Edge-Based Live Learning for Robot Survival

Eric Sturzinger, Jan Harkes, Padmanabhan Pillai, *Member, IEEE*, and Mahadev Satyanarayanan, *Life Fellow, IEEE*

***Abstract*—We introduce *survival-critical machine learning (SCML),* in which a robot encounters dynamically evolving threats that it recognizes via machine learning (ML), and then neutralizes. We model survivability in SCML, and show the value of the recently developed approach of *Live Learning.* This edge-based ML technique embodies an iterative human-in-the-loop workflow that concurrently enlarges the training set, trains the next model in a sequence of "best-so-far" models, and performs inferencing for both threat detection and pseudo-labeling. We present experimental results using datasets from the domains of drone surveillance, planetary exploration, and underwater sensing to quantify the effectiveness of Live Learning as a mechanism for SCML.**

***Index Terms*—Edge computing, machine learning, robotics.**

I. Introduction

As CONFIDENCE grows in Machine Learning (ML), it is starting to be used in roles that are critical to the survival of the system on which it is hosted [1]. This is analogous to an immune system, upon which rests the survival of a biological organism. Continuous learning to defend against dynamically evolving threats is the essence of an immune system. Although it will probably take time for ML-based threat response systems to attain the sophistication of biological immune systems that have evolved over a billion years, they both rely on continuous learning. We refer to ML in such systems as *survival-critical machine learning (SCML).*

SCML forces rethinking of current ML practices. Today, learning is rigidly separated into distinct phases: (a) data collection (including labeling and training set construction); (b) training; and (c) inferencing. We show that SCML forces us to blur the rigid boundaries between these phases. As threats evolve and morph continuously, there is no "pause" button to allow learning to catch up. All phases of learning have to be pipelined and executed concurrently. Falling behind in learning can be fatal. In classic ML settings, only the best model trained on a fully assembled training set is of interest. In SCML, "best so far" models are useful. For example, if a new type of threat appears, survivability is lowered until it can be reliably detected. While a large training set for the new threat is being collected (which may take many hours or days), it may help to deploy a weak model that is based only on the limited learning possible so far. In offensive settings, learning may be required because programmable camouflage [2], [3] has morphed the visual appearance of a target.

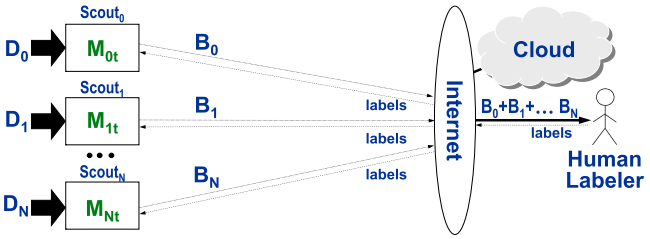


Figure 1. Conceptual view of live learning (source: [4]).

In this paper, we show how a new edge-based technique called *Live Learning* can be leveraged for SCML. This pipelined and iterative workflow for ML was introduced in 2023 by George et al. [4]. It was originally conceived as a mechanism for selective transmission of high-bandwidth real-time sensor data for human-in-the-loop labeling over extreme low-bandwidth wireless networks. Figure 1 illustrates the original problem setting of Live Learning. A team of autonomous unmanned probes called *scouts* are on a mission to collect new ML training data. Examples of scouts include aerial drones, satellites, interplanetary spacecraft, and underwater drones. From the viewpoint of ML, the critical requirement of a scout is that it has sufficient computing resources and storage to perform training as well as inferencing. Today, this can be provided by embedded hardware such as the NVIDIA Jetson AGX Orin [5]. In contrast to Federated Learning [6], edge data in Live Learning is unlabeled. Labels can be obtained only from a distant human during a mission. Intelligent and self-improving transmission of a small subset of incoming sensor data for labeling was the original motivation for Live Learning.

In spite of its different heritage, and independent of bandwidth considerations, we show in this paper that Live Learning is directly applicable to SCML. This paper makes four contributions:

- It identifies the unique ML challenges of survival-critical settings and highlights the shortcomings of current ML cost metrics in these settings.

Received 31 January 2024; revised 23 July 2024; accepted 26 September 2024. Date of publication 17 October 2024; date of current version 10 March 2025. This work was supported in part by U.S. Army Research Office and the U.S. Army Futures Command under Contract W519TC-23-C-0003, in part by United States Navy under Award N00174-23-1-0001, and in part by National Science Foundation under Grant CNS-2106862. *(Corresponding author: Mahadev Satyanarayanan.)*

Eric Sturzinger, Jan Harkes, and Mahadev Satyanarayanan are with Carnegie Mellon University, Pittsburgh, PA 15213 USA (e-mail: esturzin@andrew.cmu.edu; jaharkes@cs.cmu.edu; satya@cs.cmu.edu).

Padmanabhan Pillai is with Intel Labs, Hillsboro, OR 97124 USA (e-mail: padmanabhan.s.pillai@intel.com).

Digital Object Identifier 10.1109/TETC.2024.3479082

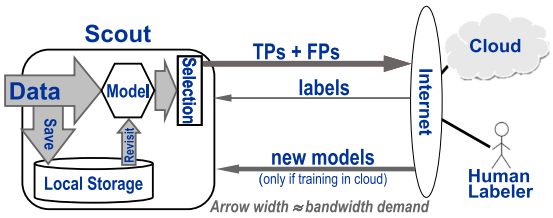


Figure 2. End-to-end pipeline of Hawk (source: [4]).

- It develops an analytical model for reasoning about survivability in SCML.
- It shows how Live Learning can be adapted for SCML.
- It presents experimental results from multiple state-of-the-art datasets to quantify the effectiveness of Live Learning as a mechanism for SCML.

II. Background and Related Work

A. Live Learning and Hawk

As originally conceived, Live Learning addresses the problem of severe bandwidth mismatch between incoming sensor data rate at scouts and wireless backhaul bandwidth. This mismatch is overcome via a self-improving ML-based transmission system embodied in an open source implementation called *Hawk* [7]. Our repurposing of Live Learning for SCML requires changes to Hawk. Specifically, low bandwidth is no longer a motivation — Live Learning is valuable at any backhaul network bandwidth. We describe the original implementation of Hawk below, and describe our modifications later in the paper (Section A).

Starting from a weak model that is trained via few-shot learning (FSL) on just a few initial examples, Hawk seamlessly pipelines semi-supervised learning, active learning, and transfer learning, with asynchronous bandwidth-sensitive data transmission to a distant human for labeling. When a significant number of true positives (TPs) have been labeled, Hawk trains an improved model to replace the old model. This iterative workflow is Live Learning, which continues until a sufficient number of TPs have been collected. The workflow is independent of the specific deep neural network (DNN) architecture that is used for learning.

Figure 2 illustrates the Hawk processing pipeline. Fresh data arrives continuously at a high rate into a scout. Only a tiny fraction of this incoming data can be transmitted for labeling. Incoming data is both written to local storage, and processed. The processing consists of three steps. The video stream is first decoded into individual frames. Each frame is broken up into small tiles, and the tile stream is inferenced using the current model. After inferencing, a small subset of the tiles is selected for transmission and labeling. A tunable data sourcing policy guides processing new incoming data versus re-processing previously discarded data. Transmitted items are queued in the cloud for labeling by a human expert. This data is also available for training in the cloud. Labels are transmitted back to the scout as they are generated. Hawk can choose between training in the cloud or on the scout. If training in the cloud is chosen, the new model is transmitted to the scout after training. With multiple scouts, precious TPs are shared across the team so that models improve at their cumulative TP discovery rate. For training on scouts, negatives are obtained locally. For cloud training, negatives are obtained from archives.

B. Pre-ML Work on Survivability

For centuries before ML emerged, the concept of survivability was developed in diverse contexts. The first known use of this concept was in the creation of a mortality or life table in the late 17th century [8]. Modern actuarial analysis for life insurance is a direct descendant of that early work [9], [10], [11], [12]. In the 20th century, medical epidemiology has proved to be a major driver of survivability analysis. Examples of work relating to survivability in this domain include [13], [14], [15], [16], [17], [18], [19], [20].

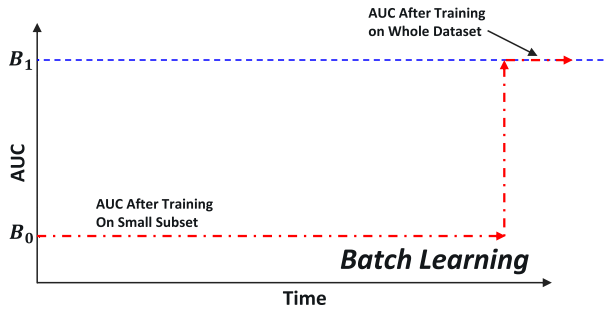
For technologically complex systems that must endure harsh and unforgiving environments, survivability analysis has been an integral part of the design process. Examples of such systems include spacecraft [21], [22], [23] and military systems [24], [25], [26]. More recently, survivability analysis has been used in digital and pervasive system design and maintenance [27], [28], [29]. Most recently [30], [31], novel ML methods have been used to improve survivability analysis in existing domains. In contrast, this work aims to use ML as a functional part of an operational system. Live Learning is thus integral to such a system, not a tool for analyzing its survivability.

III. An Analytical Model for SCML

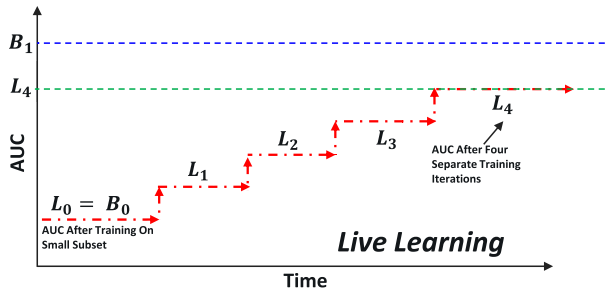
Since SCML is a new concept, we need to gain an understanding of its tradeoff space. Towards this goal, we develop an analytical model of SCML. Our intent is to capture the highlights at an intuitive level, rather than to rigorously model minutiae. In a later section (Section IV), we ground this model in the context of an implementation and obtain experimental results that reflect the constraints of a real system. For simplicity, we focus on the ML problem of classification. In practice, an SCML system also needs to localize the objects that it is classifying — i.e. the underlying problem is object detection in live data, not just classification. Since that further level of complexity adds no new insights to our discussion, we focus on classification. For every incoming data item, a scout asks: *"Is there a threat at this moment?"*

A. Precision & Recall Alone are Incomplete

Historically, time has played no role in characterizing the accuracy of a classifier. The well-known metrics of precision and recall are all that matter. The time needed to create and use a model (i.e., data collection, labeling, training, and inferencing) is not relevant. These are viewed as secondary metrics, typically outside scope in discussions of accuracy. In SCML, time can no longer be relegated to a secondary role. A single classification error can lead to an undetected threat destroying the scout. There will be no further opportunity for that model to redeem itself through future successes. This is different from classic ML systems in which a classification error is not fatal — often, it



(a) Batch Learning (classic ML)



(b) Live Learning (see Figure 14 for concrete examples)

These two graphs compare classifier AUC as a function of time between Batch Learning and Live Learning. In Batch Learning, the AUC is not improved for the current mission during data collection, labeling, training set construction, and training. The fruits of this work are available only to future missions. In contrast, Live Learning improves AUC during the current mission. Although it may be unable to achieve the same optimal model as Batch Learning (because of the constraints of online rather than offline learning), it improves survivability for the current mission. This improvement is correlated with the increase in area under the AUC curve (i.e., integral of AUC over mission duration).

Figure 3. AUC improvement over time in SCML.

merely lowers profit. In SCML, an early mistake means that the system dies young. Early survival, on the other hand, leads to later mistakes being less likely because of model improvement. Thus, time has to be viewed on an equal footing with precision and recall in SCML.

The role of time is captured by Figure 3, which contrasts Batch Learning (classic ML) and Live Learning. The Y axis is the area under the precision-recall curve (AUC). In Batch Learning (Figure 3(a)), training is deferred until ample data has been collected and labeled. Missions are not even attempted until an accurate model exists. Unfortunately, SCML has to cope with a world in which threats are not static, but evolve. Critical missions may have to be attempted even without a high-quality model for a new threat class. For that class, a weak model that is based on a small training set may be all that is available at the start of a mission. During the mission, Live Learning can be used on incoming data to improve the model. More threats will be encountered on longer missions. Hence, there is greater danger that a threat will get through undetected and destroy the scout. However, if the scout survives the early part of a long mission, Live Learning has significant opportunity to improve the model. There is thus a delicate balance between survival and learning that involves time at its heart, as shown by Figure 3(b).

The key advantage of Batch Learning is that all data is available up front for analysis to create an optimal training set. This can yield the best model achievable from that training data for a given model architecture. Unfortunately, this optimality is of little value in an SCML setting — if the mission is attempted, the scout is unlikely to survive long enough to benefit.

Like the proverbial tortoise relative to a hare, Live Learning settles for much smaller incremental gains. However, these gains are available during the current mission. The gains are suboptimal relative to Batch Learning because they have to be made in an online setting. As Figure 3(b) illustrates, the result is a sub-optimal final AUC ($L_4 < B_1$). However, survivability is improved (Section III-E).

B. The Mission Abstraction

Without loss of generality, we define a mission to be the successful delivery of some valuable payload by a scout from a source to a destination. The mission involves the risk of destruction of the scout by threats during its journey. For example, the payload may be life-saving medical supplies, food, or essential spare parts; the threats may be bullets, missiles, etc. The scout is protected by an SCML system that continuously tries to detect incoming threats. Once a threat is detected, it can be neutralized with certainty by some *countermeasure (CM)*. Exactly how this happens is not relevant to SCML, and is therefore outside the scope of this paper. In practice, of course, the neutralization may not be always successful, and there may only be a limited number of CMs available on the scout. We also assume that a single scout is assigned to the mission. In practice, a convoy or swarm of scouts may be assigned. We ignore these complications initially, in order to simplify the analytical model, and thereby gain an intuitive understanding of SCML tradeoffs. We include these aspects in the experimental evaluation (Sections IV-D and VI).

C. Characterizing Survivability

Survival is inherently probabilistic in nature. On an unlucky day, even a superb model may fail to detect just one threat out of many that it correctly detects. That unfortunate single mistake may result in scout destruction. Conversely, the scout may have a lucky day. A weak model may wrongly classify every threat, yet none of these undetected threats destroy the scout, and it survives unscathed. Both extremes are unlikely, yet either can happen on a real mission. A Bayesian prior is the density of threats during a mission. Regardless of model accuracy, higher threat density is inherently riskier. It may vary during a mission, being correlated with space (e.g., physical location) and time (e.g., greater chance of being discovered on a longer mission). For simplicity, our modeling assumes a constant threat density during a mission.

In epidemiology, the metric of *Life Expectancy* characterizes survivability. This is the mean of the probability distribution of the time interval from birth to death of a large population of organisms. We adopt this metric for SCML. Mission start corresponds to birth. Mission end may occur because of successful completion, or because of scout destruction. Too small a life expectancy leads to many mission failures.

α	threat arrival rate
β	recall of SCML system
γ	lethality of threats

Figure 4. Parameters relevant to survivability.

D. Key Parameters and Relationships

Figure 4 summarizes the key parameters of survivability α, β, and γ, which are explained below. We assume that the stochastic process generating threats is memoryless. The threat arrival rate is then exponentially distributed. In other words, the probability distribution of the interarrival time, t, of threats is given by

$$p(t) = \alpha e^{-\alpha t} \tag{1}$$

where $1/\alpha$ is the mean of the distribution. More complex interarrival time distributions can be considered, but this simple model is adequate for gaining intuition about SCML.

From an ML and Live Learning viewpoint, a key consideration is *class imbalance*. In other words, how rare are threats relative to all other data classes that are seen during a mission? Extremely rare threats are good news from the viewpoint of survivability. However, they also imply slow learning because few TPs can be collected for a training set. Frequent exposure to threats (and survival by good luck) leads to much faster learning and model improvement. This subtle relationship between risk and learning is expressed by:

$$\alpha = \frac{\text{total data samples}}{\text{mission duration}} \times \text{threat class base rate} \tag{2}$$

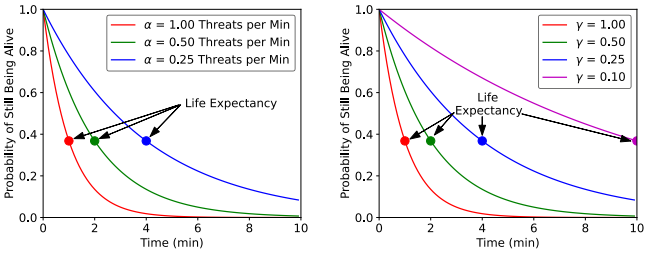
Small values of threat class base rate correspond to extreme class imbalance. For a given mission duration, this leads to low α even with large number of data samples.

Since an SCML system performs classification (Section III), its ability to detect and neutralize a potential threat is given by the classic ML measure of recall, symbolized as β

$$\beta = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{3}$$

where FN correponds to false negatives (i.e., positives that are wrongly classified as negatives). Every FN is an undetected threat that may destroy the scout. Values of β close to 1.0 imply an SCML system that is effective in detecting threats. If threat neutralization is imperfect, an additional term to account for this will be needed in the analysis. As discussed earlier (Section III-B), the classic ML metric of precision is important from the viewpoint of conserving CMs. Each false positive (FP) results in a CM being wasted. Achieving high recall at poor precision is suboptimal because it may result in a finite supply of CMs being exhausted long before mission completion. The scout is then vulnerable to all further threats encountered.

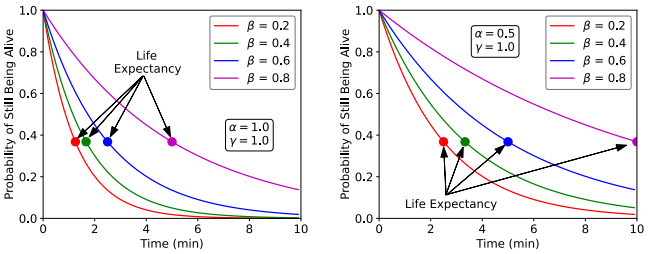
In practice, not every undetected threat is fatal. In the context of an immune system, for example, a pathogen that is undetected may still not successfully infect an organism or kill it. We use the term *lethality*, symbolized as γ, for the probability that an undetected threat leads to scout destruction. For "dumb" threats such as bullets, γ may only be a few percent. With precision guided munitions, it may be much higher and approach 100%.



(a) $\beta = 0$ and $\gamma = 1.0$ (b) $\alpha = 1.0$ and $\beta = 0$

For all curves above, the mean value is $1/e$, which is ~0.368.

Figure 5. Impact of threat arrivals & lethality (no SCML).



(a) $\alpha = 1.00$ and $\gamma = 1.0$ (b) $\alpha = 0.5$ and $\gamma = 1.0$

For all curves above, the mean value is $1/e$, which is ~0.368.

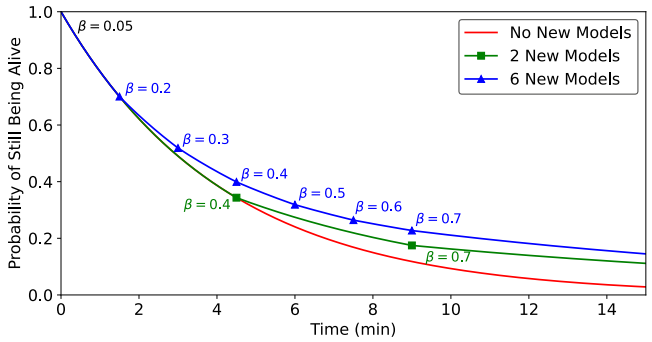
Figure 6. Using a fixed pre-trained model in SCML.

Based on the parameters in Figure 4, the cumulative distribution function (CDF) $S(t)$ that an SCML system will survive at least until time t into the mission is given by:

$$S(t) = e^{-\alpha(1-\beta)\gamma t} \tag{4}$$

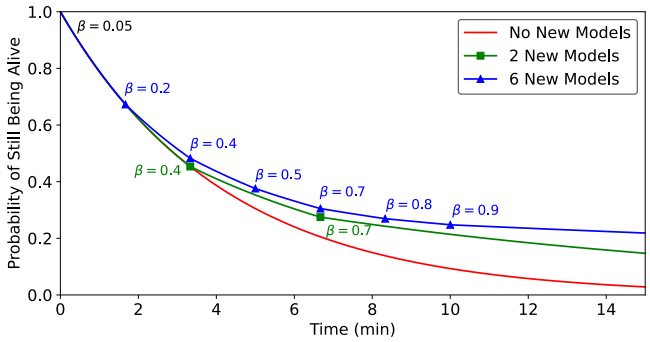
Figure 5(a) graphically illustrates this equation for different threat arrival rates (i.e., values of α), when $\beta = 0$ and $\gamma = 1.0$. In other words, SCML is nonexistent or inoperable, and hence there are no CMs. Further, every threat is lethal. As α decreases, the curve moves further to the right. Longer missions become less suicidal. The X-coordinate of the mean value of each curve represents life expectancy; it increases as α is lowered. Figure 5(b) shows the impact of different lethalities, when $\alpha = 1.0$ and $\beta = 0$. In other words, the threat arrival rate is fixed and SCML is nonexistent or inoperable. As γ decreases (i.e., threats become less lethal), the curve moves further to the right and life expectancy increases. As Figure 5(a) and (b) show, α and γ can be viewed as scaling parameters that extend life expectancy as their values are lowered. However, they do not change the shape of the curves.

Figure 6(a) and (b) show that this observation is also true when SCML is used with a pre-trained model of fixed β (i.e., no learning is done during a mission). Although life expectancy improves as β improves, the shape of the curves is unchanged. This is the state of the art in today's SCML systems: ML models are pre-trained in the cloud, and then deployed in scouts.



For all curves, $\alpha = 1.0$, $\beta = 0.05$ at $t = 0$, and $\gamma = 0.25$.

Figure 7. New models with similar asymptotic fates.



For all curves, $\alpha = 1.0$, $\beta = 0.05$ at $t = 0$, and $\gamma = 0.25$.

Figure 8. New models with divergent asymptotic fates.

Improving β during a mission changes the shape of the curves, as discussed below.

E. Model Improvement During a Mission

Suppose β_0, β_1, β_2, ..., β_N correspond to improving models that are installed at times $t_0, t_1, t_2,..., t_N$ during a single mission. Exactly how these new models are obtained by a scout can be ignored for this high-level discussion. For example, they may be transmitted wirelessly from the cloud, based on new models created via learning from other concurrent missions. Another possibility is for Live Learning on the current mission to train the models on the scouts. Regardless of how improved models are obtained, the CDF for survivability, $S(t)$, is now composed of piecewise segments corresponding to different values of β_i. During the earliest part of the mission, β_0 applies; for the next segment, β_1 applies; and so on. Using (4), the decay during each segment relative to its start can be expressed as:

$$S_0(t) = e^{-\alpha(1-\beta_0)\gamma(t-t_0)} \text{ for the piece between } t_0 \text{ and } t_1 \quad (5)$$

$$S_1(t) = e^{-\alpha(1-\beta_1)\gamma(t-t_1)} \text{ for the piece between } t_1 \text{ and } t_2 \quad (6)$$

$$\ldots$$

$$S_i(t) = e^{-\alpha(1-\beta_i)\gamma(t-t_i)} \text{ for the piece between } t_i \text{ and } t_{i+1} \quad (7)$$

Using the notation $\lambda_i = \alpha(1-\beta_i)\gamma$, a generalized expression for $S(t)$ in the interval $t_i \le t \le t_{i+1}$ is then given by:

$$S(t) = e^{-\left(\sum_{j=0}^{i-1} \lambda_j (t_{j+1}-t_j)\right) - \lambda_i (t-t_i)} \text{ for } t_i \le t \le t_{i+1} \quad (8)$$

We illustrate the subtleties of this survivability equation using two slightly different examples in Figures 7 and 8.

Figure 7 shows $S(t)$ for two hypothetical missions, indicated by the green and blue curves. Both missions improve β from a miserable value of 0.05 at the start of the mission to a respectable value of 0.7 for the last part of the mission. However, the number of steps in which this improvement occurs and their timing are different for the two missions. Both of these curves in Figure 7 diverge significantly from the red curve as the mission progresses. This confirms that learning during a mission improves survivability. The figure shows divergence between the blue and green during the period from $t = 2$ to $t = 9$. However, the two missions have the same final values of $\beta = 0.7$, and they decay at the same rate asymptotically beyond $t = 9$. In other words, regardless of the early part of their journeys, all missions arriving at a certain β will have comparable future fates beyond that point.

In the example illustrated by Figure 8, the green and blue missions reach values of $\beta = 0.4$ and $\beta = 0.7$ at the same moments in time. But their histories of model improvement are different elsewhere. Most importantly, the blue mission continues to receive model improvements of $\beta = 0.8$ and $\beta = 0.9$ beyond the last model of $\beta = 0.7$ shared with green. Hence, the later parts of the blue and green curves are divergent — suvivability is asymptotically higher for the blue mission.

F. Estimating Life Expectancy

Figures 7 and 8 qualitatively show that model improvement during a mission extends life expectancy. Quantifying this improvement requires computing the mean of a piecewise exponential distribution. Fortunately, a closed-form expression for this can be derived. The mean can be expressed as:

$$E[T] = \sum_{j=0}^{R} \int_{t_j}^{t_{j+1}} A_j t e^{-\lambda_j (t-t_j)} \quad (9)$$

In this equation, R is the number of new models installed during the mission. λ_j is the average arrival rate of lethal, unneutralized or undetected threats for interval j. The first such threat terminates the life of the mission. Improving models means that λ_j decreases over time. $t_j \to t_{j+1}$ represents the time interval over which model j performs inferencing. A_j is a scaling factor which ensures that the area under the entire probability density function is 1.0. Omitting the derivation for space reasons, this scaling factor can be shown to be:

$$A_j = \lambda_j \left[1 - \sum_{i=0}^{j-1} \frac{A_i}{\lambda_i} \left(1 - e^{-\lambda_i (t_{i+1}-t_i)}\right)\right], \forall j \quad (10)$$

Using this expression for A_j in (9), and omitting the intermediate steps of the derivation, we obtain the life expectancy for a

Figure	Curve	Life Expectancy
7	Red	4.21
	Green	6.22
	Blue	7.23
8	Red	4.21
	Green	7.16
	Blue	14.42

Figure 9. Life expectancy.

piecewise exponential as:

$$E[T] = \sum_{j=0}^{R} \left(\frac{A_j}{\lambda_j^2} \right) \left[1 - (\lambda_j t_{j+1} + 1) e^{-\lambda_j (t_{j+1} - t_j)} + \lambda_j t_j \right] \tag{11}$$

Using the above equation, Figure 9 presents the life expectancies for the curves in Figures 7 and 8. We have validated these results against Monte Carlo simulations of the missions depicted in Figures 7 and 8, and obtained identical results.

These results show that model improvement during a mission can indeed improve survivability. However, quantifying this improvement in Live Learning implementations is not so simple because of nonlinearities in learning algorithms. For a given threat arrival rate, creating more models during a mission means that each new model is based only on a small increase in training set size over the previous model. This may lead to suboptimal training. Hence, experimental results are needed to complement the intuitive understanding of Live Learning presented so far.

IV. Experimental Setup

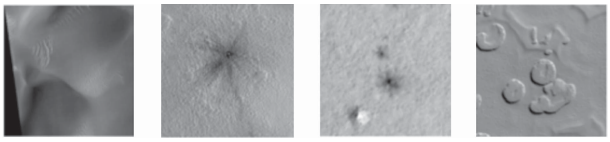
In a real implementation, do the predicted gains in survivability from Live Learning materialize? To experimentally answer this top-level question, we have created a modified version of Hawk from the open source release in GitHub [7]. Our experiments leverage ML insights from the original Hawk work [4] by closely following its experimental setup. We use the same datasets and classes, data arrival rates, ML models, hyperparameter settings, Live Learning triggering criterion for training new models, etc. Our description below first presents our modifications to Hawk for SCML (Section A), and then presents other relevant experimental details (Sections B–D).

A. Hawk Modifications for SCML

As mentioned earlier (Section II), the original Hawk implementation focused exclusively on low backhaul bandwidths (down to 12 kbps). In the context of Figure 2, this maps to severe throttling of result transmission for labeling from scouts. However, SCML has value at any network bandwidth. Human labeling bandwidth, rather than wireless network bandwidth, may sometimes be the bottleneck. We have therefore modified the Hawk implementation to use end-to-end queue backpressure to throttle result transmission from scouts. This aims to prioritize human labeling effort so that, regardless of whether the bottleneck is the human or the network, it is the most high-value results that are labeled. We achieve this using an end-to-end token-based flow control mechanism [32].

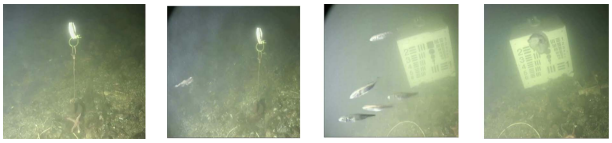
(a) Roundabout (TPs=336) (b) Swimming Pool (TPs=335) (c) Large Vehicle (TPs=357) (d) Airplane (TPs=350)

Figure 10. Examples of DOTA target classes.



(a) Dark Dune (TPs=64) (b) Impact Ejecta (TPs=64) (c) Spider (TPs=64) (d) Swiss Cheese (TPs=64)

Figure 11. Examples of HiRISE target classes.



(a) Starfish (TPs=370) (b) Shrimp (TPs=564) (c) Small Fish (TPs=564) (d) JellyFish (TPs=584)

Figure 12. Examples of Brackish target classes.

Each scout maintains a queue of results awaiting transmission. This queue is sorted by score received on the ML model that inferenced it. A data item that is processed later by a scout may receive a higher score than many earlier data items. The high score suggests that it is likely to be a TP. Since TPs are rare, they merit early attention by the human labeler in order to rapidly enlarge the training set and speed up learning. A long FIFO queue at the human labeler would delay the labeling of this high-value item. At the same time, maintaining too short a labeling queue may lead to the human labeler being stalled waiting for work during periods of low network bandwidth. Further, human think time may vary widely across data items. Some items may be labeled almost effortlessly, while a few require extended think times. The token-based mechanism balances these disparate forces. Its goal is to ensure that the data ingress rate never exceeds the bottleneck throughput, whose location and value can vary dramatically over time. As the bottleneck changes, the token flow adapts to preserve high value of human labeling effort under all conditions.

B. Datasets

Our experiments use three public datasets from drone surveillance, planetary exploration, and underwater sensing. Examples from these datasets are shown in Figures 10– 12. For all datasets and classes, the base rate is below 0.1%. While these datasets represent benign environments, any class in them could be designated as the threat class of specified lethality from an ML viewpoint. CMs are left abstract (Section B), and are not related in any way to the datasets. Real world CMs may span a wide range: projectiles, chaff, energy for evasive maneuvers, etc.

Drone Surveillance (DOTA) [Figure 10]: The drone-captured 15-class DOTA v1.0 dataset [33] consists of 2806 images, half of which are used as mission data. Image resolution ranges from 800 × 800 pixels to 4000 × 4000 pixels. Images are tiled into 256 × 256 pixels, yielding 252,231 usable tiles. During a mission, a scout receives an average of 180 tiles in 20 seconds.

Planetary Exploration (HiRISE) [Figure 11]: The HiRISE dataset [34] consists of images collected by the Mars Reconnaissance Orbiter. There are 73,031 labeled images, tiled into 227 × 227 pixels. There are 8 target classes. During a mission, a scout receives an average of 100 tiles in 20 seconds.

Underwater Sensing (Brackish) [Figure 12]: The Brackish dataset [35] contains labeled images of marine animals in a brackish strait with varying visibility. There are 14,518 images of 1080p resolution tiled into 256 × 256 pixels, for a total of 563,829 tiles. There are 6 target classes. During a mission, a scout receives an average of 100 tiles in 3 seconds.

C. Hardware

Each scout consists of a 6-core 3.6 GHz Intel Xeon E5–1650v4 processor, 32 GB memory, 4 TB disk storage for image data, and an NVIDIA GTX 1060 GPU. Today, a typical scout would be configured with embedded hardware such as an NVIDIA Jetson AGX Orin [5], weighing 1.58 kg. Such hardware could easily be carried by an autonomous drone, unmanned spacecraft, or underwater immersible. Rather than the austere network settings of the original Hawk work, we assume that ample network bandwidth (1 Gbps) is available. The bottleneck is now human labeling rate (Figure 2), emulated by code that returns the ground-truth label after a configurable think time (1 s in our experiments).

D. Mission Configuration

Although our simple analytical model (Section III) focused on missions with a single scout, our experimental evaluation mirrors the original Hawk work by using 7 scouts that share TPs. The learning rate of the whole ensemble is thus determined by their collective rate of threat encounters. The scouts have identical hardware, as described above (Section IV-C), and connectivity between them is 1 Gbps. In the context of survivability, it is necessary to specify what constitutes mission success when some scouts die but others survive. We require all scouts to reach their destination for the mission to be successful, but other criteria are possible as discussed later (Section VIII).

V. Results: Live Learning Timeline

What is the timeline of Live Learning on a scout? This question forms the backdrop for all other experimental results presented in this paper. As shown by our analytical model (Section III-E), the timing and magnitude of model improvement greatly influences survivability. When the improvement is achieved through Live Learning, its timeline depends on the dataset, attributes of the threat class (especially its base rate), and numerous ML hyperparameters.

	Class: Roundabout		Class: Pool	
Gen	Time(s)	AUC	Time(s)	AUC
0	0	0.20 (0.03)	0	0.02 (0.03)
1	146 (22)	0.29 (0.12)	560 (130)	0.23 (0.08)
2	388 (87)	0.45 (0.01)	797 (95)	0.36 (0.20)
3	547 (100)	0.47 (0.03)	932 (115)	0.51 (0.08)
4	787 (115)	0.54 (0.03)	1219 (106)	0.62 (0.03)
5	1216 (94)	0.54 (0.06)	1537 (102)	0.70 (0.03)
6	1706 (100)	0.59 (0.03)	1973 (83)	0.72 (0.07)
7	2253 (143)	0.58 (0.03)	2502 (41)	0.78 (0.04)
8	2960 (68)	0.62 (0.04)	3103 (55)	0.81 (0.06)
9			3872 (30)	0.79 (0.04)
	4010 (5)		4014 (4)	

(a) Dataset: DOTA

	Class: Impact Ejecta		Class: Swiss Cheese	
Gen	Time(s)	AUC	Time(s)	AUC
0	0	0.45 (0.05)	0	0.71 (0.04)
1	267 (48)	0.49 (0.01)	195 (28)	0.85 (0.03)
2	522 (32)	0.64 (0.07)	425 (49)	0.90 (0.03)
3	919 (116)	0.63 (0.03)	798 (71)	0.90 (0.03)
4	1449 (14)	0.69 (0.04)	1361 (42)	0.95 (0.02)
	1710 (0)		1709 (0)	

(b) Dataset: HiRISE

	Class: Starfish		Class: Jellyfish	
Gen	Time(s)	AUC	Time(s)	AUC
0	0	0.16 (0.04)	0	0.16 (0.02)
1	238 (158)	0.38 (0.06)	148 (44)	0.28 (0.03)
2	374 (164)	0.48 (0.06)	328 (47)	0.42 (0.03)
3	539 (169)	0.57 (0.06)	472 (62)	0.43 (0.02)
4	768 (133)	0.64 (0.02)	667 (26)	0.49 (0.03)
5	1082 (172)	0.67 (0.04)	907 (34)	0.52 (0.01)
6	1458 (184)	0.72 (0.03)	1287 (71)	0.55 (0.02)
7	1945 (216)	0.73 (0.04)	1750 (78)	0.57 (0.02)
8	2607 (211)	0.78 (0.01)	2211 (91)	0.59 (0.01)
9	3537 (96)	0.82 (0.01)	2909 (15)	0.61 (0.01)
10			3785 (35)	0.64 (0.01)
	4037 (5)		4037 (5)	

(c) Dataset: Brackish

These results are for representative classes of the three datasets used in our experiments (§4.2). For each class, the time into the mission at which a new generation of a model was installed and its AUC (area under the precision-recall curve for a held-out test dataset) are shown. The last row shows the time by which all data from the dataset was delivered. Numbers in parentheses are standard deviations across three runs of an experiment with different random number seeds.

Figure 13. Mission timeline for live learning.

For the class "Roundabout" from the dataset DOTA, the second and third columns of Figure 13(a) show this timeline. The initial model, based on a bootstrap training set containing just 20 TPs, is weak. Its AUC (area under the precision-recall curve on a held-out test set) is merely 0.20. However, in the face of our extreme class imbalance, even 0.20 is much better than random classification. The first new model is installed 146 seconds into the mission, and has a slightly improved AUC of 0.29. At 388 seconds, a better model with AUC 0.45 is installed; at 547 seconds, a model with AUC 0.47 is installed; and so on. The AUC improvement is mostly monotonic, with occasional plateaus or reversals due to the randomness inherent in the learning process. The final model with AUC 0.62, has clearly improved far beyond the initial model. The last two columns

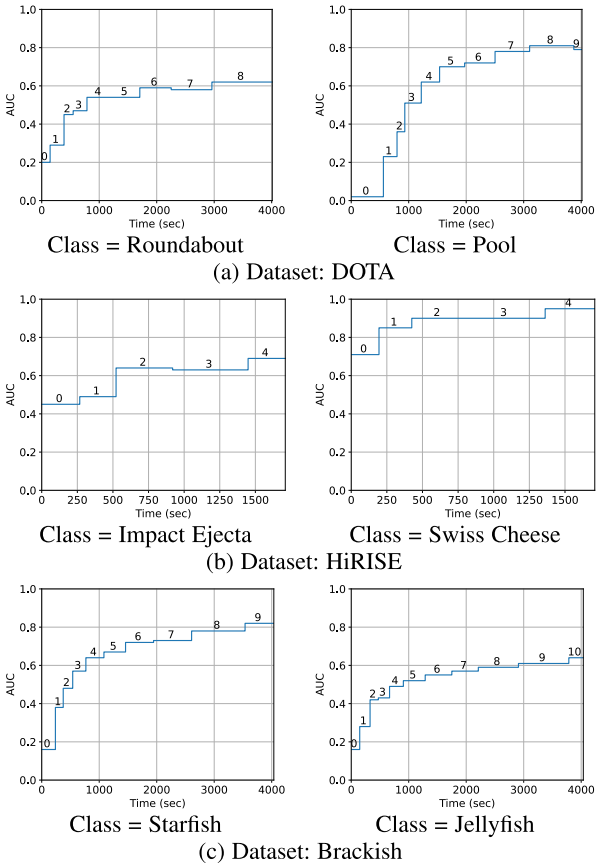


Figure 14. Visualization of live learning timeline (compare with Figure 3(b)).

Class	Tiles Processed	Tiles Labeled	TPs Found	TPs Total
Dataset: DOTA				
Round-about	252,225 (8)	4117 (216)	272 (2)	336
Pool	252,216 (18)	4346 (385)	274 (2)	335
Dataset: HiRISE				
Swiss Cheese	60,064 (0)	1529 (0)	63 (0)	64
Impact Ejecta	60,064 (0)	1529 (0)	53 (1)	64
Dataset: Brackish				
Starfish	563,755 (90)	4045 (0)	300 (7)	370
Jellyfish	563,777 (30)	4045 (0)	385 (5)	584

These results are from the same experiments as Figure 13. Numbers in parentheses are standard deviations. "TPs Total" refers to the number of TPs known to be present from ground truth.

Figure 15. Transmission & labeling efficency of live learning.

of Figure 13(a) confirm that this theme of model improvement during a mission holds for a different class of the DOTA dataset. Figure 13(b) and (c) confirm that it also holds for classes of the HiRISE and Brackish datasets.

The data in Figure 13 is visualized in Figure 14 using the canonical Live Learning timeline shown earlier in Figure 3(b). For all datasets and classes, significant improvement in model accuracy is seen over the life of the mission. The magnitude of the improvement is dataset and class dependent.

Figure 15 shows the transmission and labeling efficiency of Live Learning for the experiments shown in Figure 13. For class Roundabout of dataset DOTA, the total number of tiles processed by the scouts is 252,225. Of these, only 4117 were transmitted for labeling by the human in the loop. In spite of the extreme class imbalance (less than 0.1% base rate), 272 TPs were found out of 336 ground truth TPs. This general theme holds for all of the datasets and classes shown in Figure 15. These results confirm that Live Learning is frugal in its use of network bandwidth and human effort, but highly effective in discovering TPs.

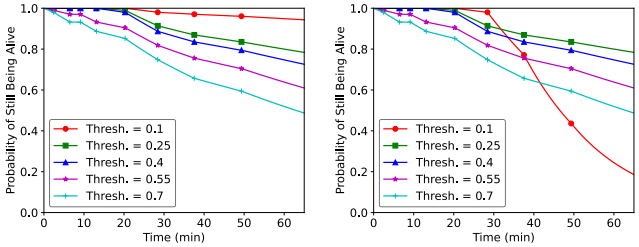
VI. Results: Importance of Model Precision

Our analytical modeling (Sections III-D and III-F) focused exclusively on the recall metric (β) of the model used for SCML. High recall ensures that almost all threats are detected. With an infinite supply of CMs, detection ensures neutralization. Our analysis ignored the complementary metric of model precision, which determines how many CMs are wasted. In practice, no real mission can carry an infinite supply of CMs. Wasted CMs hurt survivability. If the supply of CMs on a mission is exhausted, no further neutralization of threats is possible. Even perfect ML (i.e., detection of every threat beyond that point) cannot save the mission. Any threat (detected or undetected) may then be fatal, and survival is determined solely by the arrival rate of threats (α) and their lethality (γ).

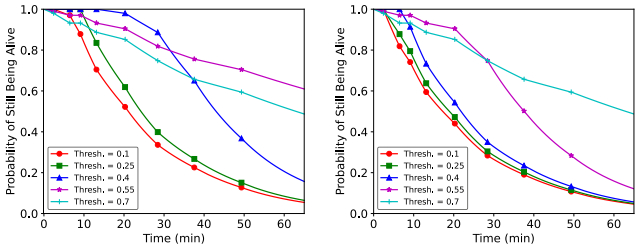
The balance between recall and precision is determined by a tunable *threshold,* ranging between 0.0 and 1.0. Data items that are scored above the threshold are deemed to be positives; the rest are deemed to be negatives. Some of the positives may be FPs, and some of the negatives may be FNs. With an ample supply of CMs, there is no incentive to be frugal — FPs don't matter at all, only FNs do. As the supply of CMs falls, FPs become problematic.

For class Roundabout of the dataset DOTA and a low threat lethality of $\gamma = 0.01$, Figure 16 shows survivability for four different initial supplies of CMs. With an infinite supply of CMs (Figure 16(a)), lowering threshold is always a win. A threshold of 0.1 results in a likelihood of survival to mission completion that is over 90%. Higher thresholds lead to lower likelihod of mission success. At a threshold of 0.7, the likelihood of success is only about 50%. Figure 16(b) shows that a threshold of 0.1 results in too many FPs when the initial supply is only 100,000 CMs. All CMs are used up by roughly 30 minutes into the mission, and survivability sharply drops after that. For this case, a threshold of 0.25 leads to the highest chances of mission success (roughly 80%). This general trend continues when CMs are limited to 20,000 (Figure 16(c)) and 10,000 (Figure 16(d)). In the latter case, the high threshold of 0.7 offers the best chances of survival to mission success.

From the viewpoint of conserving CMs, the importance of model improvement during a mission becomes clear by examining the total CMs used on successful missions. Figure 17 compares the numbers of CMs consumed by a mission using Live Learning to that consumed when no improvement is made



(a) CMs = ∞ (b) CMs = 100,000



(c) CMs = 20,000 (d) CMs = 10,000

All graphs above assume a threat lethality of $\gamma = 0.01$. These results correspond to class Roundabout for the DOTA dataset.

Figure 16. Survivability with limited CMs at 1% lethality.

Thres-hold	Live Learning	No Learning
0.10	185,313	206,148
0.25	62,155	138,454
0.40	28,290	97,308
0.55	15,102	69,766
0.70	7,904	48,399

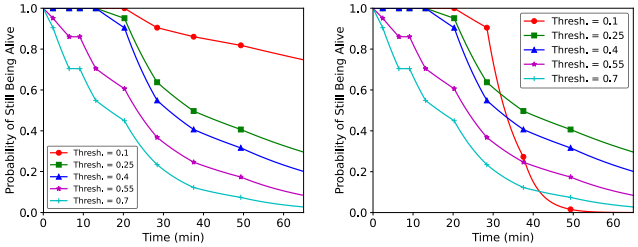
Dataset: DOTA Class: Roundabout

Figure 17. Total CM usage.

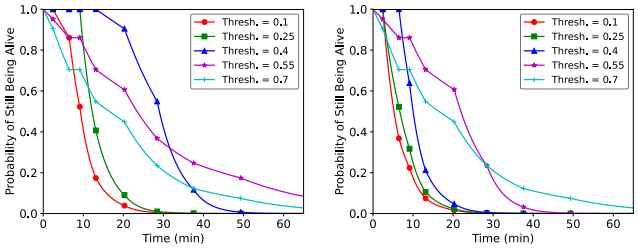
to the initial model. Note that only data from successful missions is included here. The thresholds used in Figure 17 are the same as in Figure 16. At low threshold, the difference between the CMs used in the two cases is only modest — both are equally profligate in their use of CMs. As threshold rises, the improved model quality of Live Learning begins to have substantial effect. Far fewer CMs are used at higher thresholds with Live Learning.

The lethality of threats is low ($\gamma = 0.01$) for the results shown in Figures 16 and 17. In other words, the impact of FNs is mild — there is only a 1% chance that an undetected threat will kill. What happens when threats are more lethal?

Figures 18 and 19 show the impact of increasing lethality to 5% and 10% respectively. Relative to Figure 16, the drop in survivability is apparent in all cases. Even with an infinite number of CMs and a threshold of 0.1, Figure 18 shows a mission success below 80%. Figure 19 shows a mission success barely above 50%. These are in contrast to a value well above 90% in Figure 16. At higher lethality, running out of CMs is also riskier. With a limit of 20,000 CMs or lower, Figure 19(c) and (d) show that there is virtually no hope of mission success at any threshold. With 100,000 CMs, Figure 19(b) shows a glimmer of hope for mission success at thresholds of 0.25 and 0.4, but none at other thresholds. The same general pattern holds in Figure 18.



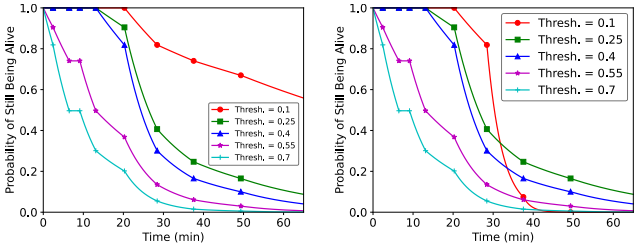
(a) CMs = ∞ (b) CMs = 100,000



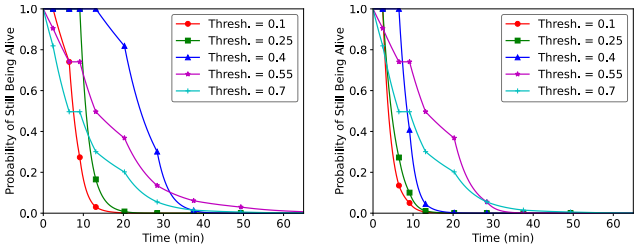
(c) CMs = 20,000 (d) CMs = 10,000

All graphs above assume a threat lethality of $\gamma = 0.05$. These results correspond to class Roundabout for the DOTA dataset.

Figure 18. Survivability with limited CMs at 5% lethality.



(a) CMs = ∞ (b) CMs = 100,000



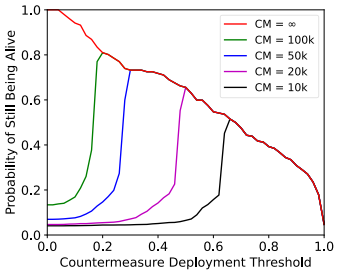
(c) CMs = 20,000 (d) CMs = 10,000

All graphs above assume a threat lethality of $\gamma = 0.10$. These results correspond to class Roundabout for the DOTA dataset.

Figure 19. Survivability with limited CMs at 10% lethality.

The same general pattern of results also holds across diverse classes and datasets, but we omit the relevant graphs due to space limitations.

If a certain number of CMs are available at the start of a mission, what threshold should a scout select for highest likelihood of success? If too low a threshold is selected, CMs may be exhausted before the end of the mission. This is likely to lead to mission failure, because of inability to neutralize a correctly detected threat late in the mission. If too high a threshold is selected, recall will be unnecessarily low. This will also hurt



Total CMs	Opt. Threshold	Success Likelihood
10,000	0.65	53.1%
20,000	0.49	65.6%
50,000	0.29	74.0%
100,000	0.19	82.6%

Dataset: DOTA Class: Roundabout $\gamma = 0.01$

The Y axis of the graph on the left shows the probability of still being alive at the end of the mission. The table on the right gives numeric values extracted from the graph on the left.

Figure 20. Optimal threshold for mission success.

the chances of survival to mission completion. The optimal threshold results in the last CM being consumed just before mission completion. Ending the mission with CMs left over does not enhance survivability.

With the number of CMs as a parameter, Figure 20 shows how threshold affects likelihood of mission completion. With an infinite number of CMs (red curve), there is no point at which CMs are exhausted. Reducing threshold therefore always improves survivability. Operating with a threshold close to zero is optimal. For all cases involving a finite number of CMs, the point of intersection of the relevant curve with the red curve gives the optimal threshold. Any lower a value causes CMs to be used up too soon. Any higher a value, leads to CMs being left over at the end of a completed mission.

VII. Results: Recall-Biased Loss Functions

Our analysis and experimental results from the previous sections show that in SCML:

- Recall is always important.
- Precision may or may not be important, depending on the CMs available relative to mission duration and threat density. As long as a scout does not run out of CMs, precision can be traded off for improved recall.

These observations lead to the shape of the precision-recall curve playing an important role in SCML. In other words, it is not just the AUC of a model that matters, but also how that AUC is achieved. This is illustrated by Figure 21, which shows two hypothetical precision-recall curves that have an AUC of 0.5. Figure 21(a) trades off precision and recall evenly everywhere. In contrast, Figure 21(b) is biased in favor of recall at low precision. If one knows ahead of time that a mission can be executed at low precision, then Figures 21(b) is preferable to 21(a). Live Learning can then be tuned to preferentially improve recall rather than precision. In the original Hawk implementation, FNs resulted in missed TPs. However, because discards are revisited later using improved models, "near misses" can be rediscovered and correctly interpreted as TPs. On the other hand, FPs always waste bandwidth. The incentives are different in SCML, where a single FN can be fatal. This suggests biasing learning differently from the original Hawk implementation. Tuning for bias can be achieved via the loss function used for training new models

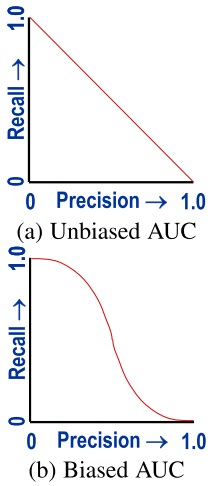


Figure 21. AUC = 0.5 .

Threshold	Unbiased		Biased	
	CMs	FNs	CMs	FNs
0.10	185,313	6	251,255	0
0.25	62,155	25	223,433	3
0.40	28,290	33	126,078	12
0.55	15,102	51	49,378	26
0.70	7,904	74	21,587	45

Dataset: DOTA Class: Roundabout

Figure 22. Impact of biased loss function.

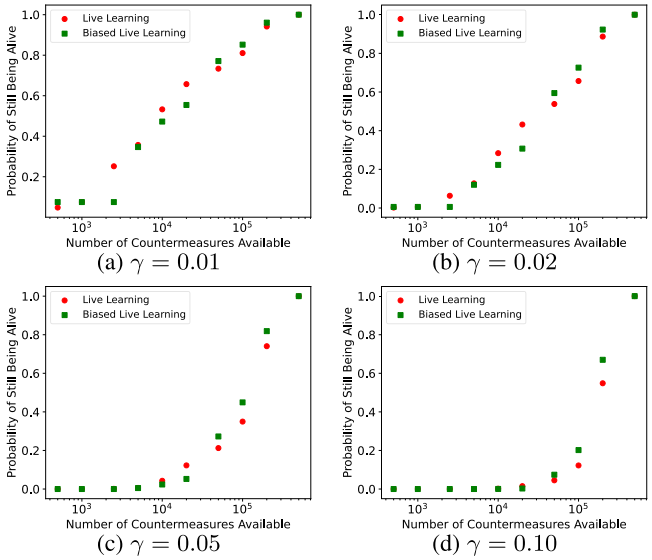
FNs	Unbiased		Biased	
	CMs	TH	CMs	TH
0-1	237,043	0.05	239,900	0.20
12-13	127,455	0.15	126,078	0.40
25-26	62,155	0.25	49,738	0.55
40-42	81,719	0.50	28,304	0.65
74-76	7,904	0.70	8,961	0.85

Dataset: DOTA Class: Roundabout

Figure 23. Targeting a specific FN range.

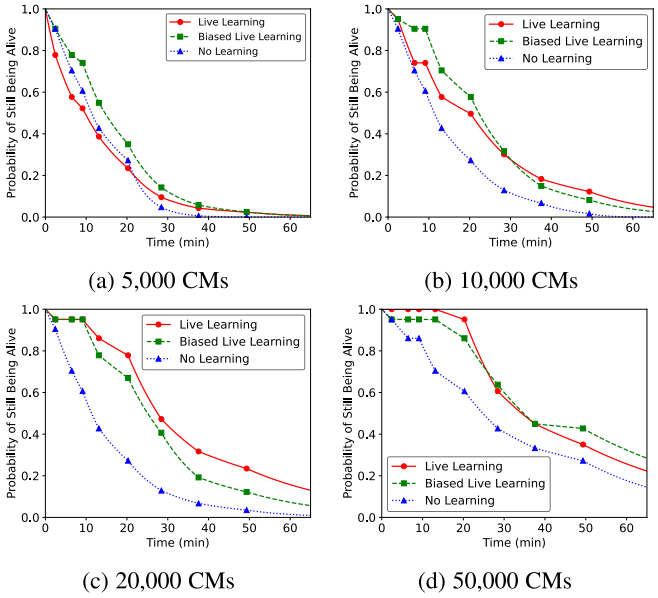
during Live Learning. Our experiments use a biased loss function that weights classification errors on positives (i.e., FNs) twice as heavily as errors on negatives. Hawk's original loss function weights errors on positives and negatives equally.

For class Roundabout of dataset DOTA, Figure 22 compares CMs used and FNs for these two different loss functions. At all thresholds, Figure 22 shows fewer FNs at the price of higher CM usage. Figure 23 uses a different viewpoint to compare biased and unbiased loss functions. It asks "If the total FNs during a completed mission should lie within a targeted range, at what thresholds should the models operate? How many CMs do they use at those thresholds?" At very few FNs (0–1, 12–13), Figure 23 shows that both biased and unbiased models use roughly the same number of CMs; however, their optimal thresholds are quite different. For intermediate ranges of FNs (25–26 and 40-42), the biased models use significantly fewer CMs at optimal threshold. At the high end (74–76 FNs), the situation is reversed: the unbiased model uses noticeably fewer CMs. Thus,



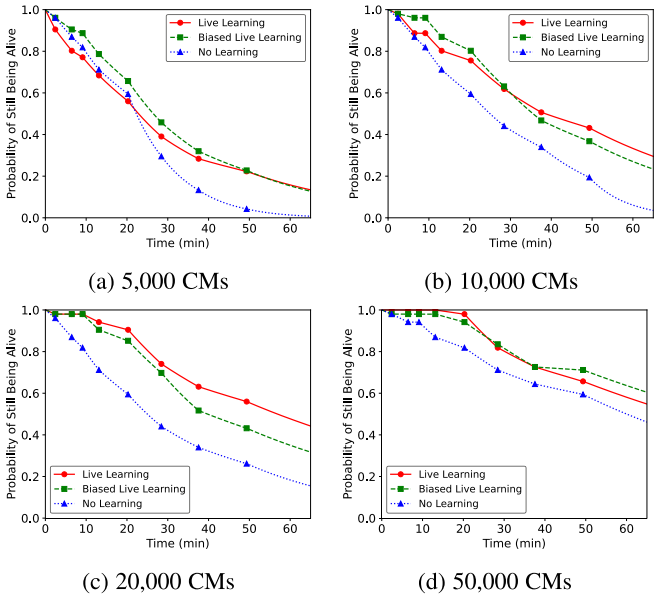
These results correspond to class Roundabout for dataset DOTA. For each data point, the survivability shown is for its optimal threshold (i.e., at which survivability is highest). Note that the X axis is in log scale.

Figure 24. Impact of loss function bias on survivability.



These results correspond to class Roundabout for dataset DOTA, and are obtained from the experiment for Figure 24(c).

Figure 25. Survivability at optimal threshold for $\gamma = 0.05$.



These results correspond to class Roundabout for dataset DOTA, and are obtained from the experiment for Figure 24(b).

Figure 26. Survivability at optimal threshold for $\gamma = 0.02$.

the choice of the loss function should be informed by the desired target FN range.

The impact on survivability of a biased loss function is shown by Figure 24. Note that the X axis is in log scale in these graphs. For each data point on this scatter plot, the survivability shown is for the optimal threshold. This is the threshold at which survivability is highest for that number of CMs and lethality. The benefit of bias is most apparent at a lethality of $\gamma = 0.10$ (Figure 24(d)). For all CM values, the survivability with a biased loss function is equal to or higher than that attained using an unbiased loss function.

At the lower lethality of $\gamma = 0.05$ (Figure 24(c)), the benefit of a biased loss function is still visible. As before, the biased loss function does as well or better than the unbiased loss function. The sole exception is for 20,000 CMs.

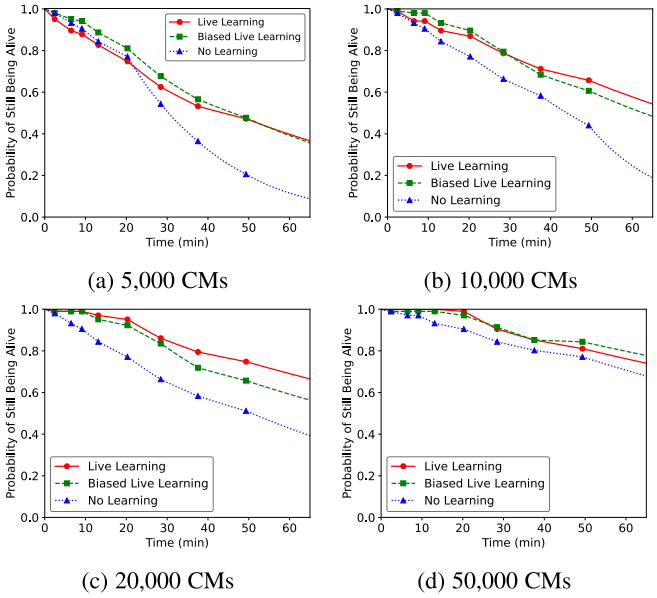
When lethality is further reduced (Figure 24(b) and (a)), there is no longer a clear win from a biased loss function. At these lower lethalities, the penalty for an FN is small. The win from biasing for fewer FNs is much less now, and is swamped by other attributes of the loss function.

Figure 25 drills deeper into the data from Figure 24(c). For four different points on the X axis of Figure 24(c) (i.e., different CM values), Figure 25(a) through Figure 25(d) show the full survivability curve for biased and unbiased loss functions. The heights of the points in Figure 24(c) correspond to the final Y values at the end of a mission for the curves in Figure 25. For calibration, the "No Learning" curve is also shown. This corresponds to the initial model remaining unchanged throughout a mission. The importance of Live Learning (biased or unbiased) relative to No Learning is clearly seen in Figure 25. The message from Figure 25 reinforces the message from Figure 24(c): at higher CM usage (50,000 CMs), the biased loss function improves survivability.

For lethalities of $\gamma = 0.02$ and $\gamma = 0.01$, Figures 26 and 27 enable in-depth comparison of biased, unbiased and No Learning cases. As expected, No Learning is always inferior to Live Learning. Between biased and unbiased flavors of Live Learning, there is not a clear winner at lower lethality.

VIII. Future Extensions

As the first work to explore the convergence of survivability and ML, this paper has explored four major sets of questions:



(a) 5,000 CMs (b) 10,000 CMs

(c) 20,000 CMs (d) 50,000 CMs

These results correspond to class Roundabout for dataset DOTA, and are obtained from the experiment for Figure 24(a).

Figure 27. Survivability at optimal threshold for $\gamma = 0.01$.

- *Can a simple analytical model provide useful input about survivability in the context of Live Learning? What broad insights does such analysis reveal?* [Section III]
- *How fast does model improvement happen with Live Learning? Does model improvement lead to improved survivability? How robust are these improvements relative to the threat environment?* [Sections V and VI]
- *When the supply of CMs is limited, how should precision and recall be balanced for optimal survivability?* [Section VI]
- *Should precision and recall be asymmetrically targeted during Live Learning? Under what conditions does such asymmetry help to improve survivability?* [Section VII]

In answering these questions, our work has opened the door to many topics of further research. We briefly discuss some of these topics below, in no particular order of presentation.

Adaptive Thresholding: One possible way to improve upon the work presented here would be to dynamically adapt the threshold used for classification as a mission progresses. Early in a mission, when the model is weak, it makes sense to use a low threshold in order to ensure good recall (i.e., high β). As the model improves, the threshold can be raised without unduly hurting β. How best to do this adaptation is an open question. It will likely need to take into account the supply of CMs. Especially when CMs are precious, it may be optimal to waste more of them early in the mission because the model is weak.

Non-uniform threat profile: Our analysis (Section III) has shown the importance of threat arrival rate (α) and threat lethality (γ). These parameters have been assumed to be constant throughout a mission. In real-world settings, this may not be a valid assumption. In military settings, certain high-value targets may have higher threat densities and lethalities than low-value targets. When stealth is important, long mission duration increases the risk of detection and, hence, threat profile. Understanding SCML in the context of non-uniform spatial and temporal threat profiles will be important.

M-out-of-N success criteria: This work has assumed that when multiple scouts are used, they all have to reach the destination for the whole mission to succeed. In many real-world settings, a weaker success criterion may be acceptable. As long as M out of N scouts reach their destination, success may be assured. This complicates Live Learning because the total rate of encounters with threats changes as scouts die off. Both analytical and experimental investigation of optimal strategies for M-out-of-N success criteria will thus be important.

Convoy/Swarm Replacements and Relays: This work assumes that when N scouts are available, they are launched together as a convoy or swarm with the same initial model. In contrast, if the launch of individual scouts is temporally staggered, the later scouts can benefit from the learning of the earlier scouts. Of course, the rate of that learning will be slower because fewer threats are encountered by the smaller initial cohort of scouts. How these factors balance out would be valuable to study. A variant of this approach is to maintain a fixed number of scouts in flight. When a scout dies, a new one is launched to replace it. That scout can start with the best model learned so far.

Real-world Threat Distributions: Our analysis has assumed a simple exponential model for threat arrivals. The experimental results use the threat distribution embedded in the DOTA, HiRISE, and Brackish datasets. Real-world applications of SCML may face very different threat distributions. Obtaining empirical data on real-world threat distributions and using them in Live Learning experiments would be valuable.

Wider Exploration of Loss Functions: Our work has looked at a specific way of biasing the loss function for Live Learning. A broader investigation of loss functions, possibly including adaptation during a mission, would be valuable.

Exploration of Hawk Hyperparameter Space: The current system inherits many ML hyperparameters, such as the trigger for training a new model, from the original Hawk implementation. It would be valuable to investigate how these hyperparameters should be tuned to optimize SCML.

Relationship to edge-cloud reinforcement learning (RL): Survival is the ultimate reward. In that sense, SCML could be viewed as a form of RL. Establishing a theoretical basis to more deeply connect RL with Live Learning would be valuable.

Biological Immunity: This paper began by pointing to biological immunity as the inspiration for SCML. Coming full circle, it would be valuable to evaluate the immune systems of organisms from the viewpoint of Live Learning. Humans are highly evolved, and likely to be too complex for such evaluation. However, much simpler organisms may prove to be tractable.

IX. Conclusion

In this paper, we have introduced and explored the concept of SCML using both an analytical approach and an experimental evaluation of a real implementation. The insights from these components complement each other. Our analysis highlights the

importance of model improvement during a mission. Our experimental results quantify the insights from analysis, and provide baseline measurements upon which future improvements can be judged. Our work is applicable to any edge-based system that operates in a threat environment that continuously evolves, and hence requires the full power of learning for survival. We show that the recently developed pipelining approach of Live Learning is an excellent fit for SCML. Our experimental results confirm the effectiveness of Live Learning as a mechanism for SCML.

Acknowledgment

The authors would like thank to the anonymous reviewers for their thoughtful and constructive feedback that improved the presentation of this paper. The content of the information does not necessarily reflect the position or the policy of the government and no official endorsement should be inferred. This work was done in the CMU Living Edge Lab, which is supported by Intel, Arm, Vodafone, Deutsche Telekom, CableLabs, Crown Castle, InterDigital, Seagate, Microsoft, the VMware University Research Fund, IAI, and the Conklin Kistler family fund. Any opinions, findings, conclusions or recommendations expressed in this document are those of the authors and do not necessarily reflect the view(s) of their employers or the above funding sources.

References

[1] E. Sturzinger, S. George, and M. Satyanarayanan, "Tactical agility for AI-enabled multi-domain operations," *Proc. SPIE*, vol. 12538, pp. 209–220, Jun. 2023.
[2] "Better camouflage is needed to hide from new electronic sensors," *Economist*, Mar. 29, 2023. [Online]. Available: https://www.economist.com/science-and-technology/2023/03/29/better-camouflage-is-needed-to-hide-from-new-electronic-sensors
[3] C. Feng et al., "Programmable microfluidics for dynamic multiband camouflage," *Microsystems Nanoeng.*, vol. 9, no. 1, Apr. 2023, Art. no. 43.
[4] S. George, H. Turki, Z. Feng, D. Ramanan, P. Pillai, and M. Satyanarayanan, "Low-bandwidth self-improving transmission of rare training data," in *Proc. 29th Annu. Int. Conf. Mobile Comput. Netw.*, Madrid, Spain, Oct. 2023, pp. 1–15.
[5] "NVIDIA Jetson Orin," 2024. Accessed: Jul. 22, 2024. [Online]. Available: https://www.nvidia.com/en-us/autonomous-machines/embedded-systems/jetson-orin/
[6] T. Li, A. K. Sahu, A. Talwalkar, and V. Smith, "Federated learning: Challenges, methods, and future directions," *IEEE Signal Process. Mag.*, vol. 37, no. 3, pp. 50–60, May 2020.
[7] S. George, J. Harkes, T. Eiszler, and E. Sturzinger, "Hawk source code," 2023. Accessed: Jul. 22, 2023. [Online]. Available: https://github.com/cmusatyalab/hawk
[8] M. Greenwood, "The first life table," *Notes Records Roy. Soc. London*, vol. 1, no. 2, pp. 70–72, 1938.
[9] S. Richards, "A handbook of parametric survival models for actuarial use," *Scand. Actuarial J.*, vol. 2012, no. 4, pp. 233–257, 2012.
[10] A. Barakat, A. Mittal, D. Ricketts, and B. A. Rogers, "Understanding survival analysis: Actuarial life tables and the Kaplan–Meier plot," *Brit. J. Hosp. Med.*, vol. 80, no. 11, pp. 642–646, 2019.
[11] E. W. Frees, J. Carriere, and E. Valdez, "Annuity valuation with dependent mortality," *J. Risk Insurance*, vol. 63, pp. 229–261, 1996.
[12] C. Czado and F. Rudolph, "Application of survival analysis methods to long-term care insurance," *Insurance: Math. Econ.*, vol. 31, no. 3, pp. 395–413, 2002.
[13] R. Swaminathan and H. Brenner, "Stastistical methods for cancer survival analysis," *IARC Sci. Pub.*, vol. 162, pp. 7–13, 2011.
[14] D. Collett, *Modelling Survival Data in Medical Research*. Boca Raton, FL, USA: CRC Press, 2023.
[15] E. Domingo and J. Holland, "RNA virus mutations and fitness for survival," *Annu. Rev. Microbiol.*, vol. 51, no. 1, pp. 151–178, 1997.
[16] L. Ohno-Machado, "Modeling medical prognosis: Survival analysis techniques," *J. Biomed. Informat.*, vol. 34, no. 6, pp. 428–439, 2001.
[17] X. Liu, *Survival Analysis: Models and Applications*. Hoboken, NJ, USA: Wiley, 2012.
[18] A. Mathew, M. Pandey, and N. Murthy, "Survival analysis: Caveats and pitfalls," *Eur. J. Surg. Oncol.*, vol. 25, no. 3, pp. 321–329, 1999.
[19] E. L. Kaplan and P. Meier, "Nonparametric estimation from incomplete observations," *J. Amer. Stat. Assoc.*, vol. 53, no. 282, pp. 457–481, 1958.
[20] J. T. Rich, J. G. Neely, R. C. Paniello, C. C. Voelker, B. Nussenbaum, and E. W. Wang, "A practical guide to understanding Kaplan-Meier curves," *Otolaryngol.-Head Neck Surg.*, vol. 143, no. 3, pp. 331–336, 2010.
[21] M. Biswal and M. Kumar, "Statistical reliability analysis of interplanetary spacecraft operating at different interplanetary extremity," in *Proc. Region VII Student Paper Competition AIAA Sydney Sect. Student Conf.*, 2020.
[22] N. R. Boone and R. A. Bettinger, "Spacecraft survivability in the natural debris environment near the stable Earth-moon lagrange points," *Adv. Space Res.*, vol. 67, no. 8, pp. 2319–2332, 2021.
[23] M. Trisolini, H. G. Lewis, and C. Colombo, "Spacecraft design optimisation for demise and survivability," *Aerosp. Sci. Technol.*, vol. 77, pp. 638–657, 2018.
[24] M. G. Richards, D. E. Hastings, D. H. Rhodes, and A. Weigel, "Defining survivability for engineering systems," in *Proc. Conf. Syst. Eng. Res.*, 2007, pp. 1–12.
[25] R. E. Ball, *The Fundamentals of Aircraft Combat Survivability: Analysis and Design*. Reston, VA, USA: Amer. Inst. Aeronaut. Astronaut., 2003.
[26] G. L. Guzie, "Integrated survivability assessment," Survivability/Lethality Analysis Directorate, Army Research Laboratory, White Sands Missile Range, NM, Tech. Rep. ARLTR-3186, 2004.
[27] V. R. Westmark, "A definition for information system survivability," in *Proc. IEEE 37th Annu. Hawaii Int. Conf. System Sci. 2004*, 2004, p. 10.
[28] S. Jha and J. M. Wing, "Survivability analysis of networked systems," in *Proc. Int. Conf. Softw. Eng.: Proc. 23rd Int. Conf. Softw. Eng.*, Toronto, ON, Canada, vol. 12, no. 19, 2001, pp. 307–317.
[29] A. Ayara and F. Najjar, "A formal specification model of survivability for pervasive systems," in *Proc. 2008 IEEE Int. Symp. Parallel Distrib. Process. Appl.*, 2008, pp. 444–451.
[30] P. Wang, Y. Li, and C. K. Reddy, "Machine learning for survival analysis: A survey," *ACM Comput. Surv.*, vol. 51, no. 6, pp. 1–36, 2019.
[31] J. L. Katzman, U. Shaham, A. Cloninger, J. Bates, T. Jiang, and Y. Kluger, "DeepSurv: Personalized treatment recommender system using a cox proportional hazards deep neural network," *BMC Med. Res. Methodol.*, vol. 18, no. 1, pp. 1–12, 2018.
[32] K. Ha, Z. Chen, W. Hu, W. Richter, P. Pillai, and M. Satyanarayanan, "Towards wearable cognitive assistance," in *Proc. 12th Annu. Int. Conf. Mobile Syst., Appl., Serv.*, 2014, pp. 68–81.
[33] G.-S. Xia et al., "DOTA: A large-scale dataset for object detection in aerial images," in *Proc. 2018 IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 3974–3983.
[34] G. Doran, S. Lu, L. Mandrake, and K. Wagstaff, "Mars orbital image (HiRISE) labeled data set version 3," NASA, Washington, DC, USA, 2019, doi: 10.5281/zenodo.2538136.
[35] M. Pedersen, J. B. Haurum, R. Gade, and T. Moeslund, "Detection of marine animals in a new underwater dataset with varying visibility," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. Workshops*, 2019, pp. 18–26.

Eric Sturzinger received the BS degree in electrical engineering from Oregon State University, Corvallis, OR, USA, and the MS degree in electrical and computer engineering from the University of California, Davis, Davis, CA, USA. He is currently working toward the PhD degree in computer science with Carnegie Mellon University, Pittsburgh, PA, USA. His research interests include continuous machine learning at the edge, distributed open world recognition, and survivable ML systems.

Jan Harkes received the MSc degree in computer science from the Vrije Universiteit, Amsterdam, The Netherlands. He is currently a principal project scientist with the Computer Science Department, Carnegie Mellon University, Pittsburgh, PA, USA. His research interests include distributed systems, storage, security, Internet of Things, and edge computing.

Padmanabhan (Babu) Pillai (Member, IEEE) received the BS degree in electrical and computer engineering from Carnegie Mellon University, Pittsburgh, PA, USA, and the MS and PhD degrees in computer science from the University of Michigan, Ann Arbor, MI, USA. He is currently a senior research scientist with Intel Labs, Santa Clara, CA, USA. His research interests include the use of edge computing to support immersive interactive experiences, and perform real-time understanding of the world.

Mahadev (Satya) Satyanarayanan (Life Fellow, IEEE) received the PhD degree in computer science from Carnegie Mellon University (CMU). He is currently the Jaime Carbonell University professor of computer science with CMU, Pittsburgh, PA, USA. His multidecade research career has focused on the challenges of performance, scalability, availability, and trust in information systems that reach from the cloud to the mobile edge of the Internet. He is a Fellow of ACM.