tuned Fixed on A-YOLOM and YOLOPX are each statistically confirmed. Deploying the same frozen scheduler through ONNX Runtime on GPU reproduces this accuracy and preserves comparable latency savings for YOLOP and A-YOLOM, indicating that these properties hold beyond the native-inference research pipeline; this deployment check was not repeated for YOLOPX (Section 5.5).

5.5 Limitations

- The 200-video held-out partition has previously been used once to evaluate the rejected Learned Freshness baseline; this reuse is not repeated beyond the single confirmation reported here.
- The motion weight w^* is selected independently per model and is not a value that transfers to an unseen MTL model; a new model requires re-running the same calibration procedure over its own candidate set.
- The motion signal uses only the mean grayscale frame difference; other signals such as the 90th-percentile difference, optical flow, or per-task local motion were not evaluated.
- This work evaluates three cold-start window positions (31, 32, and 33 frames) rather than a fully continuous long-running streaming session.
- Each video provides a single annotated keyframe for ground-truth scoring.

- The temporal cache uses hold-last reuse without optical-flow warping or ego-motion compensation.
- All latency numbers are measured on an RTX 3060; absolute values on other devices require separate measurement.
- The deployment validation in Section 5.4 uses $N = 20$ held-out videos without a bootstrap confidence interval; it is a runtime-fidelity check, not an independent re-establishment of the statistical significance already reported in Section 5.3. It was performed for YOLOP and A-YOLOM only; YOLOPX is evaluated with native PyTorch inference only.
- YOLOPX shares its `lib` codebase, ELANNet backbone, and PaFPNELAN neck with YOLOP; only its detection head is a different, anchor-free design. It is therefore weaker evidence of architectural generality than A-YOLOM, whose codebase and backbone are fully independent of YOLOP’s.
- YOLOPX’s latency saving (33.24%) is smaller than YOLOP’s (42.19%) and A-YOLOM’s (51.54%) because a separate branch-latency profiling measurement shows its shared common path accounts for 52.0% of full forward latency, leaving less headroom for branch-skipping regardless of which scheduler selects the branch. This is a property of the architecture’s compute distribution, not a weakness of AgedNet-Motion.

6 Conclusion

This work presents AgedNet-Motion as a resource-aware streaming perception system with a training-free selective-refresh method that addresses the static execution of every task-specific branch on every frame in streaming multi-task perception. Without modifying the trained weights or output structures of an existing MTL model, AgedNet-Motion executes the common feature-extraction path on each frame, refreshes selected task branches, and reuses the remaining outputs from a temporal cache. It determines task-refresh priorities by combining a calibrated age-regret lookup table with the grayscale frame difference accumulated since each task was last refreshed. The motion weight is selected from a finite, predefined candidate set using only calibration performance, requiring neither an additional predictor nor gradient training.

This work implements AgedNet-Motion on YOLOP, A-YOLOM, and YOLOPX and evaluates it on original BDD100K video streams. Relative to full refresh, it reduces mean model and post-processing latency by 42.19% on YOLOP, 51.54% on A-YOLOM, and 33.24% on YOLOPX. On YOLOP, it improves mean ground-truth quality over Age-Only by 0.453 percentage points, with a video-cluster bootstrap 95% confidence interval above zero. Its point estimate is also higher than that of Tuned Fixed, although this difference is not statistically confirmed. On A-YOLOM, calibration selects a zero motion weight, making AgedNet-Motion identical to Age-Only while exceeding Tuned Fixed by 0.922 percentage points. On YOLOPX, calibration selects an interior motion weight, and AgedNet-Motion exceeds Tuned Fixed by 0.940 percentage points with a confidence interval above zero, while its small difference from Age-Only is not statistically distinguishable in either direction. Across all

three models, AgedNet-Motion is never confirmed worse than the stronger non-learned baseline. These results demonstrate that AgedNet-Motion can reduce task-specific computation without additional scheduler training and can remove the motion multiplier, or select only a marginal one, when calibration does not support a large motion contribution; this produced the same schedule as Age-Only on A-YOLOM and a near-identical one on YOLOPX. YOLOPX also shares its backbone codebase with YOLOP rather than constituting a fully independent implementation, so the strength of the cross-architecture evidence in this work is graded: A-YOLOM provides an independent-codebase replication, while YOLOPX provides a same-lineage, different-detection-head replication.

Statements and Declarations

Funding. The authors received no financial support for the research, authorship, or publication of this article.

Competing Interests. The authors have no relevant financial or non-financial interests to disclose.

Data Availability. The BDD100K data analyzed in this study are available from the BDD100K project. The calibration and evaluation manifests and the additional experimental artifacts are available from the corresponding author on reasonable request.

Code Availability. The implementation and evaluation code are available from the corresponding author on reasonable request.

Author Contributions. Sungjun Jang contributed to conceptualization, methodology, software, validation, formal analysis, investigation, data curation, visualization, and writing of the original draft. Taegeun Oh contributed to supervision, validation, and review and editing of the manuscript. Both authors read and approved the final manuscript.

References

- [1] Aich A, Schuster S, Roy-Chowdhury AK, et al (2023) Efficient controllable multi-task architectures. In: 2023 IEEE/CVF International Conference on Computer Vision (ICCV), IEEE, pp 5717–5728
- [2] Burges C, Shaked T, Renshaw E, et al (2005) Learning to rank using gradient descent. In: Proceedings of the 22nd international conference on Machine learning, pp 89–96
- [3] Cao Z, Qin T, Liu TY, et al (2007) Learning to rank: from pairwise approach to listwise approach. In: Proceedings of the 24th international conference on Machine learning, pp 129–136

- [4] Crankshaw D, Wang X, Zhou G, et al (2017) Clipper: A low-latency online prediction serving system. In: 14th USENIX Symposium on Networked Systems Design and Implementation (NSDI 17), pp 613–627
- [5] Fan Z, Sarkar R, Jiang Z, et al (2022) M³vit: Mixture-of-experts vision transformer for efficient multi-task learning with model-accelerator co-design. *Advances in Neural Information Processing Systems* 35:28441–28457
- [6] Habibian A, Abati D, Cohen TS, et al (2021) Skip-convolutions for efficient video processing. *arXiv preprint arXiv:210411487*
- [7] Han C, Zhao Q, Zhang S, et al (2022) Yolopv2: Better, faster, stronger for panoptic driving perception. *arXiv preprint arXiv:220811434*
- [8] Han Y, Huang G, Song S, et al (2021) Dynamic neural networks: A survey. *IEEE transactions on pattern analysis and machine intelligence* 44(11):7436–7456
- [9] Jiao Z, Zhao X, Nong C, et al (2024) Yolopx: Anchor-free multi-task learning network for panoptic driving perception. *Pattern Recognition* 148:110152
- [10] Kaul S, Yates R, Gruteser M (2012) Real-time status: How often should one update? In: 2012 Proceedings IEEE INFOCOM, IEEE, pp 2731–2735
- [11] Li Y, Shi J, Lin D (2018) Low-latency video semantic segmentation. In: 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, IEEE, pp 5997–6005
- [12] Li Y, Padmanabhan A, Zhao P, et al (2020) Reducto: On-camera filtering for resource-efficient real-time video analytics. In: Proceedings of the Annual conference of the ACM Special Interest Group on Data Communication on the applications, technologies, architectures, and protocols for computer communication, pp 359–376
- [13] Lucki J, Becktor J, Georgakis G, et al (2025) Visual perception engine: Fast and flexible multi-head inference for robotic vision tasks. *arXiv preprint arXiv:250811584*
- [14] Meng Y, Lin CC, Panda R, et al (2020) Ar-net: Adaptive frame resolution for efficient action recognition. In: European conference on computer vision, Springer, pp 86–104
- [15] Neseem M, Agiza A, Reda S (2023) Adamtl: Adaptive input-dependent inference for efficient multi-task learning. In: 2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), IEEE, pp 4730–4739
- [16] ONNX Runtime developers (2018) Onnx runtime. <https://github.com/microsoft/onnxruntime>

- [17] Reddi VJ, Cheng C, Kanter D, et al (2020) Mlperf inference benchmark. In: 2020 ACM/IEEE 47th Annual International Symposium on Computer Architecture (ISCA), IEEE, pp 446–459
- [18] Sagduyu YE, Ulukus S, Yener A (2023) Age of information in deep learning-driven task-oriented communications. In: IEEE INFOCOM 2023-IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS), IEEE, pp 1–6
- [19] Shelhamer E, Rakelly K, Hoffman J, et al (2016) Clockwork convnets for video semantic segmentation. In: European Conference on Computer Vision, Springer, pp 852–868
- [20] Shisher MKC, Sun Y (2022) How does data freshness affect real-time supervised learning? In: Proceedings of the Twenty-Third International Symposium on Theory, Algorithmic Foundations, and Protocol Design for Mobile Networks and Mobile Computing, pp 31–40
- [21] Shisher MKC, Sun Y, Hou IH (2024) Timely communications for remote inference. *IEEE/ACM Transactions on Networking* 32(5):3824–3839
- [22] Teerapittayanon S, McDanel B, Kung HT (2016) Branchynet: Fast inference via early exiting from deep neural networks. In: 2016 23rd international conference on pattern recognition (ICPR), IEEE, pp 2464–2469
- [23] Wang J, Wu QJ, Zhang N (2024) You only look at once for real-time and generic multi-task. *IEEE Transactions on Vehicular Technology* 73(9):12625–12637
- [24] Wang X, Yu F, Dou ZY, et al (2018) Skipnet: Learning dynamic routing in convolutional networks. In: European Conference on Computer Vision, Springer, pp 420–436
- [25] Wooders S, Mo X, Narang A, et al (2023) Ralf: Accuracy-aware scheduling for feature store maintenance. *Proceedings of the VLDB Endowment* 17(3):563–576
- [26] Wu D, Liao MW, Zhang WT, et al (2022) Yolop: You only look once for panoptic driving perception. *Machine Intelligence Research* 19(6):550–562
- [27] Wu Z, Nagarajan T, Kumar A, et al (2018) Blockdrop: Dynamic inference paths in residual networks. In: 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, IEEE, pp 8817–8826
- [28] Yates RD, Sun Y, Brown DR, et al (2021) Age of information: An introduction and survey. *IEEE Journal on Selected Areas in Communications* 39(5):1183–1210
- [29] Yu F, Chen H, Wang X, et al (2020) Bdd100k: A diverse driving dataset for heterogeneous multitask learning. In: 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pp 2633–2642, <https://doi.org/10.1109/CVPR42600.2020.00271>

- [30] Zhu X, Xiong Y, Dai J, et al (2017) Deep feature flow for video recognition. In: 2017 IEEE conference on computer vision and pattern recognition (CVPR), IEEE, pp 4141–4150