# Appendix A. Videos

## A.1. Outdoor Scenes Dataset

The Outdoor Scenes video dataset includes seven publicly available videos from Youtube, with 7–15 minutes in duration. These videos span different levels of scene variability and were captured with four types of cameras: Stationary, Phone, Headcam, and DashCam. For each video, we manually select 5–7 classes that are detected frequently by our best semantic segmentation model (DeeplabV3 with Xception65 backbone trained on Cityscapes data) at full resolution. Table 4 shows summary information about these videos. In Figure 7 we show six sample frames for each video. For viewing the raw and labeled videos, please refer to https://github.com/modelstreaming/ams.

## A.2. Prior Work Videos

In our experiments, we also evaluate AMS on three long video datasets from prior work: Cityscapes [13] driving sequence in Frankfurt (1 video, 46 mins long)[5], LVS [46] (28 videos, 8 hours in total), A2D2 [23] (3 videos, 36 mins in total). Table 4 shows the summary information of the classes present in each video in these datasets. Overall, LVS includes fewer classes per video, and A2D2 and Cityscapes only include driving scenes. Hence, we introduced the Outdoor Scenes dataset that includes more diverse scenes and more classes.

# Appendix B. Other Related Work

**Continual/Lifelong Learning.** The goal of lifelong learning [41] is to accumulate knowledge over time [53]. Hence the main challenge is to improve the model based on new data over time, while not forgetting data observed in the past [37, 42]. However, as the lightweight models have limited capacity, in AMS we aim to track the best model at each point in time, and these models are allowed not to have the same performance on the old data.

**Meta Learning.** Meta learning [19, 51, 20] algorithms aim to learn models that can be adapted to any target task in a set of tasks, given only few samples (shots) from that task. Meta learning is not a natural framework for continual model specialization for video. First, as videos have temporal coherence, there is little benefit in handling an arbitrary order of task arrival. Indeed, it is more natural to adapt the latest model over time instead of always training from a meta-learned initial model.[6] Second, training such a meta model usually requires two nested optimization steps [19], which would significantly increase the server's computation overhead. Finally, most meta learning work considers a finite set of tasks but this notion is not well-suited to video.

**Federated Learning.** Another body of work on improving edge models over time is federated learning [43], in which the training mainly happens on the edge devices and device updates are aggregated at a server. The server then broadcasts the aggregated model back to all devices. Such updates happen at a time scale of hours to days [4], and they aim to learn better generalizable models that incorporate data from all edge devices. In contrast, the training in AMS takes place at the server at a time scale of a couple of seconds and intends to improve the accuracy of an individual edge device's model for its particular video.

**Unsupervised Adaptation.** Domain adaptation methods [2, 34] intend to compensate for shifts between training and test data distributions. In a typical approach, an unsupervised algorithm fine-tunes the model over the entire test data at once. However, frame distribution can drift over time. As our results show, one-time fine-tuning can have a much lower accuracy than continuous adaptation. However, it is too expensive to provide a fast adaptation (at a timescale of 10–100 seconds) of these models at the edge, even using unsupervised methods. Moreover, using AMS we benefit from the superior performance of supervised training by running state-of-the-art models as the "teacher" for knowledge distillation [30] in the cloud.

# Appendix C. Impact of model capacity and training horizon on online adaptation

AMS improves the performance of a lightweight model on edge devices through continual online adaptation. There exists a tradeoff in this approach between boosting the model's accuracy for the current data distribution and overfitting, which degrades performance when the data distribution changes. This tradeoff depends on the nature of the video (how fast the scenes change) and the model capacity. For instance, a low-capacity model may perform better with frequent updates despite overfitting.

Prior work on online model adaptation for video largely targets the overfitting regime: Just-In-Time [46] trains the lightweight model when it detects accuracy has dropped below a threshold, and it focuses its training on boosting the model's performance on the most recent frames. Just-In-Time must therefore repeatedly retrain its model as scenes change to maintain the desired accuracy. While this approach is sensible when training and inference both occur on the same powerful machine, it is impractical for online model training at a remote server. As our evaluation results show (§4.2), Just-In-Time's approach requires very frequent model updates, incurring a high communication overhead.

[5] This video sequence is not labeled and was the only long video sequence available from Cityscapes (upon request).

[6] An exception is a video that changes substantially in a short period of time, for example, a camera that moves between indoor and outdoor environments. In such cases, a meta model may enable faster adaptation.

| Dataset | Description | Classes |
|---|---|---|
| **Outdoor Scenes** | Interview | Building, Vegetation, Terrain, Sky, Person, Car |
| | Dance Recording | Sidewalk, Building, Vegetation, Sky, Person |
| | Street Comedian | Road, Sidewalk, Building, Vegetation, Sky, Person |
| | Walking in Paris | Road, Building, Vegetation, Sky, Person, Car |
| | Walking in NY | Road, Building, Vegetation, Sky, Person, Car |
| | Driving in LA | Road, Sidewalk, Building, Vegetation, Sky, Person, Car |
| | Running | Road, Vegetation, Terrain, Sky, Person |
| **A2D2** [23] | Driving in Gaimersheim | Road, Sidewalk, Building, Sky, Person, Car |
| | Driving in Munich | Road, Sidewalk, Building, Sky, Person, Car |
| | Driving in Ingolstadt | Road, Sidewalk, Building, Sky, Person, Car |
| **Cityscapes [13]** | Driving in Frankfurt | Road, Sidewalk, Building, Sky, Person, Car |
| **LVS** [46] | Badminton | Person |
| | Squash | Person |
| | Table Tennis | Person |
| | Softball | Person |
| | Hockey | Person |
| | Soccer | Person |
| | Tennis | Person |
| | Volleyball | Person |
| | Ice Hockey | Person |
| | Kabaddi | Person |
| | Figure Skating | Person |
| | Drone | Person |
| | Birds | Bird |
| | Dogs | Car, Dog, Person |
| | Horses | Horse, Person |
| | Ego Ice Hockey | Person |
| | Ego Basketball | Car, Person |
| | Ego Dodgeball | Person |
| | Ego Soccer | Person |
| | Biking | Bicycle, Person |
| | Streetcam1 | Car, Person |
| | Streetcam2 | Car, Person |
| | Jackson Hole | Car, Person |
| | Murphys | Bicycle, Car, Person |
| | Samui Street | Bicycle, Car, Person |
| | Toomer | Car, Person |
| | Driving | Bicycle, Car, Person |
| | Walking | Bicycle, Car, Person |

**Table 4:** Summary of the video datasets and their target classes in evaluations.


**Figure 7:** Sample video frames. Rows (from top to bottom) correspond to Interview, Dance Recording, Street Comedian, Walking in Paris, Walking in NY, Driving in LA, and Running.

We seek to avoid the need for frequent model updates by training the model over a suitable time horizon — not too short (which can lead to perpetual overfitting), but also not too long (which can hurt accuracy). The key observation is that although practical lightweight models (e.g., those customized for mobile devices) lack the generalization capacity of state-of-the-art models, they still have adequate capacity to generalize over a narrower distribution of frames (e.g., video captured in the same street, a specific room in a house, etc.).

To illustrate these issues, consider the model adaptation framework described in §3 with two knobs: (i) $T_{update}$, the *model update interval*; each $T_{update}$ seconds the model is trained and updated. (ii) $T_{horizon}$, the *training horizon*; each update uses (sampled) frames from the last $T_{horizon}$ seconds of video to train the model. For effective model adaptation, these two knobs are inter-dependent. When $T_{horizon}$ is small, the model updates tend to overfit and therefore must be frequent (small $T_{update}$), while a larger $T_{horizon}$ can produce models that generalize better and maintain high accuracy over a larger $T_{update}$ interval. Picking too large of a $T_{horizon}$, however, is also not ideal, as the model might not have sufficient capacity to generalize over a wide distribution of frames, reducing its accuracy.

Figure 8 illustrates this intuition for the video semantic segmentation task. We consider two variations of the lightweight model: *(i)* DeeplabV3 with MobileNetV2 backbone, *(ii)* a smaller version with the same architecture but with half the number of channels in each convolutional layer. We pick 50 points in time uniformly distributed over the course of a video of driving scenes in Los Angeles. For each time $t$, we train the two lightweight models on the frames in the interval $[t - T_{horizon}, t)$ and then evaluate them on frames in $[t, t + T_{update})$ (with $T_{update} = 16$ sec).

We plot the average accuracy of the two variants for each value of $T_{horizon}$ in Figure 8a. As expected, the smaller model's accuracy peaks at some training horizon ($T_{horizon} \approx 256$ sec) and degrades with larger $T_{horizon}$ as the model capacity becomes insufficient. The default model exhibits a similar behavior, but with a more gradual drop for large $T_{horizon}$. Figure 8b shows the impact of training horizon on the model update frequency required for high accuracy. For the same driving video, we plot accuracy vs. the model update interval ($T_{update}$) for the default model training using $T_{horizon} = 16, 64, 256$ sec. As expected, more frequent model updates improves accuracy in all cases, but the accuracy of models trained with a small training horizon ($T_{horizon} = 16$ sec) drops much more sharply as we increase $T_{update}$.

The best values of $T_{horizon}$ and $T_{update}$ depend on both

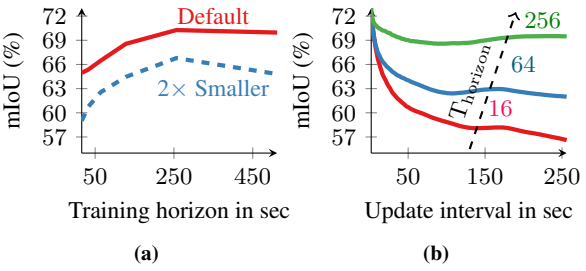


**Figure 8:** Impact of training horizon and model update interval on the mean-intersection-over-union (mIoU) accuracy for semantic segmentation.

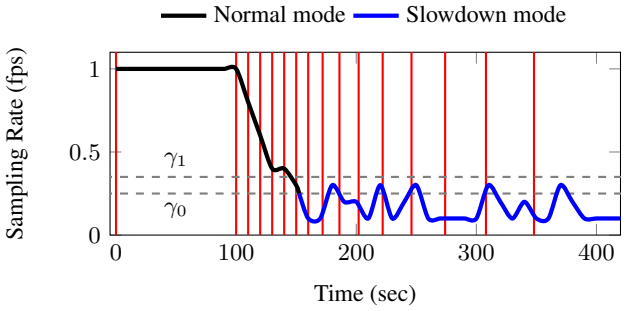


**Figure 9:** ATR updates of the model update intervals w.r.t. the average sampling rate over time for Vid1. Vertical lines represent model updates. Model updates become distant after entering the slowdown mode.

the video and the model capacity. Overall we have found that for semantic segmentation using mobile-friendly models, a training horizon of 3–5 minutes works well with model updates every 10–40 seconds across a variety of videos (§4).

## Appendix D. Adaptive Training Rate (ATR)

We dynamically update the model update interval $T_{update}$ for each device based on its video characteristics. For this purpose, for each interval $n$, we look at ASR's sampling rate decision $r_n$ (see §3.2). A small sampling rate typically implies the scenes are changing slowly, and conversely a large sampling rate suggests fast variations.

We introduce a *slowdown* mode to capture relatively stationary scenes. We enter the slowdown mode if the scenes are highly similar, $r_n < \gamma_0$, and exit this mode as variations increase, $r_n > \gamma_1$. Our implementation uses $\gamma_0 = 0.25$ fps and $\gamma_1 = 0.35$ fps. We start at the maximum training rate ($T_{update} = \tau_{min}$), and update the training interval every $\delta t$ seconds according to:

$$T_{update}(n+1) = \begin{cases} T_{update}(n) + \Delta, & \text{in slowdown mode} \\ \tau_{min}, & \text{otherwise} \end{cases} \quad (2)$$

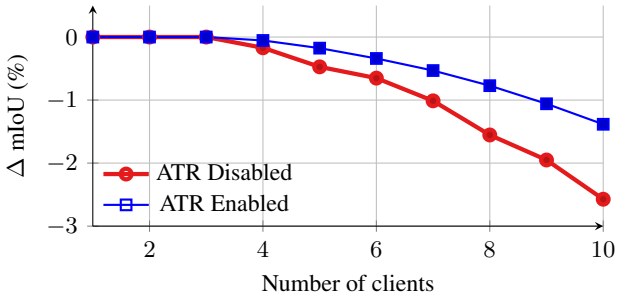


**Figure 10:** Average multiclient mIoU degradation compared to single-client performance.

This rule gradually increases $T_{update}$ by a fixed $\Delta$ (e.g., $\Delta = 2$ sec) in slowdown mode, and aggressively resets it to $\tau_{min}$ as soon as we exit the slowdown mode to catch up with scene changes.

The sever communicates the newest $T_{update}$ (and sampling rate) with the edge so that the edge device can accordingly synchronize its sample buffering and upload process (see §3.2).

In Figure 9, we plot the behavior of ATR algorithm for the Interview video with relatively stationary scenes from Outdoor Scenes dataset. We observe that ATR enters the slowdown mode after 150 seconds as the average sampling rate goes below the entrance threshold $\gamma_0$, and it stays in this mode as the scenes rarely change after this point. We denote the model updates using the vertical lines in this plot. ATR increases the distance between the model updates in the slowdown mode to save the training cycles for other videos.

## Appendix E. Server Utilization

Every user that joins a cloud server requires its own share of GPU resources for inference and training operations. GPUs are expensive. At the current time, renting a GPU like the NVIDIA Tesla V100 in the cloud costs at least $1 per hour. Hence, it is important to use server GPU resources efficiently and serve multiple edge devices per GPU to keep per-user cost low.

In our prototype, we use a simple strategy that iterates in a round-robin fashion across multiple video sessions, completing one inference and training step per session. By allowing only one process to access the GPU at a time, we minimize context switching overhead. In Figure 10 we show the decrease (w.r.t. single client) in average mIoU when different number of clients share a GPU. We observe that even with a simple round-robin scheduling algorithm, AMS scales to up to 7 edge devices on a single V100 GPU with less than 1% loss in mIoU without adaptive training rate (ATR) enabled. Enabling ATR (see Appendix D) increases the number of supported edge devices to 9. Note that these results depend on the distribution of the videos and for this purpose, we have

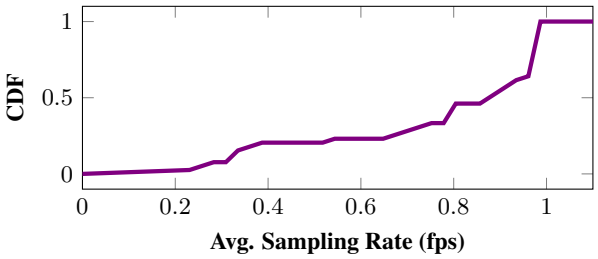


**Figure 11:** Cumulative distribution of average ASR sampling rate across all videos.

assumed a uniform sampling of videos from the Outdoor Scenes dataset and reported the average result of multiple runs here. As most of the videos in this dataset (5 out of 7) tend to experience relatively high levels of scene dynamic, majority of videos get high training frequency. Hence, we expect to be able to support at least equal or even more devices by prioritizing certain videos that need more frequent model updates over the stationary ones in real-world distribution of videos.

Furthermore, we note that ASR (see §3.2) also significantly reduces the overhead of running teacher inference over redundant frames at the server. The impact is particularly pronounced because the teacher model usually runs at high input resolution and consumes a significant amount of GPU time (up to 200 ms for labeling each frame on an NVIDIA V100 GPU for the task of semantic segmentation).

## Appendix F. Uplink Sampling Rate

Figure 11 shows the distribution of ASR's average sampling rate across different videos in four datasets. Notice that we set the ASR's maximum sampling rate (see §3.2), $r_{max}$, to 1 fps as our results show sampling faster than 1 frame-per-second provides negligible improvement in accuracy along increasing bandwidth usage and server inference overhead. We use $r_{min} = 0.1$ fps.