Resource-efficient Video Analytics on the Extreme Edge

Remembrall

Murali Ramanujam, Yinwei Dai, Kyle Jamieson, and Ravi Netravali

Video analytics

Classification, detection, segmentation

Video analytics

Classification, detection, segmentation

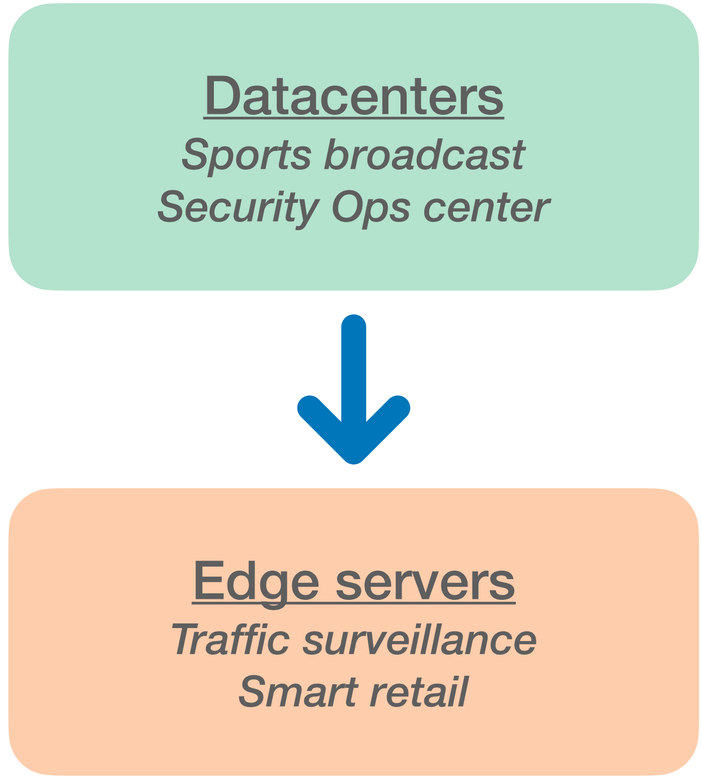
Datacenters
Sports broadcast
Security Ops center

Edge servers
Traffic surveillance
Smart retail

SoC-based mobile edge
Autonomous mobility
Search-and-rescue

Edge Challenges

Smaller models → repeated adaptation



Edge Challenges

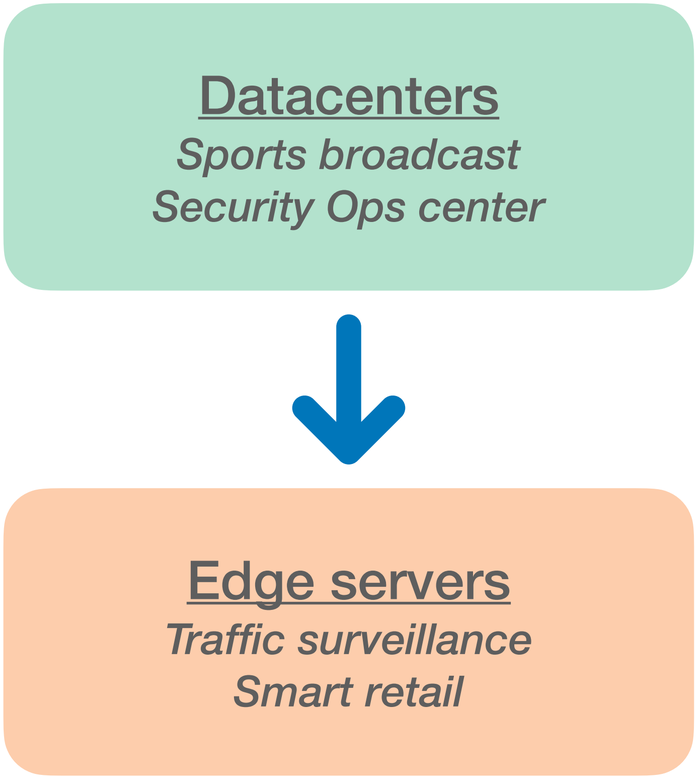
Smaller models → repeated adaptation

Restrictions when moving from Datacenters to Edge

Resource shrink mandates smaller models
e.g., ResNet50 instead of ResNet101

Vulnerable to scene drift
e.g., rush hour, weather changes → accuracy drops

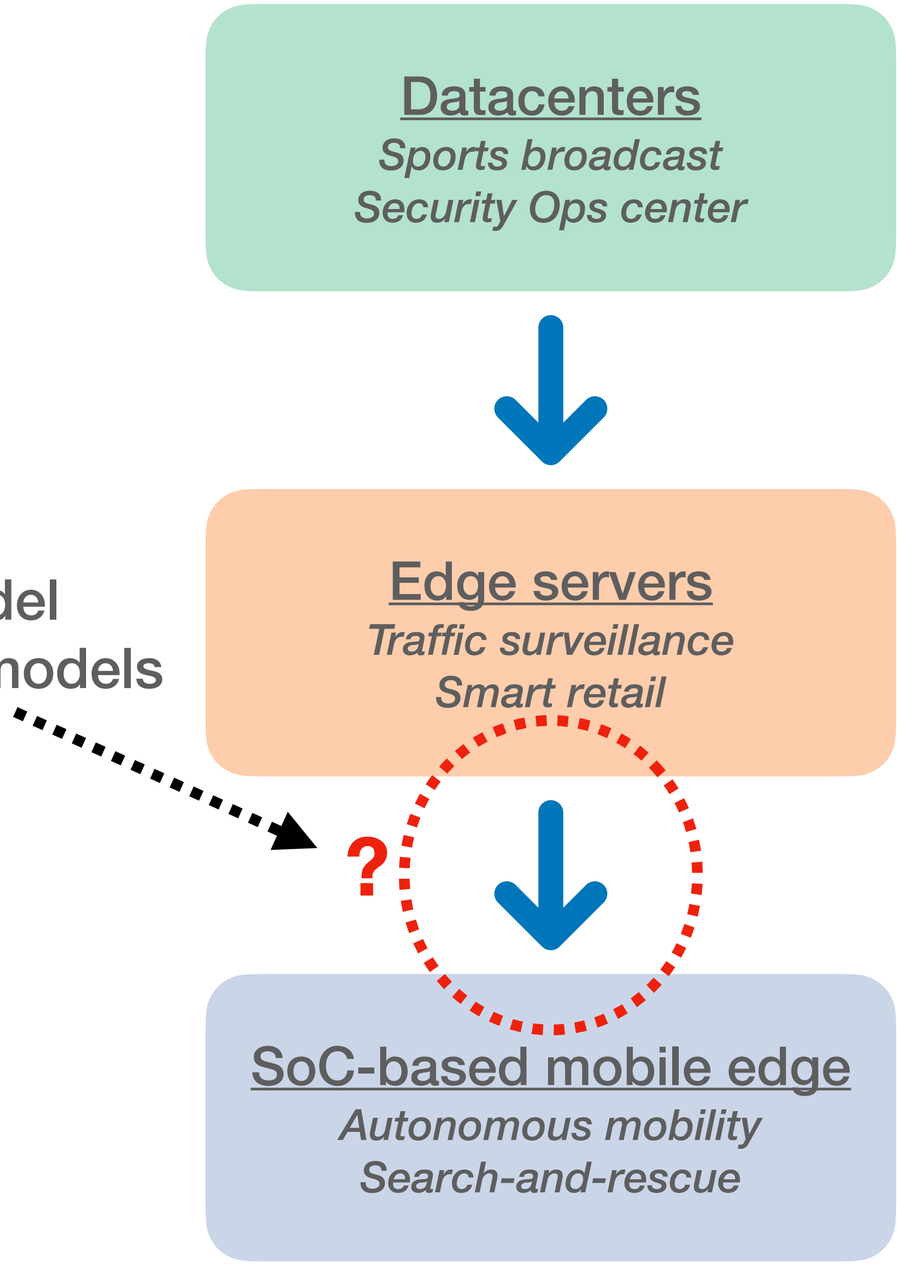
Approach 1: Repeatedly train from most recent model
Approach 2: Store and reuse across many specialized models



Edge Challenges

Smaller models → repeated adaptation

Approach 1: Repeatedly train from most recent model
Approach 2: Store and reuse across many specialized models

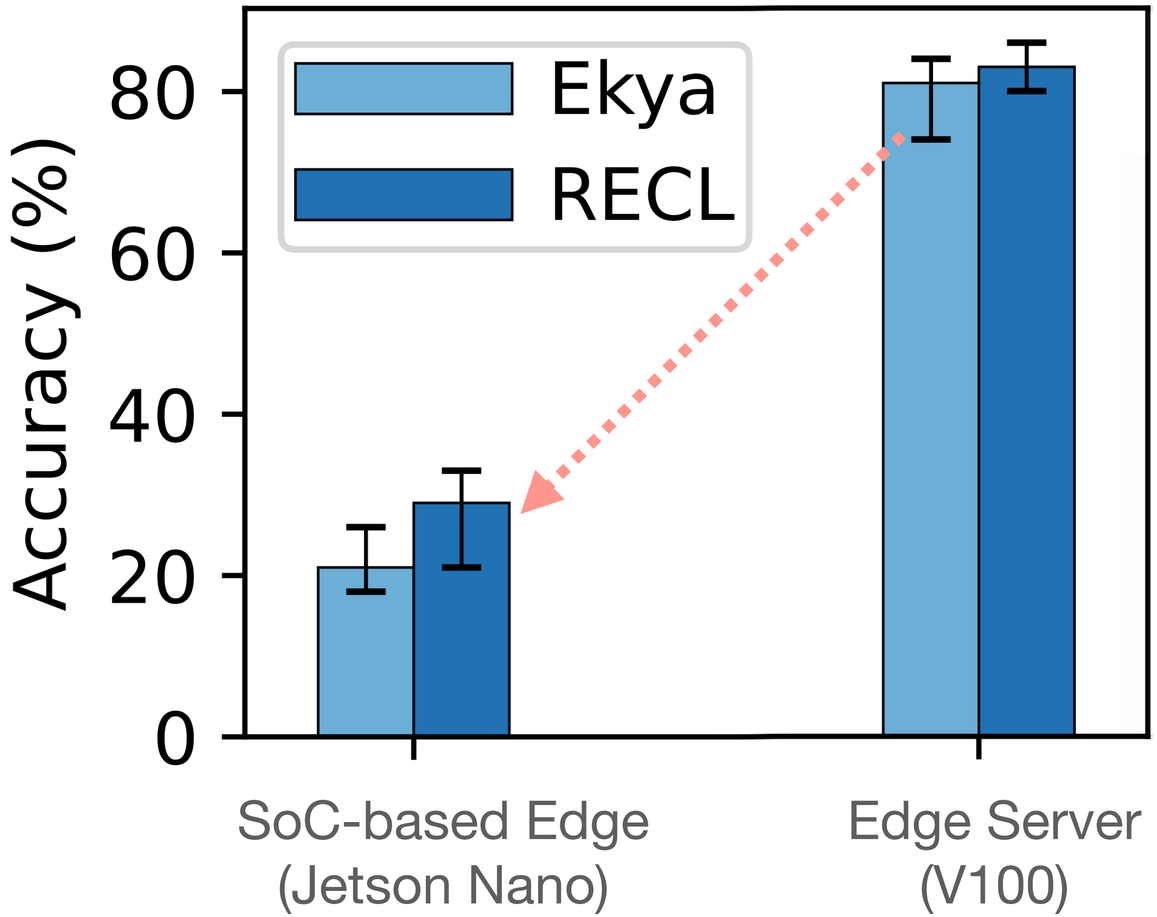


The Problem: Handling Drift

Existing approaches break down at the SoC-based edge

Approach 1: Periodically retrain the latest model used for inference with fresh data samples; e.g., Ekya (NSDI ’22).

Approach 2: Maintaining trained models in a zoo, and reusing the best candidate for each scene. The chosen model is then fine-tuned on recent data samples; e.g., RECL (NSDI ’23)



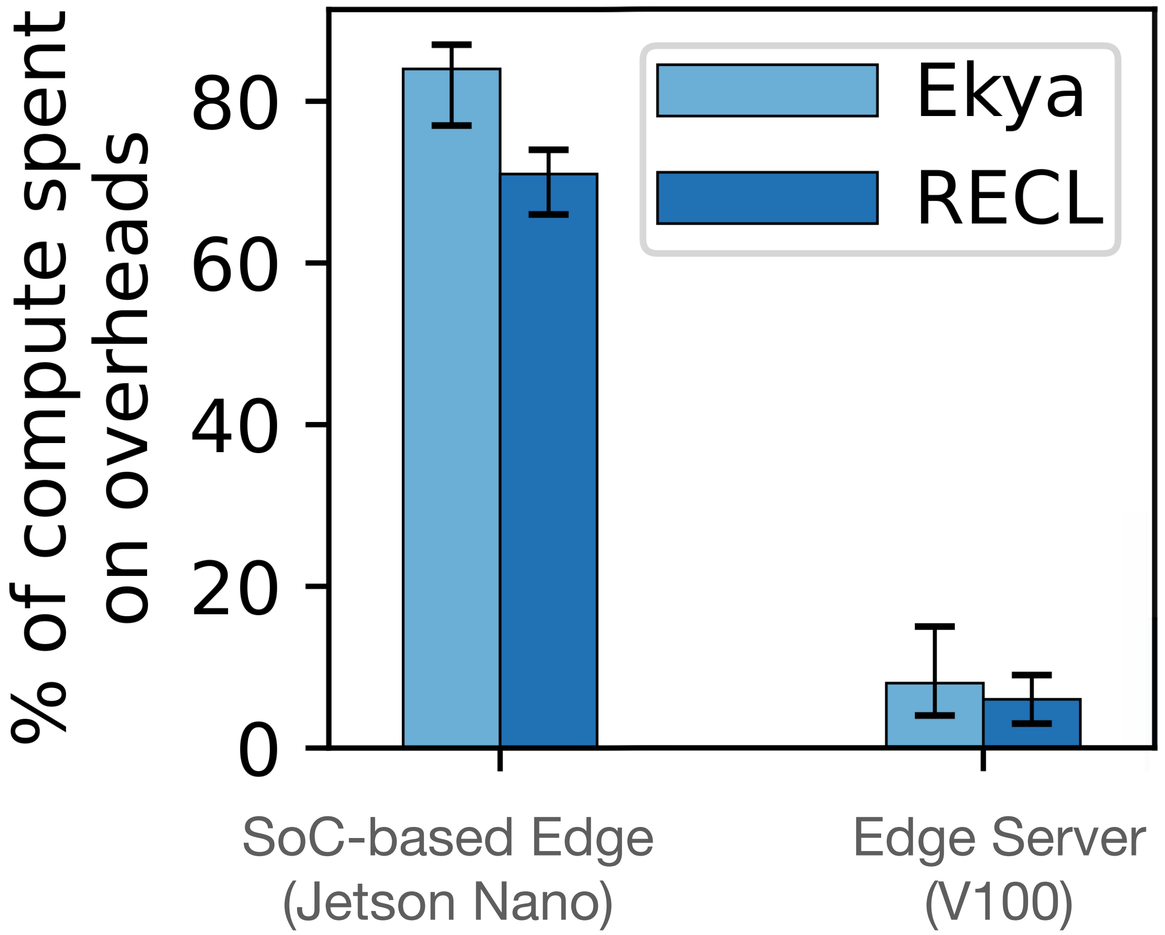
50 hours of drone/dashcam footage from YouTube.
Classification with ResNet18.
Bar shows median, errors span 25-75th percentile.

Why do existing approaches break down?

Compute overheads starve fruitful inference

Approach 1 Overheads: As resources shrink, Ekya's retraining consumes a growing share of GPU.

Approach 2 Overheads: Each time a new model is added to the zoo, RECL must retrain its gating network (used to select the best model from the zoo).



Why do overheads dominate compute?

The why behind the why: SoCs are intrinsically different.

Why do overheads dominate compute?

The why behind the why: SoCs are intrinsically different.

Size, power and thermal constraints → SoC integrates CPU, GPU and Memory on to a single die → SoCs are compute-bound, not memory-bound

Why do overheads dominate compute?

The why behind the why: SoCs are intrinsically different.

Central distinction from the discrete GPUs used in prior edge analytics work

Size, power and thermal constraints → SoC integrates CPU, GPU and Memory on to a single die → SoCs are compute-bound, not memory-bound

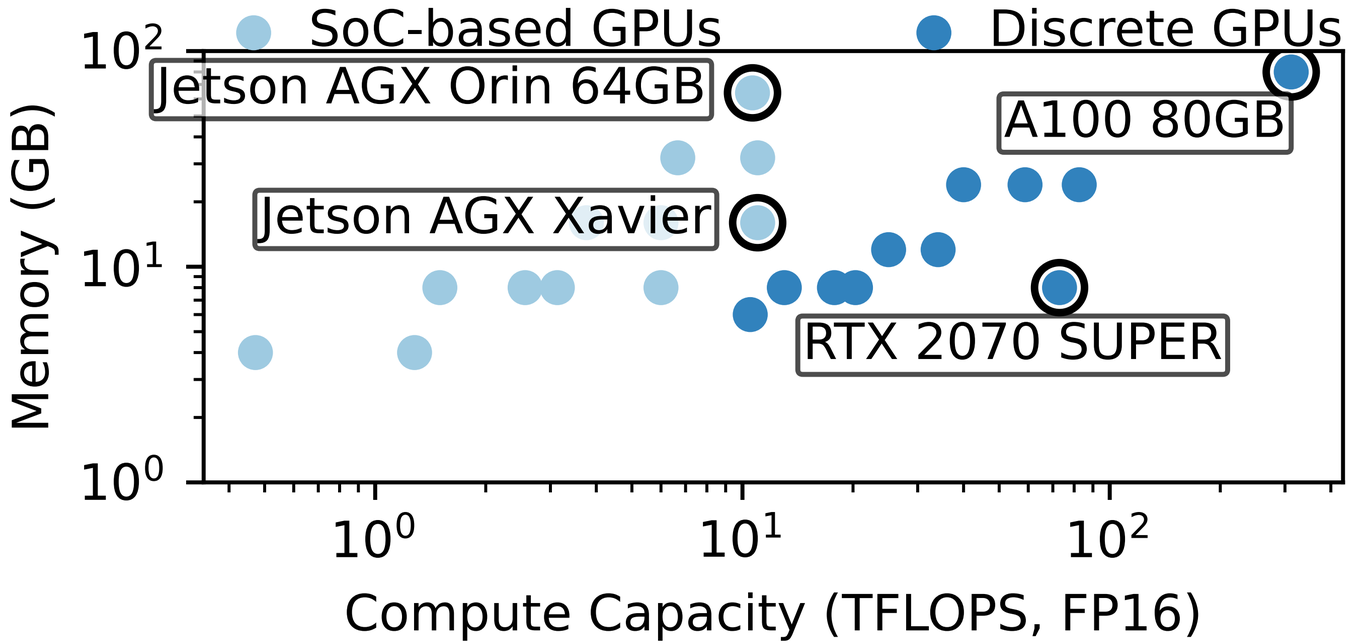
Why do overheads dominate compute?

The why behind the why: SoCs are intrinsically different.

Size, power and thermal constraints → SoC integrates CPU, GPU and Memory on to a single die → SoCs are compute-bound, not memory-bound

Central distinction from the discrete GPUs used in prior edge analytics work

SoCs often have 10× less compute than discrete GPUs, but relatively large memory pools →



Why do overheads dominate compute?

The why

Current approaches naively use memory and stress compute to tame drift. A poor fit for SoCs.

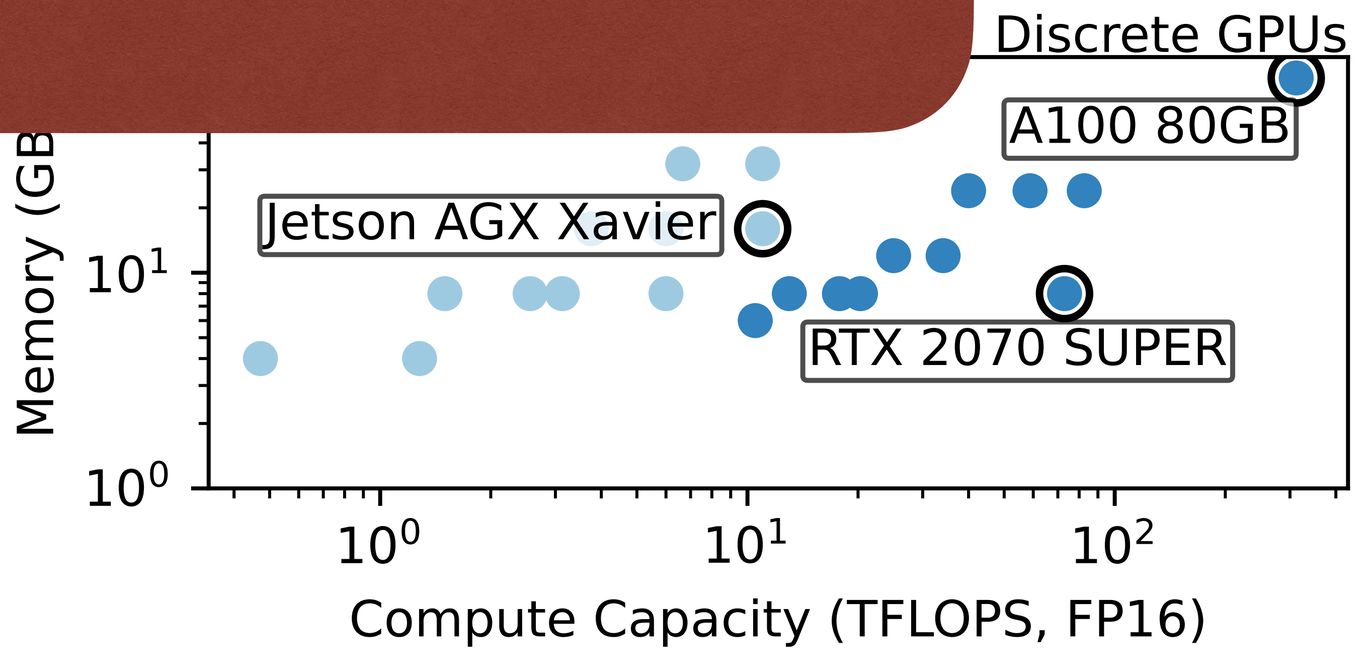
Remembrall: principled use of (relatively abundant) memory.

om the discrete
ge analytics work

Size, pow
co

ute-bound,
-bound

SoCs often have 10× less compute than discrete GPUs, but relatively large memory pools →



Why do overheads dominate compute?

The why

Current approaches naively use memory and stress compute to tame drift. A poor fit for SoCs.

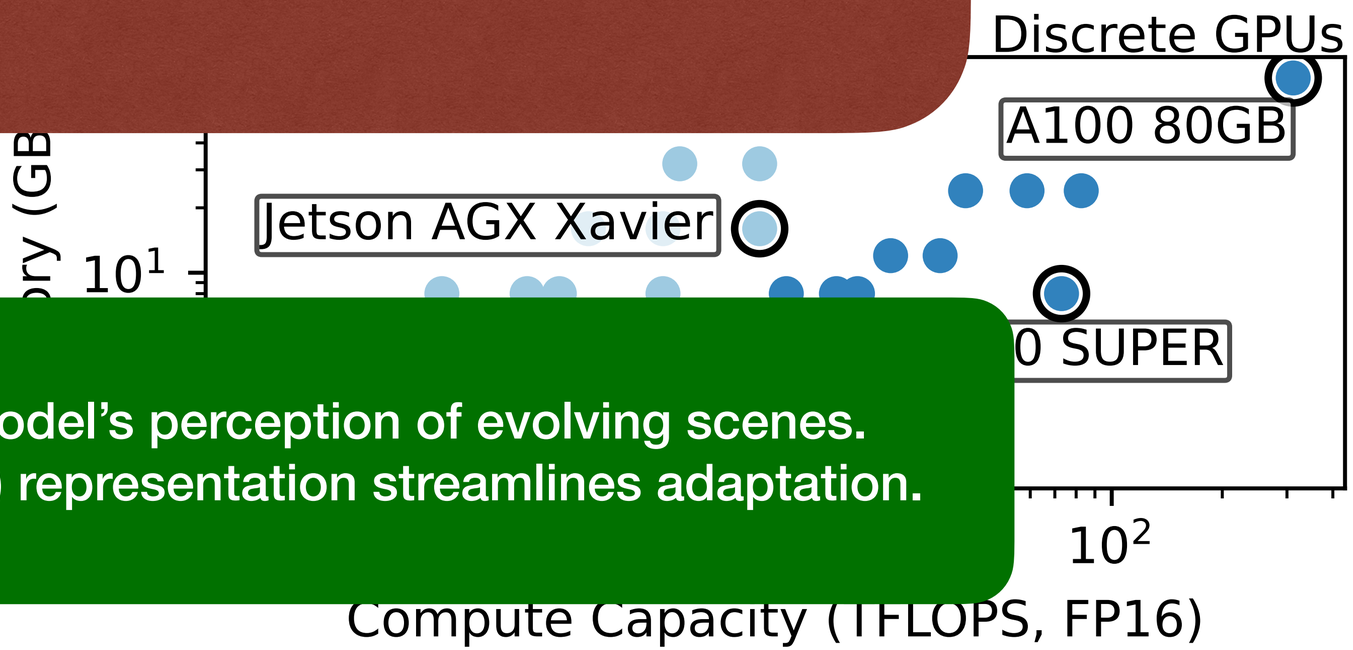
Remembrall: principled use of (relatively abundant) memory.

om the discrete
ge analytics work

Size, pow
co

ute-bound,
-bound

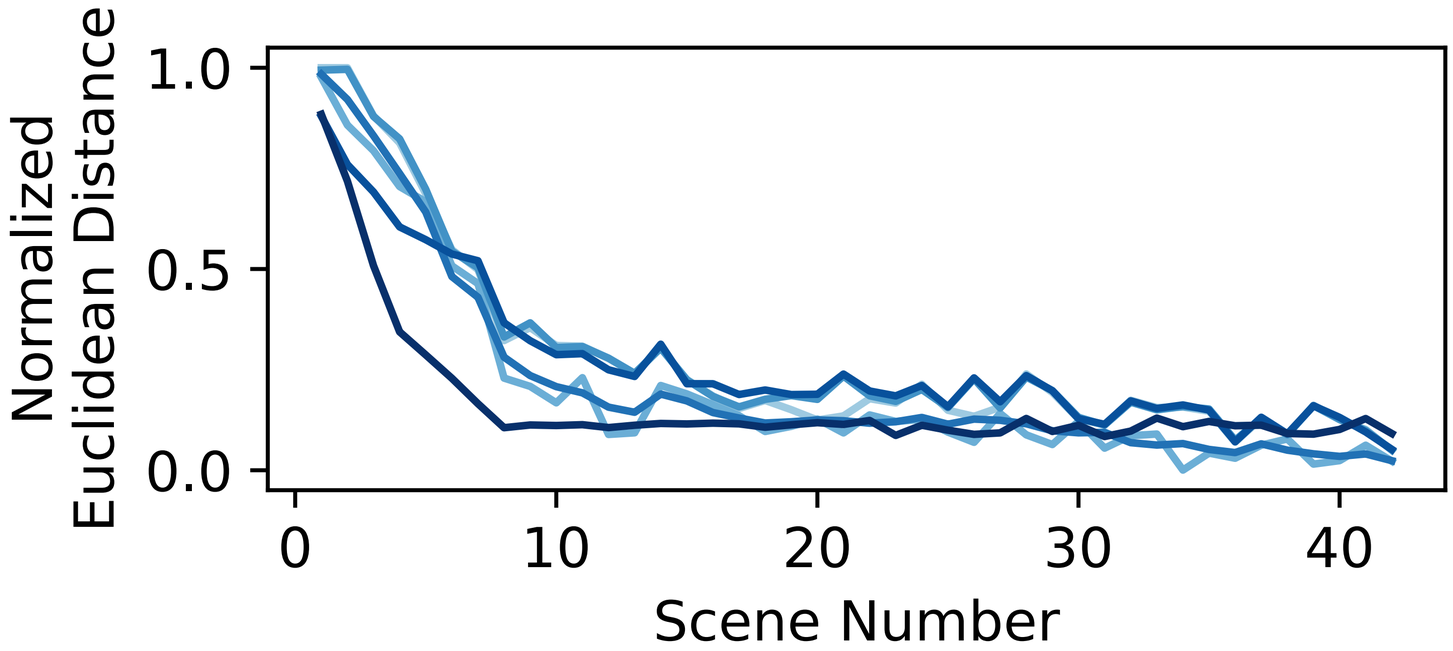
SoCs often have 10× less compute than discrete GPUs, but relatively large memory pools →



Key Idea: Strong stability in model's perception of evolving scenes. Starting from this stable (shared) representation streamlines adaptation.

Studying drift from the model's perspective

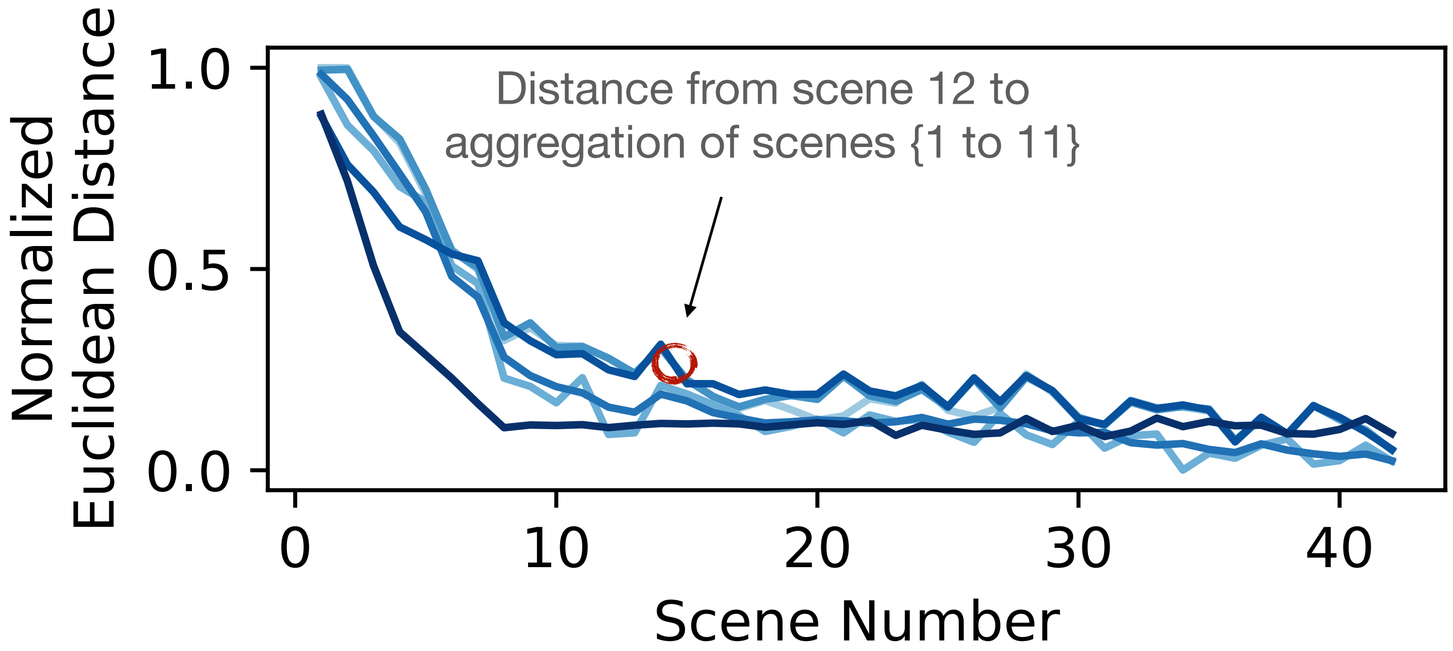
Evolving scenes exhibit stability in their embeddings



Videos are split into 3-minute scenes. 5 videos.
Compute per-scene centroid from ResNet18 penultimate-layer object embeddings.

Studying drift from the model's perspective

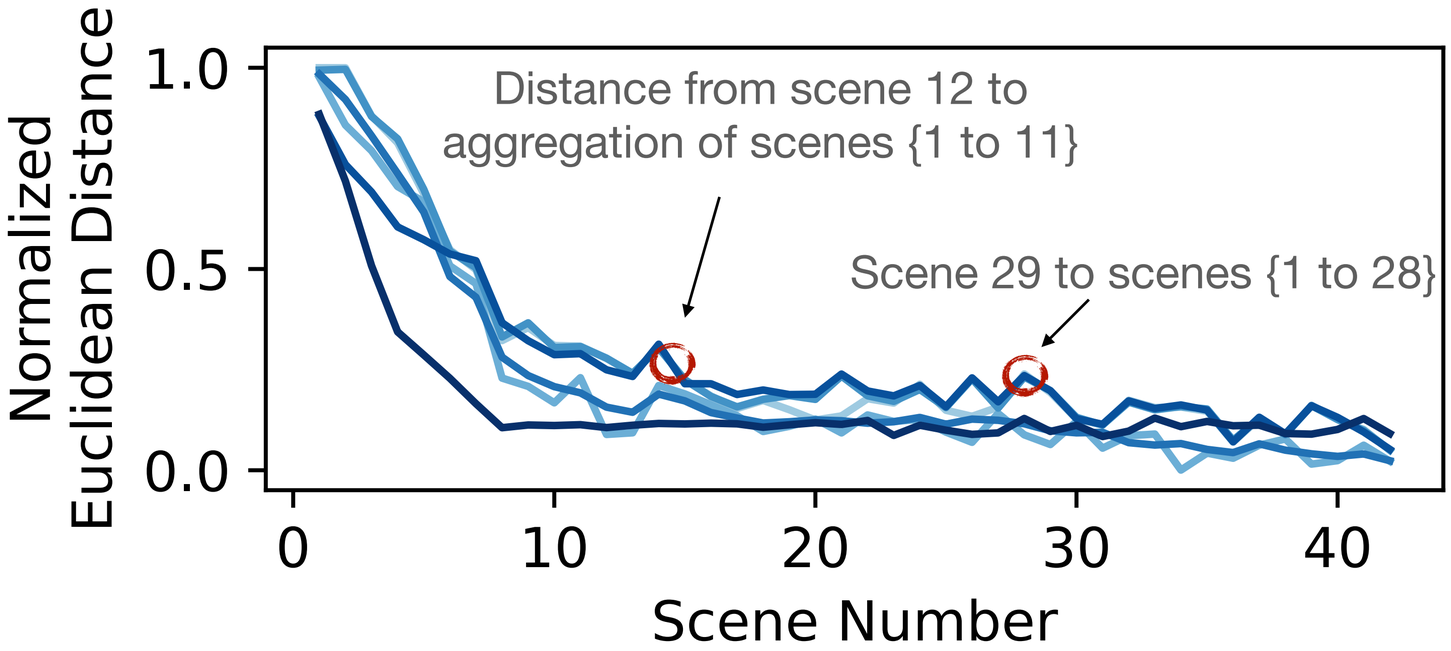
Evolving scenes exhibit stability in their embeddings



Videos are split into 3-minute scenes. 5 videos.
Compute per-scene centroid from ResNet18 penultimate-layer object embeddings.

Studying drift from the model's perspective

Evolving scenes exhibit stability in their embeddings



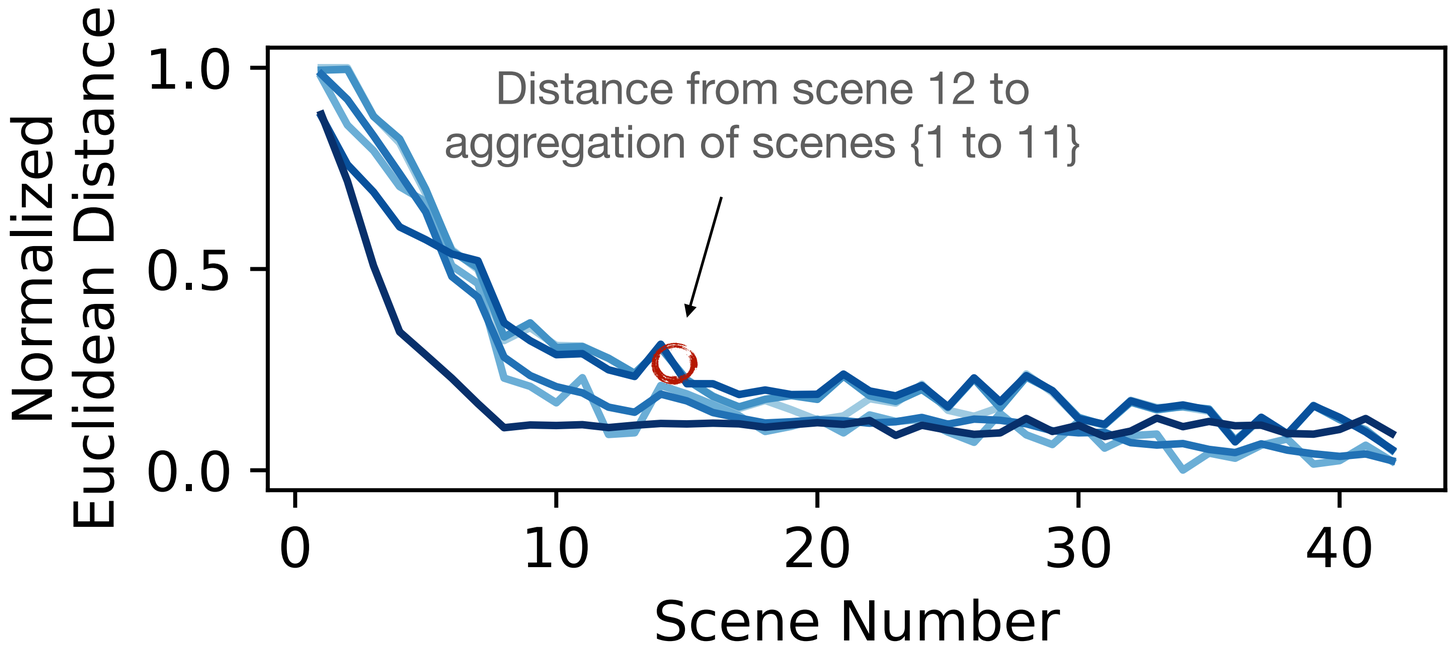
Videos are split into 3-minute scenes. 5 videos. Compute per-scene centroid from ResNet18 penultimate-layer object embeddings.

Studying drift from the model's perspective

Evolving scenes exhibit stability in their embeddings

Distance from each scene's centroid to centroid of all prior scenes steadily decreases, then plateaus

Indicates representation saturation: the model's internal view converges despite ongoing content drift



Videos are split into 3-minute scenes. 5 videos. Compute per-scene centroid from ResNet18 penultimate-layer object embeddings.

Why do embeddings converge

.. despite visual drift?

Analyzed adjacent scene pairs using

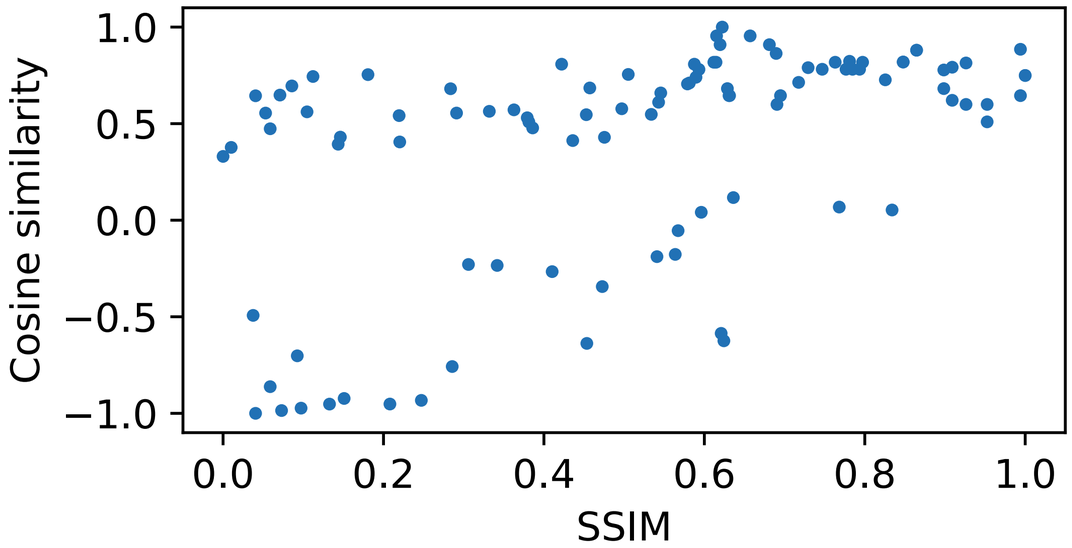
- **Cosine similarity** of ResNet18 embeddings
- **Pixel-level similarity** via SSIM

Why do embeddings converge

.. despite visual drift?

Analyzed adjacent scene pairs using

- **Cosine similarity** of ResNet18 embeddings
- **Pixel-level similarity** via SSIM

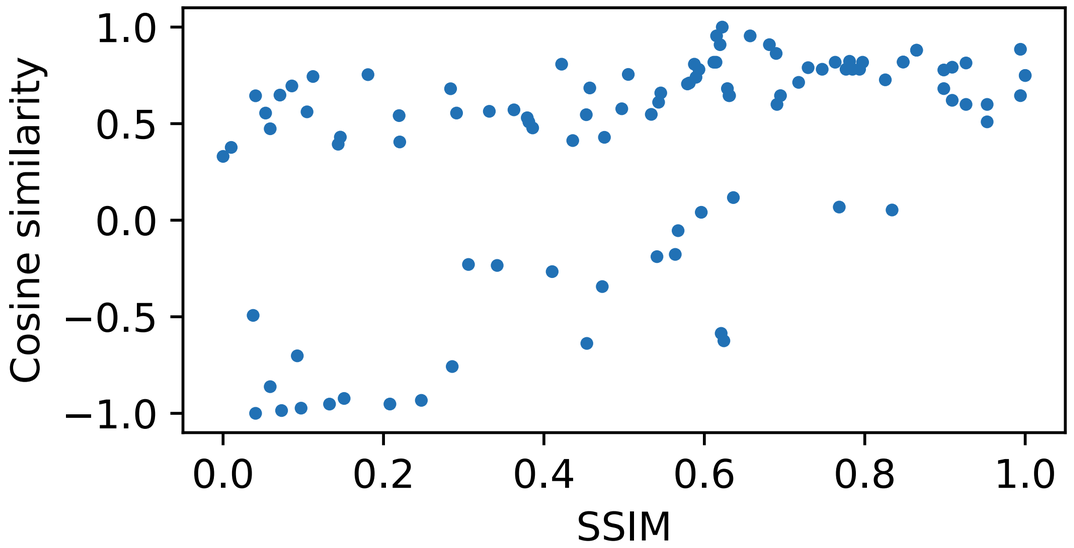


Low SSIM ≠ Low embedding similarity
→ Embeddings remain similar even across visually distinct scenes

Why do embeddings converge

.. despite visual drift?

Analyzed adjacent scene pairs using
- **Cosine similarity** of ResNet18 embeddings
- **Pixel-level similarity** via SSIM



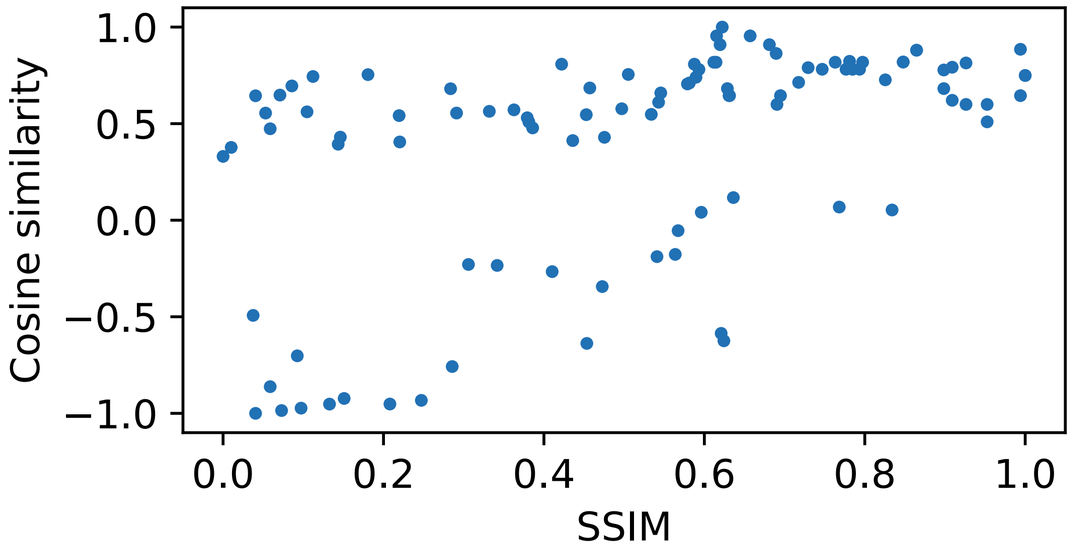
Low SSIM ≠ Low embedding similarity
→ Embeddings remain similar even across visually distinct scenes

Numbers indicate average cosine similarity between objects, which ranges from -1 (opposite) to 1 (identical) →

Why do embeddings converge

.. despite visual drift?

Analyzed adjacent scene pairs using
- **Cosine similarity** of ResNet18 embeddings
- **Pixel-level similarity** via SSIM



Low SSIM ≠ Low embedding similarity
→ Embeddings remain similar even across visually distinct scenes

Root cause: Models are trained to prioritize **structural cues** (e.g., geometry, layout) over surface-level features

Numbers indicate average cosine similarity between objects, which ranges from -1 (opposite) to 1 (identical) →

Embedding convergence reduces retraining costs

Diverse scenes map to similar embeddings → their specialized models must also remain close

Embedding convergence reduces retraining costs

Diverse scenes map to similar embeddings → their specialized models must also remain close

Idea: The shared structural cues could serve as an efficient starting point to train for all scenes

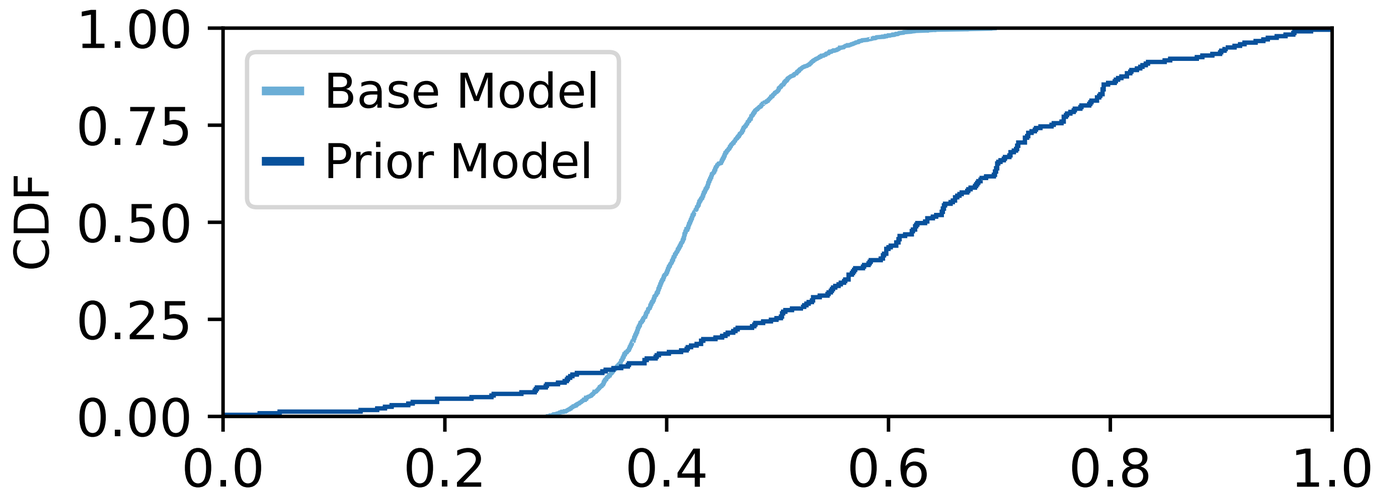
Lightweight retraining with few samples from a *base model* that incorporates this shared structure.

Embedding convergence reduces retraining costs

Diverse scenes map to similar embeddings → their specialized models must also remain close

Idea: The shared structural cues could serve as an efficient starting point to train for all scenes

Lightweight retraining with few samples from a *base model* that incorporates this shared structure.



Normalized Euclidean Distance from each scene's specialized model to (i) prior scene's model (Ekya-style), and (ii) a base model (aggregated centroid) of prior scenes

Embedding convergence reduces retraining costs

Diverse scenes map to similar embeddings → their specialized models must also remain close

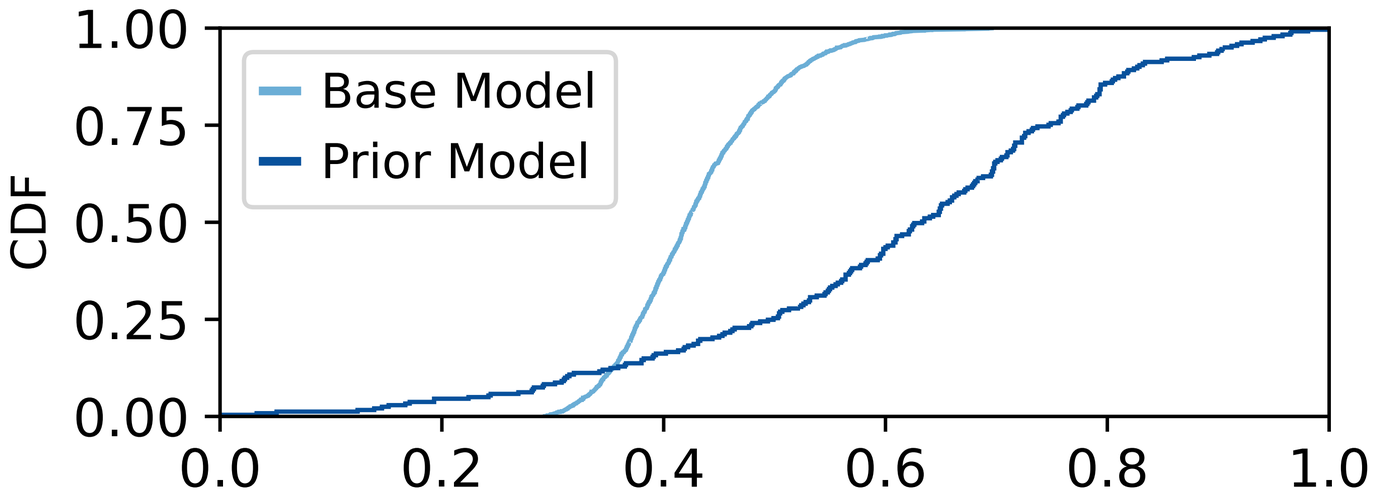
Idea: The shared structural cues could serve as an efficient starting point to train for all scenes

Lightweight retraining with few samples from a *base model* that incorporates this shared structure.

Base model consistently lies closer to the next specialized model than the previous one does.

Prior specialized models often overfit to scene-specific idiosyncrasies (e.g., transient lighting, rare classes), which must be unlearned first.

In contrast, starting from a base model that preserves general structure requires smaller weight updates, making retraining faster.



Normalized Euclidean Distance from each scene's specialized model to (i) prior scene's model (Ekya-style), and (ii) a base model (aggregated centroid) of prior scenes

Remembrall's Approach

Maintain two models with same architecture (e.g., ResNet18):
Base Model and Specialized Model

Remembrall's Approach

Maintain two models with same architecture (e.g., ResNet18):
Base Model and Specialized Model

For each new scene:
Initialize specialized model from base and fine-tune (few-shot) to scene-specifics

Remembrall's Approach

Maintain two models with same architecture (e.g., ResNet18):
Base Model and Specialized Model

For each new scene:
Initialize specialized model from base and fine-tune (few-shot) to scene-specifics

After fine-tuning: Base model is refined to improve generality over time

Remembrall's Approach

Maintain two models with same architecture (e.g., ResNet18):
Base Model and Specialized Model

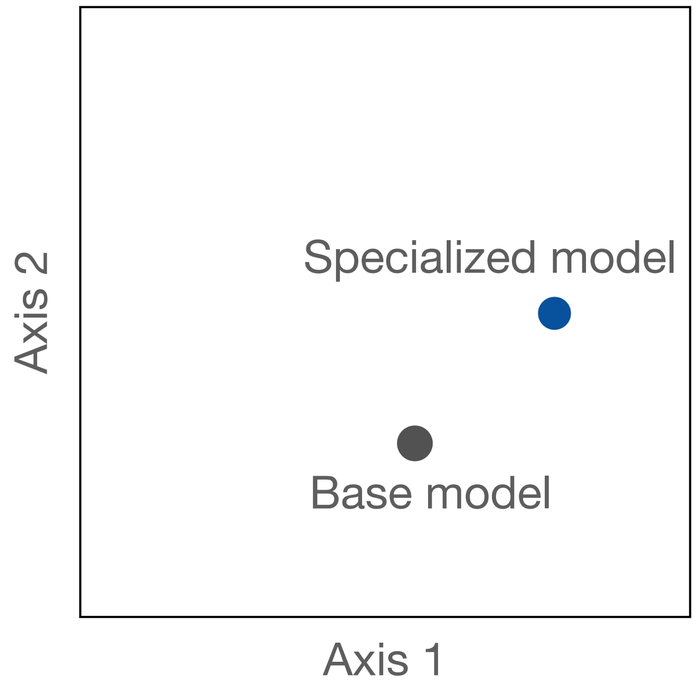
For each new scene:
Initialize specialized model from base and fine-tune (few-shot) to scene-specifics

After fine-tuning: Base model is refined to improve generality over time

Tradeoff memory (two models) for lower compute costs when adapting to drift
Caters to SoC's unique resource profile

Studying Remembrall's models

This slide provides background using illustrative model coordinates.

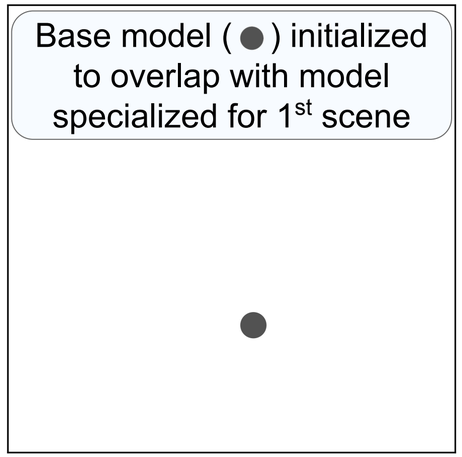


- We represent a model as a point in 2D plane, visualized by averaging ResNet18 penultimate-layer embeddings across detected objects.
- Embeddings are projected into 2D using Principal Component Analysis (PCA); axes correspond to the top two principal components.
- Absolute positions are not meaningful (axes are unlabeled), distances between points reflect changes in the model's internal representation.

Studying Remembrall's models

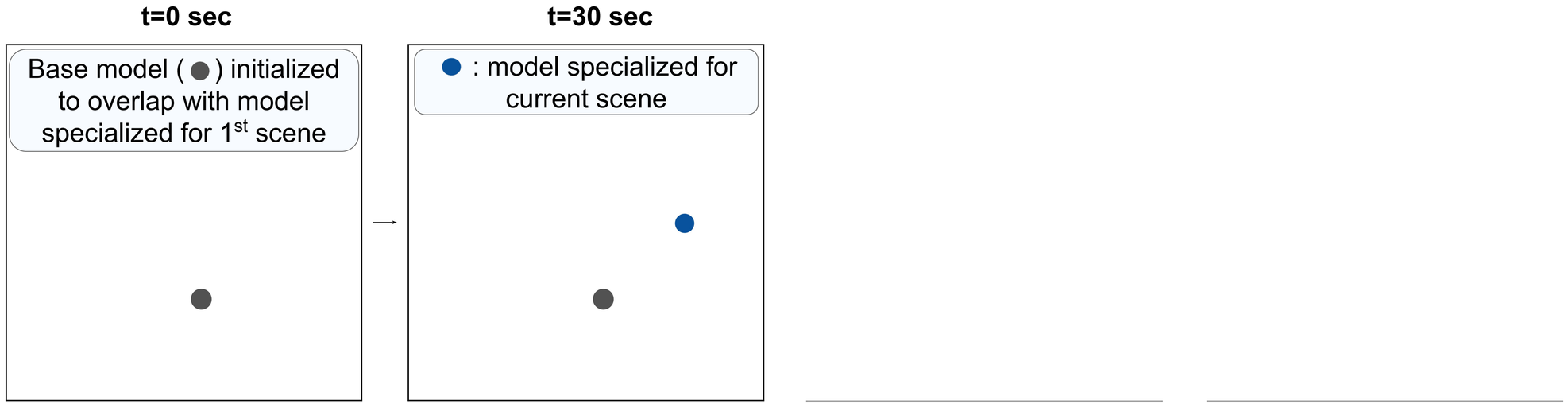
Evolution of models on an exemplar 90 min drone video stream.

t=0 sec



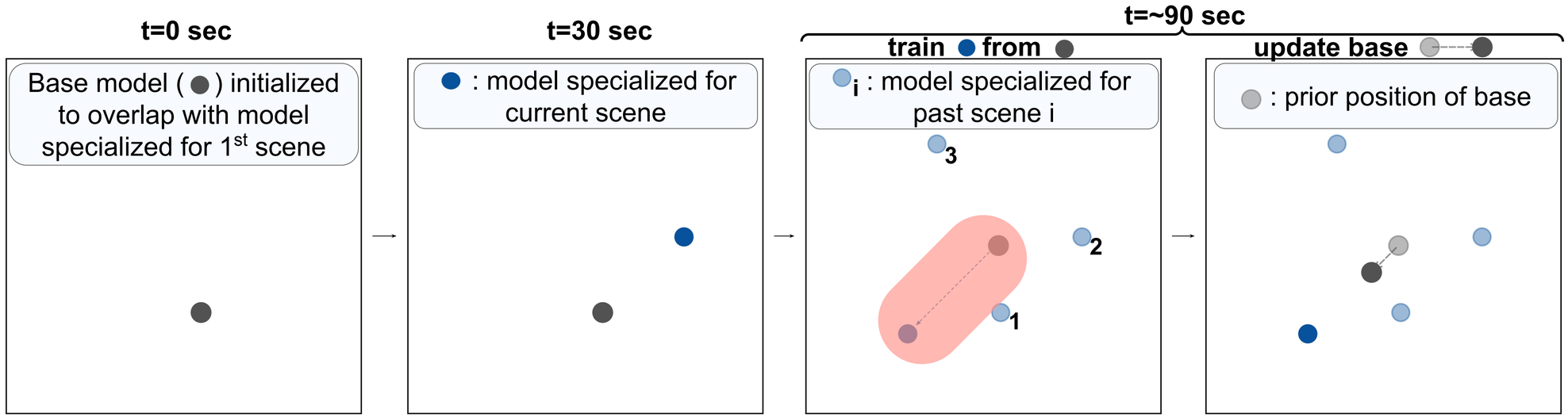
Studying Remembrall's models

Evolution of models on an exemplar 90 min drone video stream.



Studying Remembrall's models

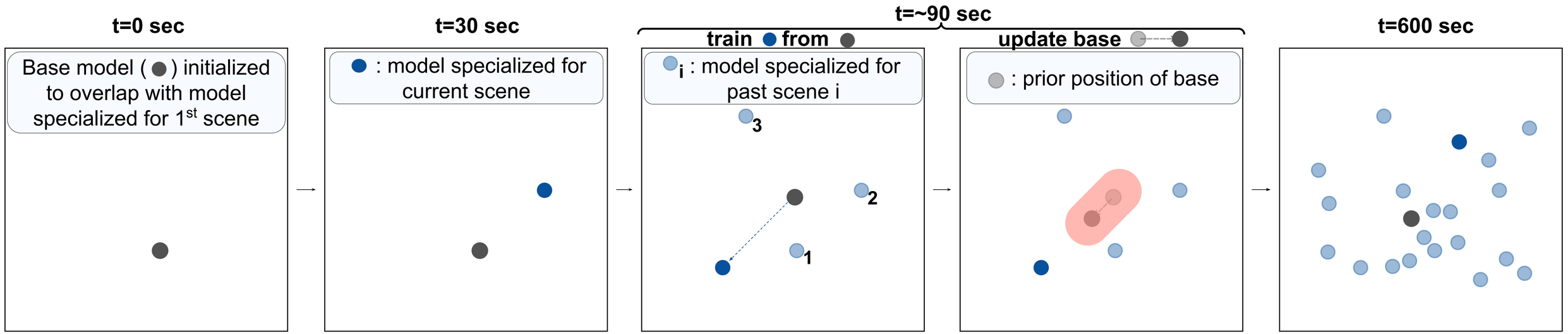
Evolution of models on an exemplar 90 min drone video stream.



By **t=90 sec**, the ● is already in a "good position", as ● → ● (Remembrall-style) is shorter than 3 → ● (training with Ekya-style continual learning).

Studying Remembrall's models

Evolution of models on an exemplar 90 min drone video stream.

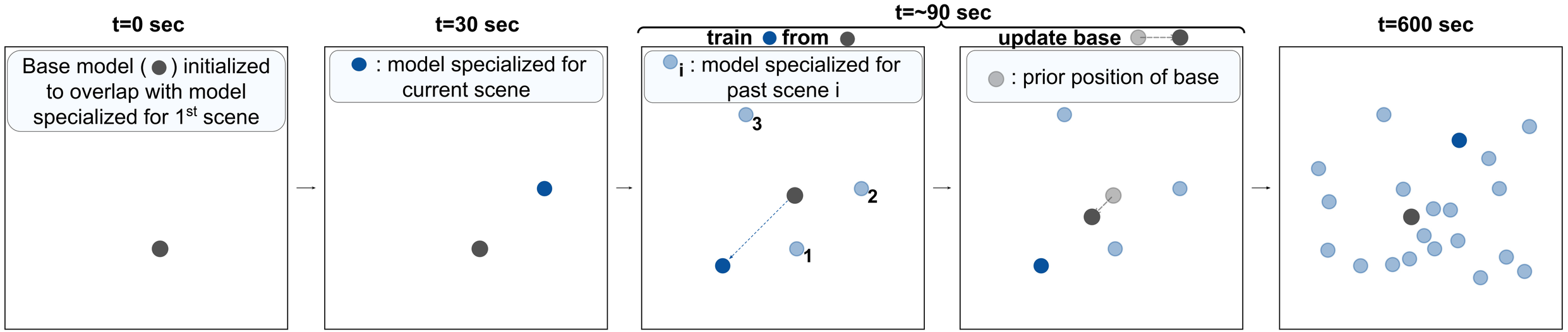


By **t=90 sec**, the ● is already in a "good position", as ● → ● (Remembrall-style) is shorter than $_3$ → ● (training with Ekya-style continual learning).

The base steadily drifts toward the center of the specialized distribution, accumulating cross-scene structure and becoming a stronger initializer for future scenes.

Studying Remembrall's models

Evolution of models on an exemplar 90 min drone video stream.



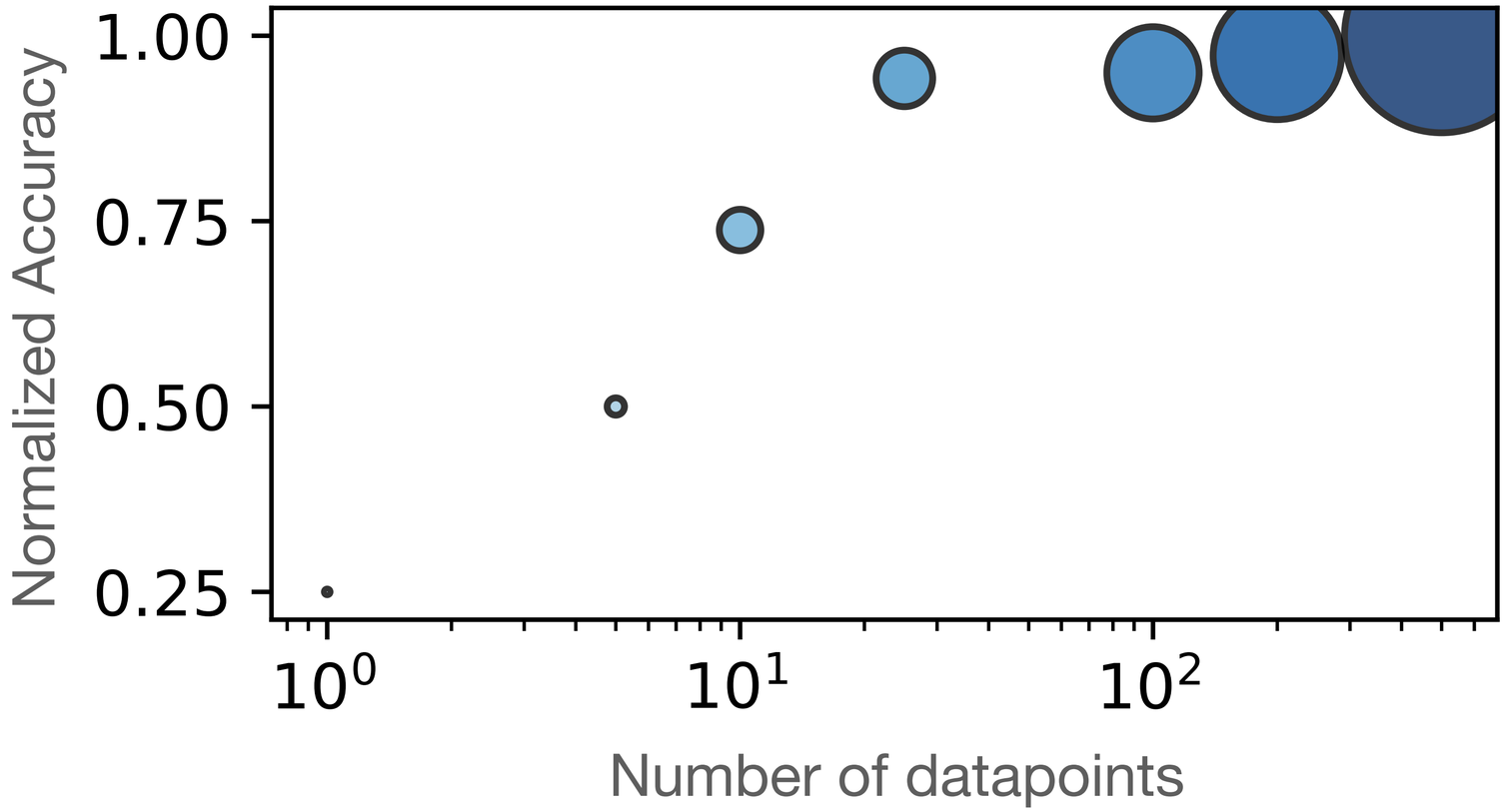
By **t=90 sec**, the ● is already in a "good position", as ● → ● (Remembrall-style) is shorter than 3 → ● (training with Ekya-style continual learning).

The base steadily drifts toward the center of the specialized distribution, accumulating cross-scene structure and becoming a stronger initializer for future scenes.

Remembrall is a natural instance of meta learning (learning-to-learn). Each scene is treated as a task and the base model serves as a learned initialization for fast per-task adaptation.

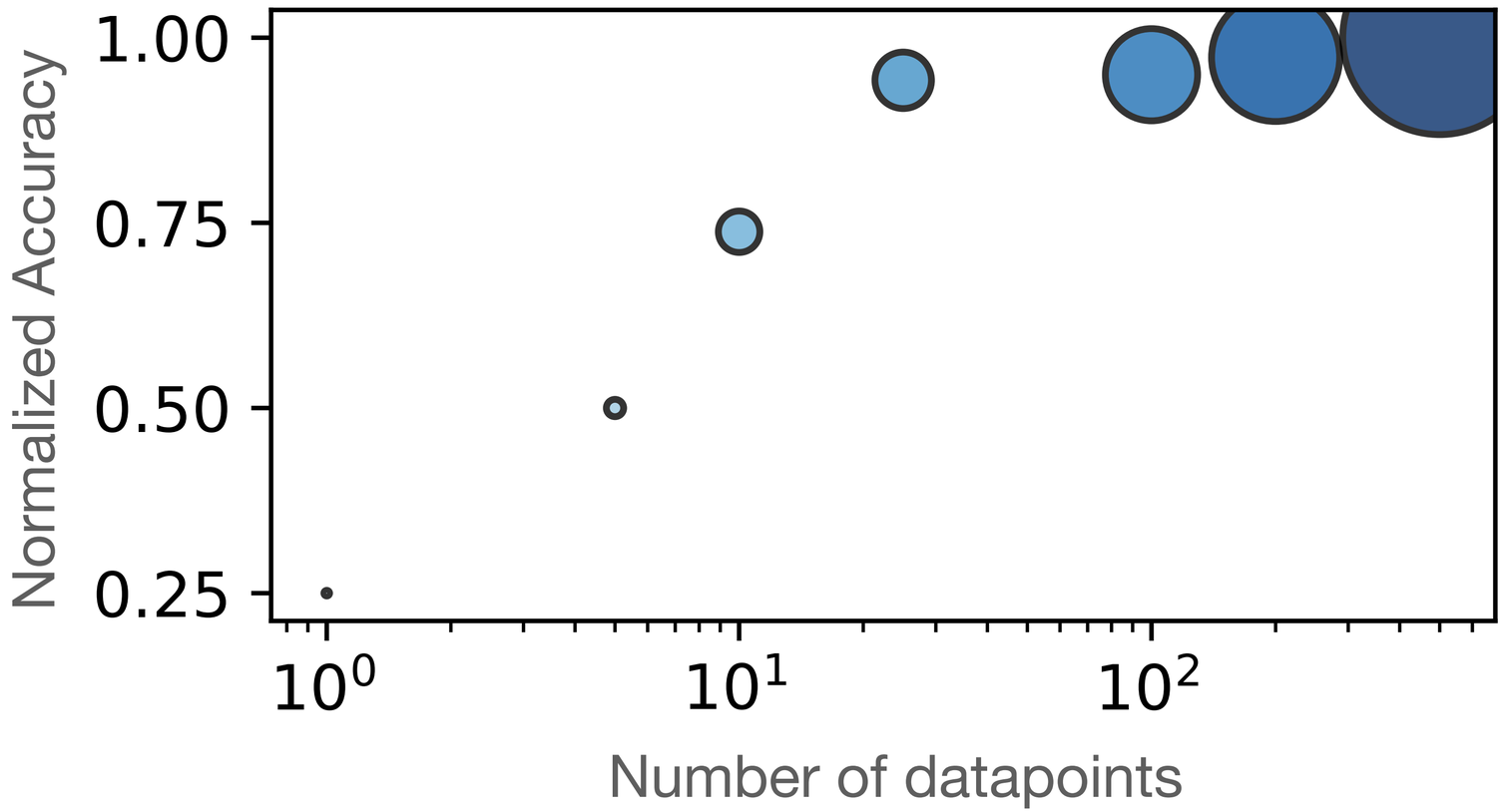
Challenge 1: Selecting training datapoints

- Post-retraining accuracy and retraining time as a function of number of datapoints.
  - Circle area denotes relative retraining time.
  - ResNet18 classification, medians shown here from training on 1hr drone video.



Challenge 1: Selecting training datapoints

- Post-retraining accuracy and retraining time as a function of number of datapoints.
  - Circle area denotes relative retraining time.
  - ResNet18 classification, medians shown here from training on 1hr drone video.
- Diminishing accuracy wins from larger training sets.
- Key challenge: right training datapoints.



Remembrall: Selecting training datapoints 1/2

- Prior methods eliminate redundant datapoints using visual heuristics (e.g., frame differencing, optical flow)
  - **brittle** or **too compute-intensive** for mobile settings

Remembrall: Selecting training datapoints 1/2

- Prior methods eliminate redundant datapoints using visual heuristics (e.g., frame differencing, optical flow)
  - **brittle** or **too compute-intensive** for mobile settings
- Idea: the model's own embeddings capture semantic similarity (e.g., shape, texture, layout)
  - a principled, low-cost way to detect redundant (proximity in Euclidean space) datapoints

Remembrall: Selecting training datapoints 1/2

- Prior methods eliminate redundant datapoints using visual heuristics (e.g., frame differencing, optical flow)
  - **brittle** or **too compute-intensive** for mobile settings
- Idea: the model's own embeddings capture semantic similarity (e.g., shape, texture, layout)
  - a principled, low-cost way to detect redundant (proximity in Euclidean space) datapoints
- Challenge**:** Recomputing embeddings using the base model adds compute overhead

Remembrall: Selecting training datapoints 1/2

- Prior methods eliminate redundant datapoints using visual heuristics (e.g., frame differencing, optical flow)
  - **brittle** or **too compute-intensive** for mobile settings
- Idea: the model's own embeddings capture semantic similarity (e.g., shape, texture, layout)
  - a principled, low-cost way to detect redundant (proximity in Euclidean space) datapoints
- Challenge**:** Recomputing embeddings using the base model adds compute overhead
- Insight: Early layers of specialized and base models remain stable
  - A structural property of metalearned models.
  - Stable, low-level features such as object edges, contours, and textures persist through adaptation.

Remembrall: Selecting training datapoints 1/2

- Prior methods eliminate redundant datapoints using visual heuristics (e.g., frame differencing, optical flow)
  - **brittle** or **too compute-intensive** for mobile settings
- Idea: the model's own embeddings capture semantic similarity (e.g., shape, texture, layout)
  - a principled, low-cost way to detect redundant (proximity in Euclidean space) datapoints
- Challenge**:** Recomputing embeddings using the base model adds compute overhead
- Insight: Early layers of specialized and base models remain stable
  - A structural property of metalearned models.
  - Stable, low-level features such as object edges, contours, and textures persist through adaptation.
- Cache and reuse embeddings computed during inference by the specialized model.

Remembrall: Selecting training datapoints 2/2

Deduplication via Embedding-Based Sampling

- Cache the embeddings from inference and apply a sampling algorithm (spatially correlated poisson sampling).
- Inclusion probability of a datapoint is inversely proportional to its distance to other datapoints
- Lightweight, runs on CPU without blocking GPU inference.

Remembrall: Selecting training datapoints 2/2

Deduplication via Embedding-Based Sampling

- Cache the embeddings from inference and apply a sampling algorithm (spatially correlated poisson sampling).
- Inclusion probability of a datapoint is inversely proportional to its distance to other datapoints
- Lightweight, runs on CPU without blocking GPU inference.

Prioritizing Informative Datapoints

- Base model already encodes general structure shared across scenes. We need datapoints that focus on the scene-specific delta.
- Use model uncertainty (entropy of base model's output) to rank and train on the most informative datapoints only (active learning).
- Minimal overhead, since applied over a small, filtered set; briefly preempt inference on GPU.

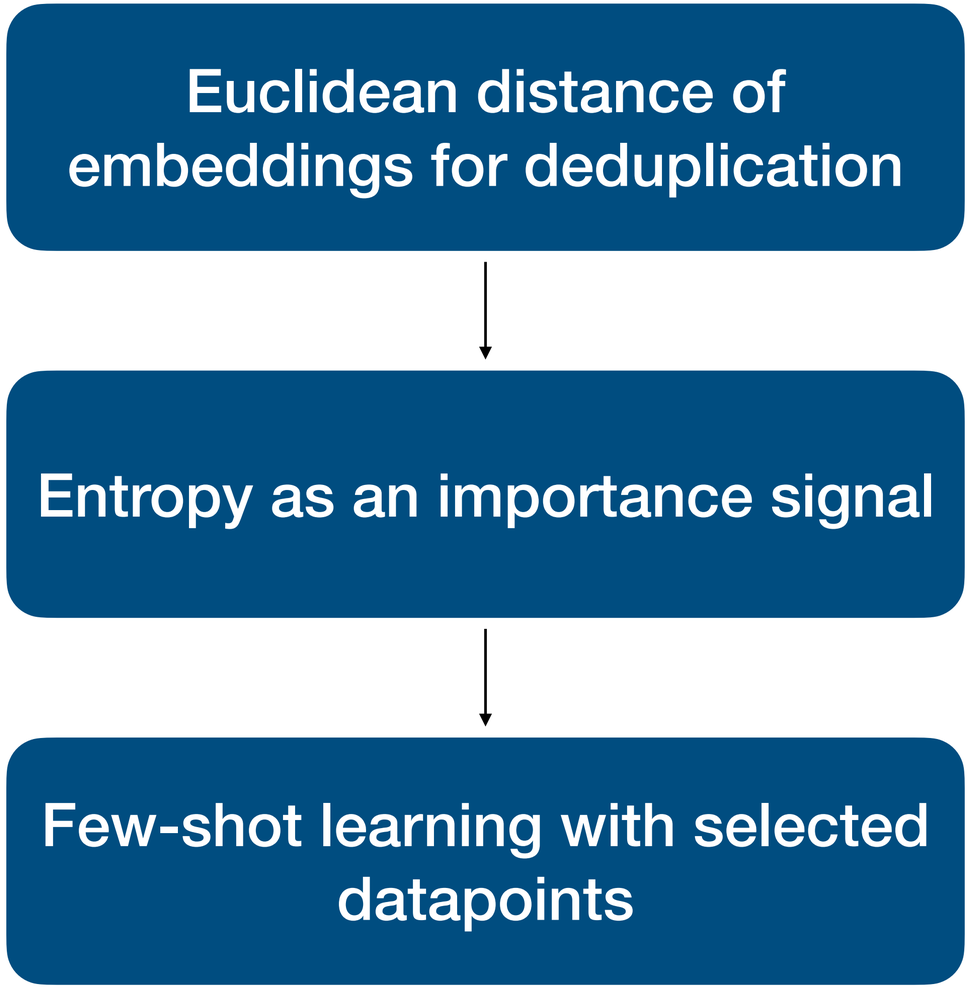
Remembrall: Selecting training datapoints 2/2

Deduplication via Embedding-Based Sampling

- Cache the embeddings from inference and apply a sampling algorithm (spatially correlated poisson sampling).
- Inclusion probability of a datapoint is inversely proportional to its distance to other datapoints
- Lightweight, runs on CPU without blocking GPU inference.

Prioritizing Informative Datapoints

- Base model already encodes general structure shared across scenes. We need datapoints that focus on the scene-specific delta.
- Use model uncertainty (entropy of base model's output) to rank and train on the most informative datapoints only (active learning).
- Minimal overhead, since applied over a small, filtered set; briefly preempt inference on GPU.



Challenge 2: Scheduling Retraining

Serialized GPU execution on SoCs forces a pause in inference during retraining

Challenge 2: Scheduling Retraining

Serialized GPU execution on SoCs forces a pause in inference during retraining

Longer training improves model quality, but

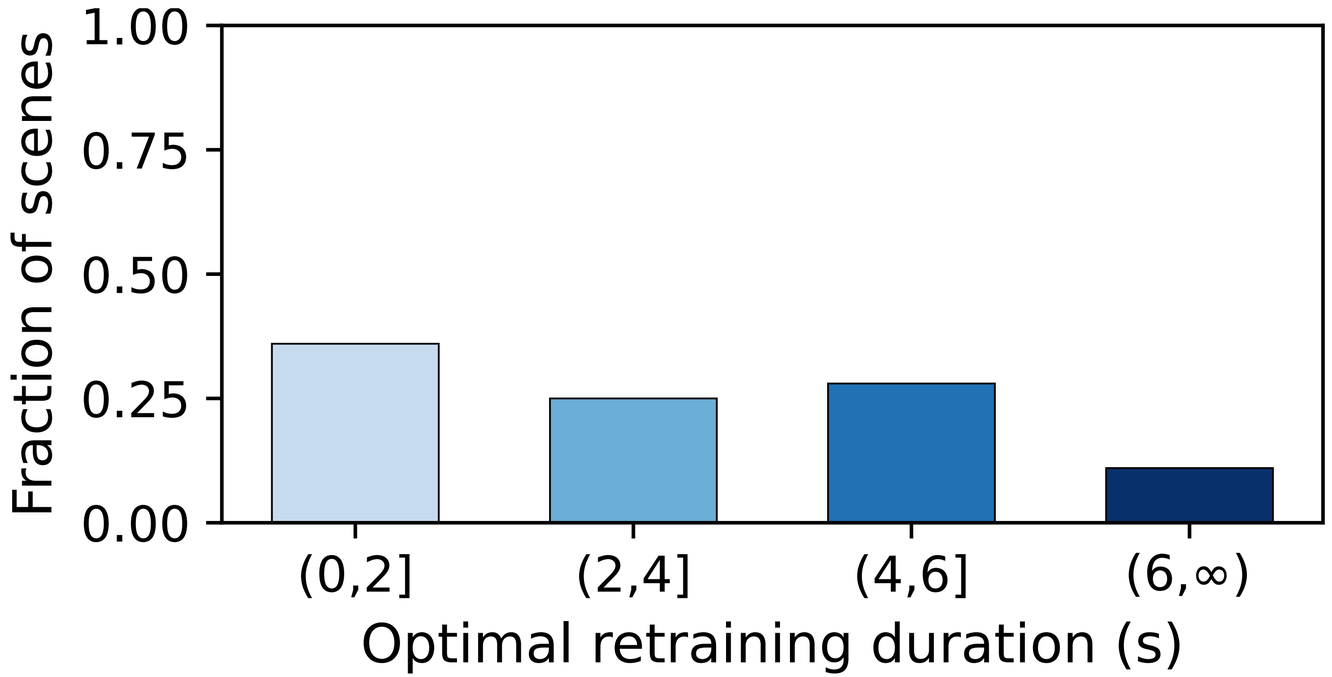
- reduces real-time frame processing
- increases risk of adapting to stale data

Challenge 2: Scheduling Retraining

Serialized GPU execution on SoCs forces a pause in inference during retraining

Longer training improves model quality, but

- reduces real-time frame processing
- increases risk of adapting to stale data



Challenge 2: Scheduling Retraining

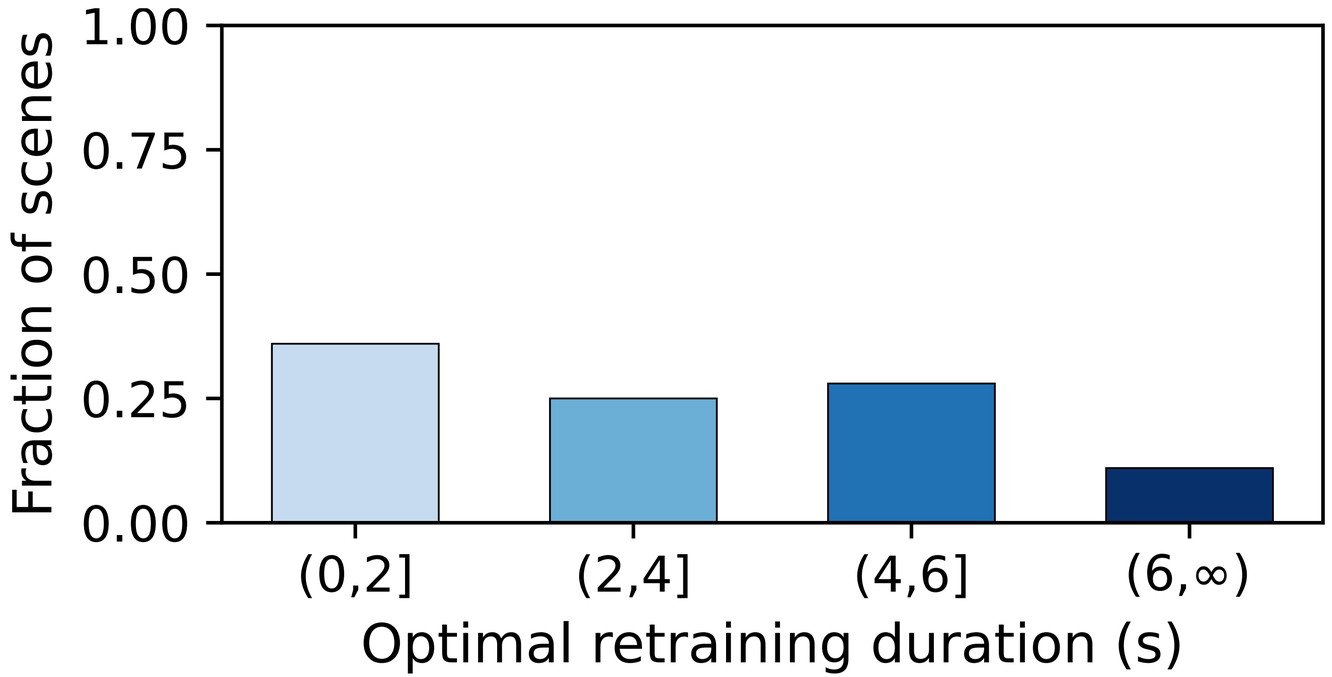
Serialized GPU execution on SoCs forces a pause in inference during retraining

Longer training improves model quality, but

- reduces real-time frame processing
- increases risk of adapting to stale data

The optimal retraining duration is scene-dependent.

- Need an adaptive stopping policy.

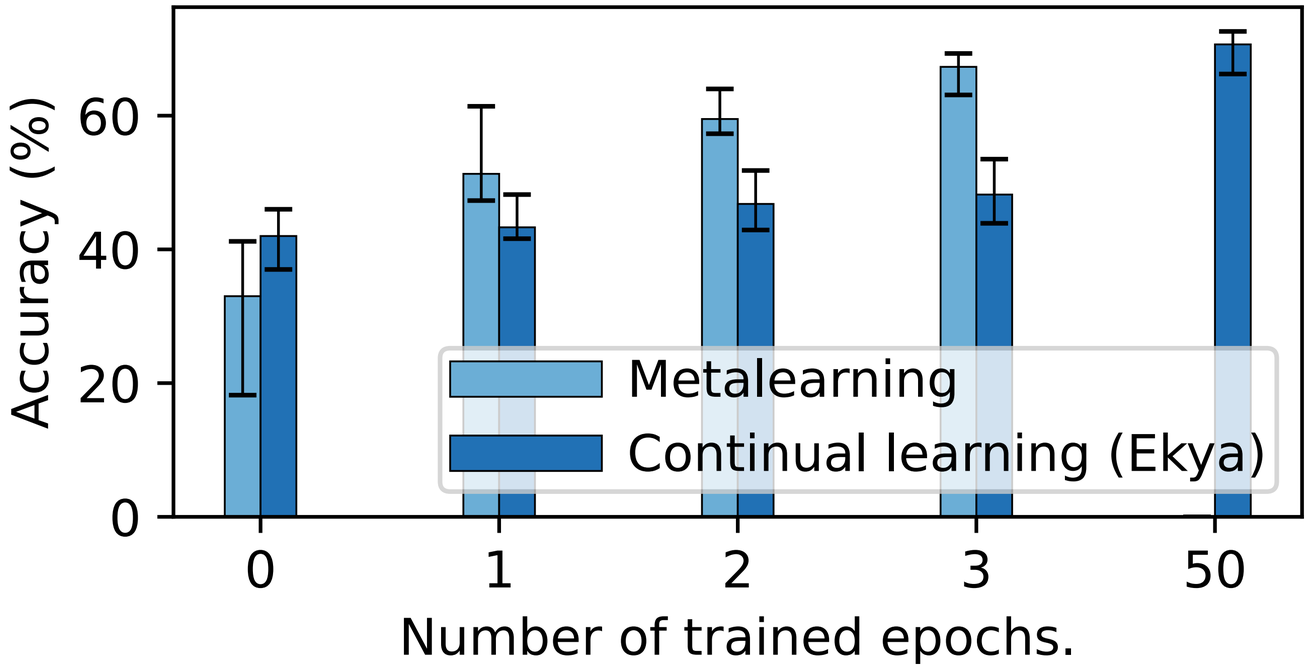


Remembrall: Scheduling retraining

- Metalearning exhibits rapid accuracy rampup.

Remembrall: Scheduling retraining

- Metalearning exhibits rapid accuracy rampup.

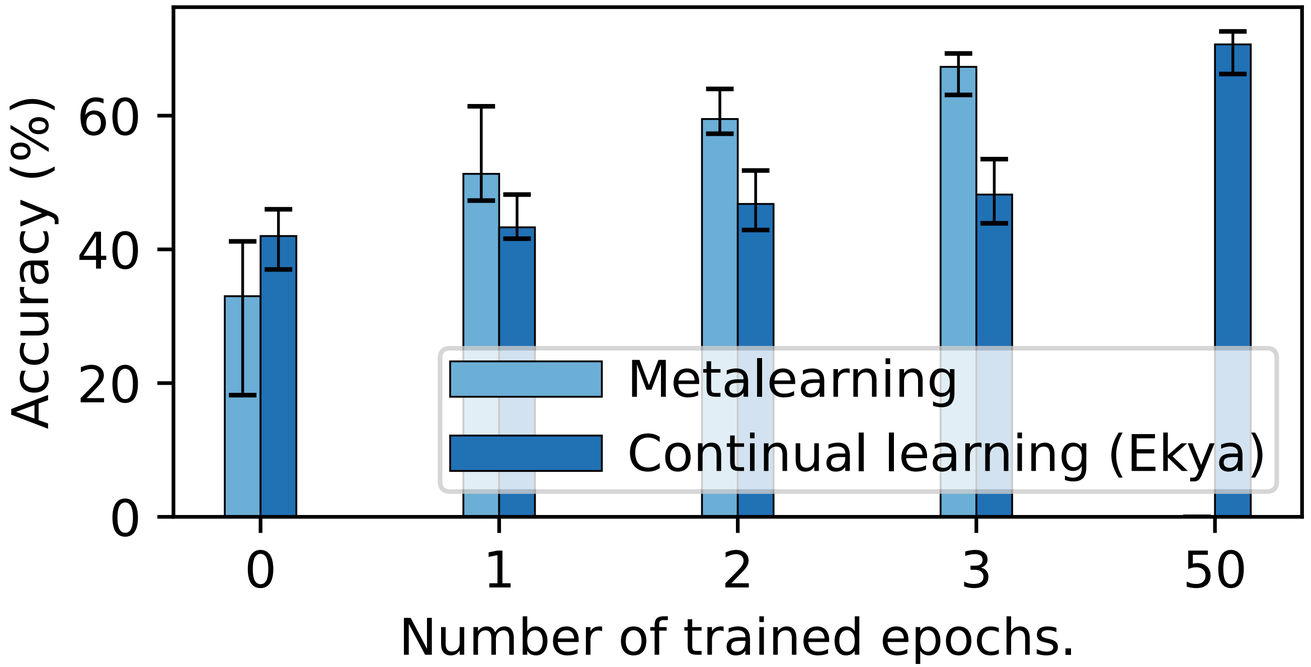


ResNet18 classification. Medians, errors span 25-75th percentiles.

Metalearning accuracy comparable to continual learning by epoch 5. Its bar at 50 is omitted to avoid overstating runtime.

Remembrall: Scheduling retraining

- Metalearning exhibits rapid accuracy rampup.
- Remembrall leverages this with *opportunistic early stopping* of retraining.

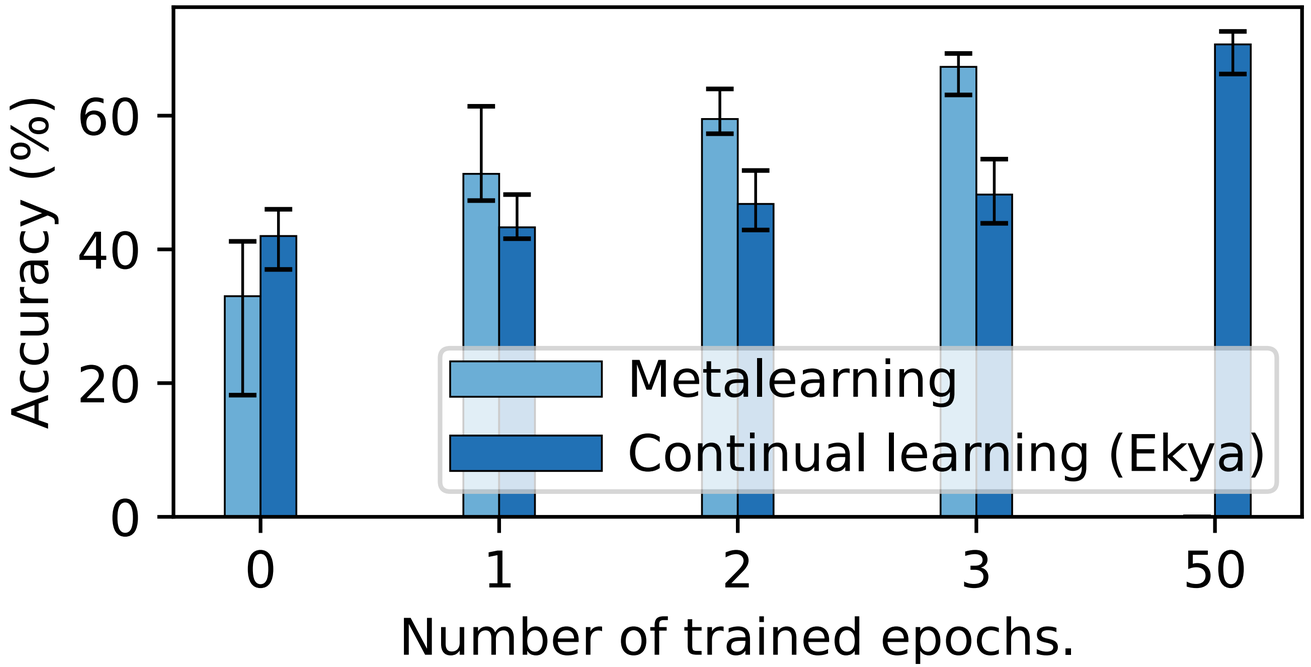


ResNet18 classification. Medians, errors span 25-75th percentiles.

Metalearning accuracy comparable to continual learning by epoch 5. Its bar at 50 is omitted to avoid overstating runtime.

Remembrall: Scheduling retraining

- Metalearning exhibits rapid accuracy rampup.
- Remembrall leverages this with *opportunistic early stopping* of retraining.
  - GPU: Retraining the new model
    - Accuracy improvement from prior epoch
  - CPU: Partial inference on realtime video
    - Quantify drift



ResNet18 classification. Medians, errors span 25-75th percentiles.

Metalearning accuracy comparable to continual learning by epoch 5. Its bar at 50 is omitted to avoid overstating runtime.

Remembrall: Scheduling retraining

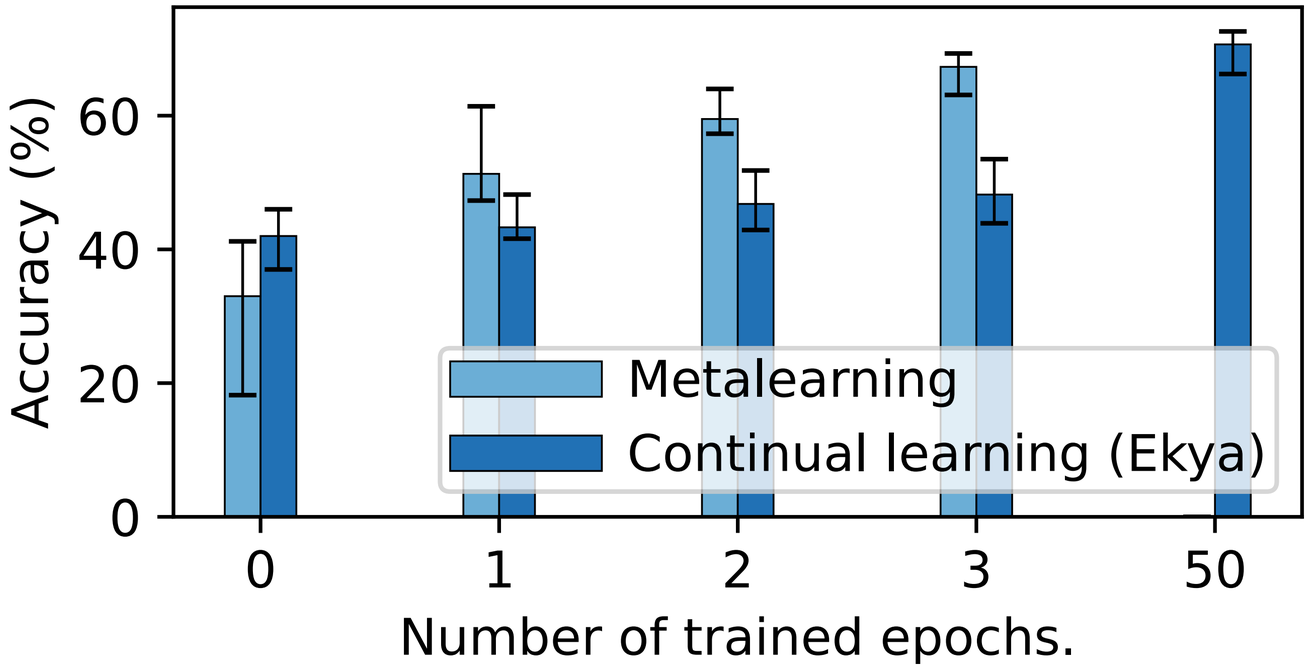
- Metalearning exhibits rapid accuracy rampup.
- Remembrall leverages this with *opportunistic early stopping* of retraining.
  - GPU: Retraining the new model
    - Accuracy improvement from prior epoch
  - CPU: Partial inference on realtime video
    - Quantify drift

$$S_t = w_1 \cdot \frac{\Delta A_t}{\Delta A_{\max}} - w_2 \cdot \frac{D_t}{D_{\max}}$$

Evaluate after each epoch

Normalized accuracy gain

Normalized drift



ResNet18 classification. Medians, errors span 25-75th percentiles.

Metalearning accuracy comparable to continual learning by epoch 5. Its bar at 50 is omitted to avoid overstating runtime.

Remembrall: Updating the base model efficiently

- After retraining, switch GPU to Inference.

Remembrall: Updating the base model efficiently

- After retraining, switch GPU to Inference.
- Lightweight update on CPU: interpolate base toward specialized model (inspired by “Reptile” metalearning).

Remembrall: Updating the base model efficiently

- After retraining, switch GPU to Inference.
- Lightweight update on CPU: interpolate base toward specialized model (inspired by “Reptile” metalearning).
- Drift-aware EWMA to avoid overfitting to transient scenes (e.g., tunnels):
  - Similar scene → aggressive update, Dissimilar scene → conservative

Remembrall: Updating the base model efficiently

- After retraining, switch GPU to Inference.
- Lightweight update on CPU: interpolate base toward specialized model (inspired by “Reptile” metalearning).
- Drift-aware EWMA to avoid overfitting to transient scenes (e.g., tunnels):
  - Similar scene → aggressive update, Dissimilar scene → conservative
- Scene similarity from cached embeddings; no extra compute

Remembrall: Updating the base model efficiently

- After retraining, switch GPU to Inference.
- Lightweight update on CPU: interpolate base toward specialized model (inspired by “Reptile” metalearning).
- Drift-aware EWMA to avoid overfitting to transient scenes (e.g., tunnels):
  - Similar scene → aggressive update, Dissimilar scene → conservative
- Scene similarity from cached embeddings; no extra compute

Why is such interpolation sufficient?

Remembrall: Updating the base model efficiently

- After retraining, switch GPU to Inference.
- Lightweight update on CPU: interpolate base toward specialized model (inspired by “Reptile” metalearning).
- Drift-aware EWMA to avoid overfitting to transient scenes (e.g., tunnels):
  - Similar scene → aggressive update, Dissimilar scene → conservative
- Scene similarity from cached embeddings; no extra compute

Why is such interpolation sufficient?

•Compact models (e.g., ResNet18) → Fully training base model offers little gain. (≤ 0.5% gap).

Remembrall: Updating the base model efficiently

- After retraining, switch GPU to Inference.
- Lightweight update on CPU: interpolate base toward specialized model (inspired by “Reptile” metalearning).
- Drift-aware EWMA to avoid overfitting to transient scenes (e.g., tunnels):
  - Similar scene → aggressive update, Dissimilar scene → conservative
- Scene similarity from cached embeddings; no extra compute

Why is such interpolation sufficient?

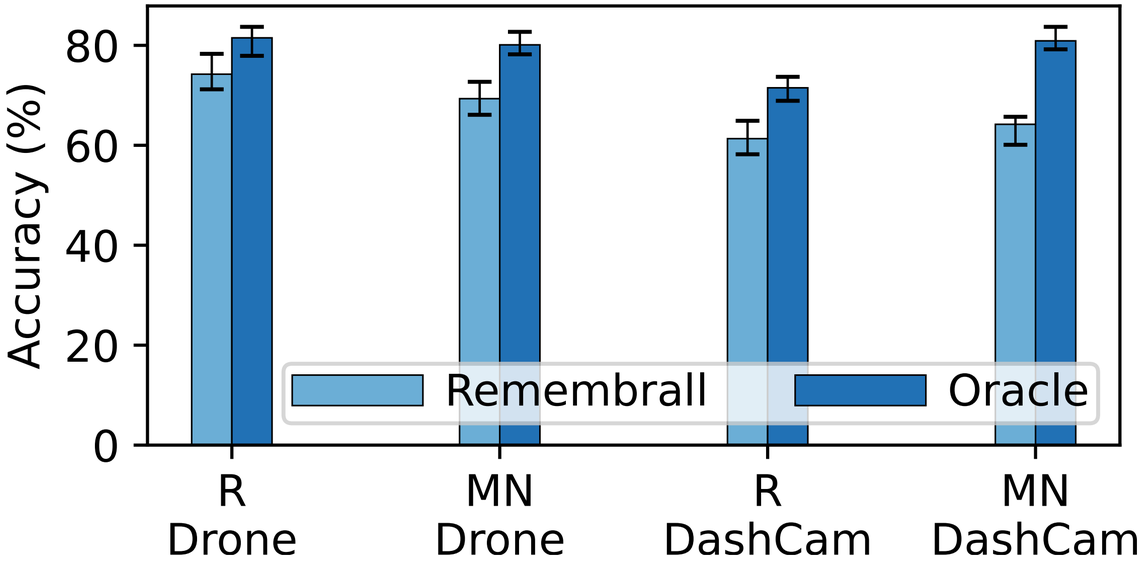
•Compact models (e.g., ResNet18) → Fully training base model offers little gain. (≤ 0.5% gap).

•Scene recurrence → benefits from gradual accumulation.

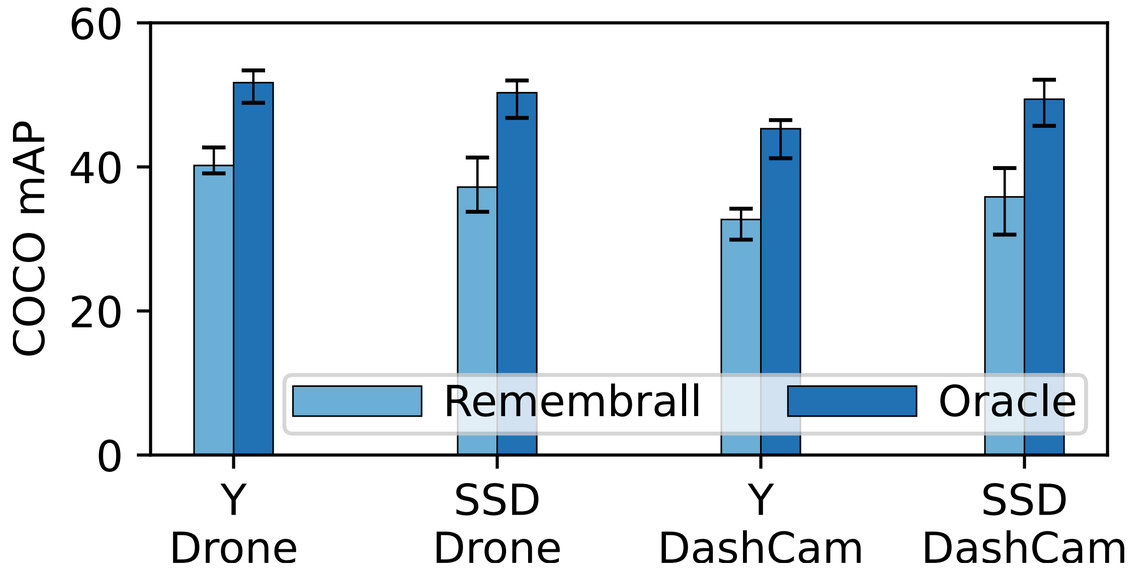
Evaluation

- How close does Remembrall's accuracy get to **oracle optimal**?
- How does Remembrall fare **against prior approaches**?
- How much does Remembrall reduce **overheads**?
- How do Remembrall's **contributions individually impact** the accuracy?
- How fast does Remembrall's base model **converge**?
- Can the converged **base model be reused** across videos?
- Robustness to transient vs. sustained drift.

Performance vs. Oracle

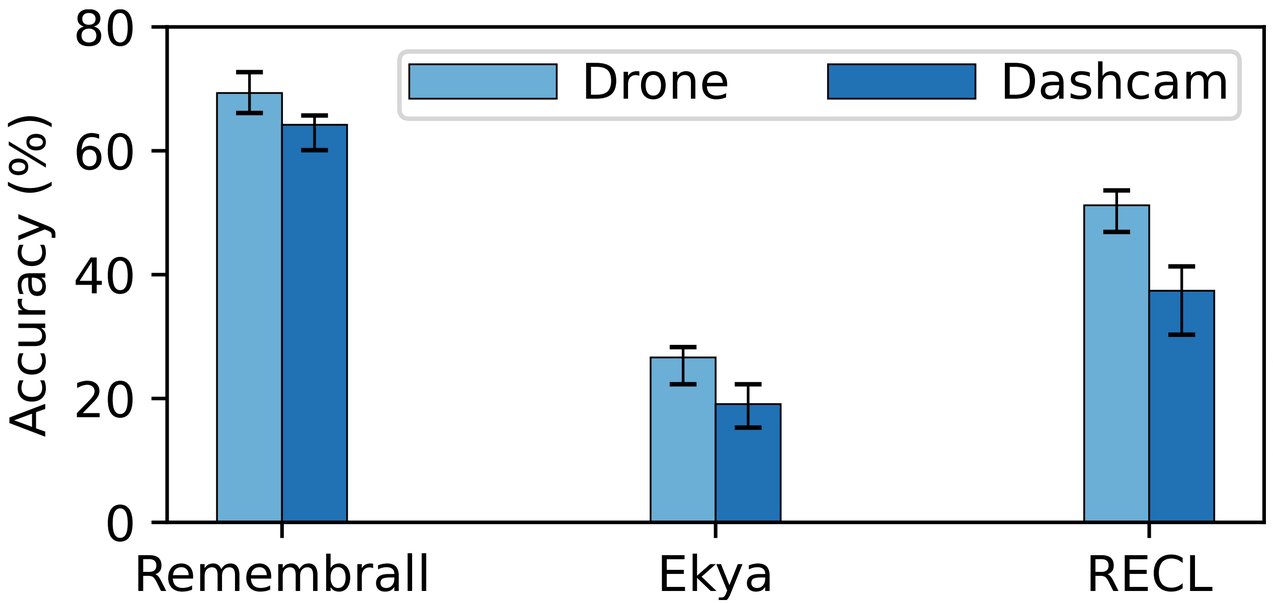


Classification. R=ResNet, MN=MobileNet.

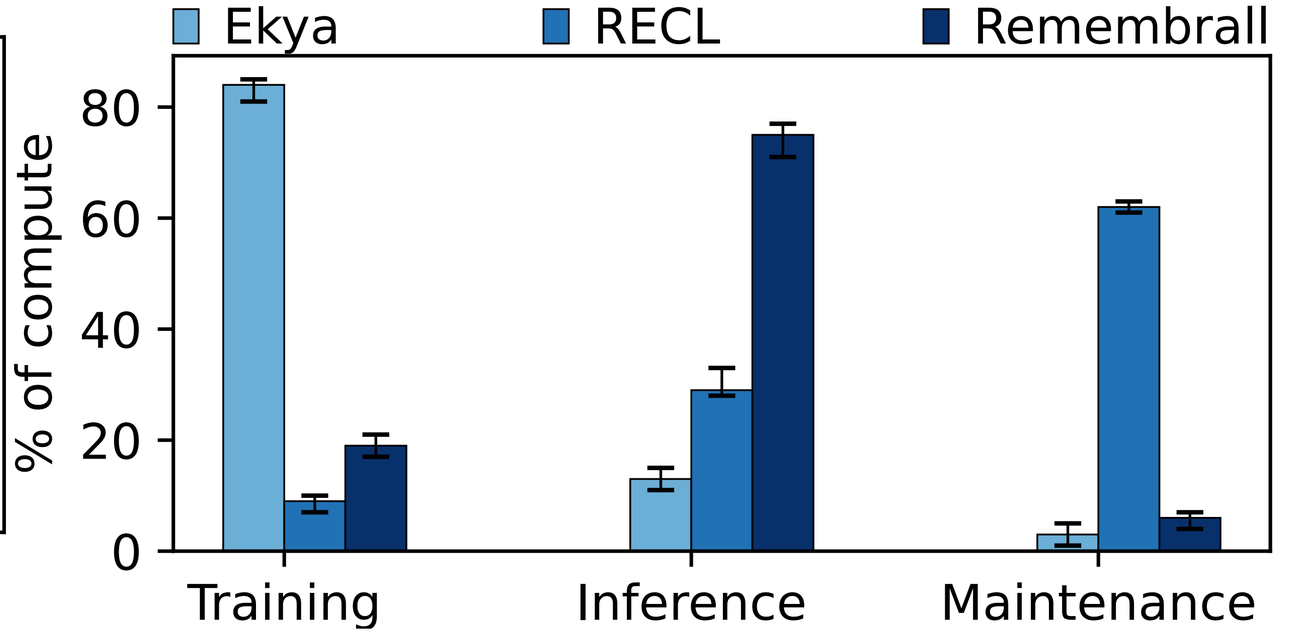


Detection. Y=YOLO, SSD=SSD-MobileNet.

Performance of Remembrall



Bars denote median overall classification accuracy (MobileNetV2) per video



Corresponding compute use on the Drone.

Error bars show 25th-75th percentile.

Summary

muralisr@princeton.edu

- ML-based video analytics is moving closer to the edge
- Prior approaches rely on compute-intensive retraining or large-scale model reuse
  - These approaches are fundamentally ill-suited to the SoC-based edge
- Remembrall instead **efficiently leverages memory to alleviate compute**
  - By incorporating cross-scene stability into a single extra base model
  - Embedding-guided sampling, inference-aware retraining, and lightweight meta-updates