Low-Bandwidth Self-Improving Transmission of Rare Training Data

Shilpa George, Haithem Turki, Ziqiang Feng, Deva Ramanan,
Padmanabhan Pillai†, Mahadev Satyanarayanan
Carnegie Mellon University and †Intel Labs

Abstract

A severe bandwidth mismatch between incoming sensor data rate and wireless backhaul bandwidth often exists on unmanned probes when collecting new training data for machine learning (ML). To overcome this mismatch, we describe a self-improving ML-based transmission system called *Hawk*. Starting from a weak model that is trained on just a few examples, it 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, called *Live Learning,* continues until a sufficient number of TPs have been collected. For very rare events on challenging datasets, and bandwidths as low as 12 kbps, a team of 7 probes using Hawk discovers up to 87% of the TPs that could have been discovered via full preview, transmission and labeling of all mission data. Hawk also uses diversity sampling and few-shot learning.

CCS Concepts

• **Information systems** → **Mobile information processing systems**; • **Networks** → **Wireless access networks**; **In-network processing**; • **Computing methodologies** → **Active learning settings**; **Semi-supervised learning settings**; **Online learning settings**; **Object detection**; **Object recognition**; • **Human-centered computing**;

Keywords

Edge Computing, Machine Learning, Computer Vision, Wireless Networks, Acoustic Networks, Remote Sensing, LPWAN

ACM MobiCom '23, October 2–6, 2023, Madrid, Spain

ACM ISBN 978-1-4503-9990-6/23/10.
https://doi.org/10.1145/3570361.3613300

ACM Reference Format:
Shilpa George, Haithem Turki, Ziqiang Feng, Deva Ramanan, Padmanabhan Pillai, Mahadev Satyanarayanan. 2023. Low-Bandwidth Self-Improving Transmission of Rare Training Data. In *The 29th Annual International Conference on Mobile Computing and Networking (ACM MobiCom '23), October 2–6, 2023, Madrid,Spain.* ACM, New York, NY, USA, 15 pages. https://doi.org/10.1145/3570361.3613300

1 Introduction

Autonomous unmanned probes such as aerial drones, satellites, interplanetary spacecraft, and underwater drones are often used to collect new ML training data in domains such as planetary exploration, oceanography, climate change tracking, and wildlife conservation. Captured data is returned via wireless network transmission. The probe itself may not return. In such missions, there is often a severe mismatch between incoming sensor data rate and backhaul bandwidth. A 4K video camera has a bandwidth demand of ~30 Mbps, but backhaul bandwidth from an interplanetary space probe is only a few tens of kbps [39]. Underwater acoustic communication is limited to just a few tens of kbps [21]. LPWAN-based IoT systems [35] experience similar bandwidth constraints.

Collecting training data from such missions is difficult. The most valuable training data is often rare. For example, a space probe may seek new training data for a surprising geological phenomenon that was visible in just a few images from an earlier mission to a distant planet. In marine biology, the goal may be to build a training set for a newly-discovered deep-sea species of which just a few images are available. The rarity of an event is expressed by its *base rate* [33], with low values mapping to *extreme class imbalance* in ML. In this paper, we only consider base rates lower than 0.1%. In other words, fewer than one in a thousand observations is relevant.

This trifecta of extremes (low bandwidth, class imbalance, and novel target) leads to a "perfect storm" for remote sensing. Ideally, only mission-relevant data should be transmitted. If an accurate pre-trained ML model for the target class is already available, selective transmission is easy. The occasional false positives (FPs) will waste little bandwidth. If, however, the whole point of the mission is to collect new true positives (TPs) to construct a larger training set, we have a chicken-or-egg problem. No existing ML approach works at

this convergence of extremes. We cannot afford the bandwidth to transmit all data for labeling. Federated learning [28] requires labeled data. Unsupervised learning is not satisfactory because of too few TPs and extreme class imbalance. Few-shot learning (FSL) [6, 12, 41, 52, 56] can build an initial model, but improving it requires more training data. With so many vexing constraints, how do we proceed?

Hawk is a self-improving selective transmission system that addresses this problem. Hawk complements FSL with an iterative workflow called *Live Learning* that seamlessly pipelines semi-supervised learning (SSL), active learning, and transfer learning, with asynchronous bandwidth-sensitive data transmission to a distant human for labeling. By transmitting a small subset of the enriched sensor stream, new TPs can be discovered via human labeling. These can be used to grow the training set and re-train the model. Over many rounds, this iterative workflow is very effective. Hawk supports diverse deep neural network (DNN) designs and training strategies. It leverages ZeroMQ [2] for weakly-connected operation.

Our experimental results show that even at 12 kbps, Hawk is effective on data from drone surveillance, planetary exploration, and underwater sensing. The result preview in Figure 1 compares the number of TPs discovered during a mission at 12 kbps using four different models for selective transmission: (a) Hawk, (b) an oracle, (c) a model trained by brute force transmission and labeling of all data and (d) Hawk with no learning during the mission. Hawk discovers up to 87% of the TPs that would have been discovered by brute force. Our extensive experimental evaluation (§6 and §7) confirms similar results across diverse datasets and classes.

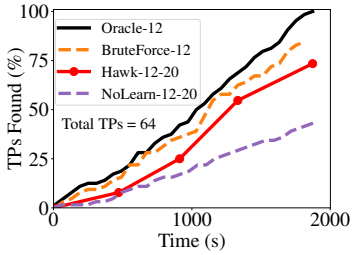


Figure 1: Result Preview

This work makes four contributions. First, it identifies a new problem of growing importance at the intersection of remote sensing, wireless networking, and ML. Second, it presents the design and implementation of the first system to address this problem. Third, it experimentally validates this system across multiple state-of-the-art datasets and challenging operating conditions. Fourth, it shows how recent ML advances can be leveraged for further improvement.

2 Background and Related Work

2.1 Compute, Storage and Power

Hawk assumes that probes have ample compute and storage, comparable to a gaming laptop today. We use the term *scout* to refer to a probe that is powerful enough to run Hawk. In the device-cloudlet-cloud taxonomy [44], a scout integrates Tier-3 (device) and Tier-2 (cloudlet) into one package.

The experiments reported in this paper used 5-year old servers as scout compute. Each has a 6-core 3.6 GHz Intel® Xeon®processor with 32 GB memory, 4 TB of storage, and an NVIDIA®GTX 1060 GPU. Both inferencing and training of DNNs are possible on such a platform. Today, a high-end gaming laptop such as the HP Omen-16 [22] has comparable processing, memory, GPU, and storage specifications. It weighs ~5.3 pounds with its battery, and includes a display that is not needed by a scout. For short missions, such a platform can be powered by its battery alone. On multi-hour missions, the battery can be recharged via the scout's propulsion system. On extended (multi-day to multi-year) missions without continuous propulsion, it can be powered by secondary fuel cells based on Polymer Electrolyte Membrane (PEM) technology [3] that are recharged from solar panels.

A plausible scout is the ScanEagle drone operated by the U.S. Navy [23]. It has a payload capacity of 11 pounds, and a takeoff weight of 58 lb. It is powered by a liquid-fuel engine that gives an endurance of 18 hours and a service ceiling of 19,000 feet. The payload limit of such a drone easily meets the weight requirements of compute for a scout. On sensitive missions, the need for low observability and jamming resistance may severely constrain wireless network bandwidth [5]. A plausible scout for planetary exploration is NASA's Curiosity rover on Mars [40]. At the size of an SUV, it can easily carry the compute for Hawk.

2.2 Learning from Edge-sourced Data

Standard ML practice today uses supervised learning. Data is captured on mobile or IoT devices, sent to the cloud, labeled there by humans, and then used for training. In a low-bandwidth setting, sending all data to the cloud is hideously impractical. At a backhaul bandwidth of 12 kbps, transmitting just a single full-resolution frame of data from a 4K video camera would take nearly 45 minutes!

Federated learning [28] is a well-known decentralized approach that avoids shipping data to the cloud. Unfortunately, it assumes that data is already pre-labeled. Since scouts are unmanned, obtaining labels is only possible by transmitting at least thumbnails of images to a distant human. This defeats the whole purpose of federated learning, especially since the privacy benefit of federated learning is not relevant to Hawk.

Further, federated learning assumes much higher bandwidth than is available in Hawk settings. It is often used for training on data across smartphones with 4G or 3G connectivity. The lower end of this range (3G) is typically a few hundred kbps to a few Mbps. This is at least one to two orders of magnitude higher than the worst-case bandwidth on scouts. At 12 kbps, transmitting a thumbnail that is $256x256$ 1-byte pixels would take about 40 seconds. This is much less than the 45 minutes for the full-resolution image, but still expensive.

The work closest to Hawk is *pseudo-labeling* for SSL [27]. Here, a weak model is used to inference unlabeled training data, and then these scores (i.e., class probabilities) are used to deduce labels for training a better model. Unfortunately, pseudo-labeling is vulnerable to *confirmation bias* [7], where erroneous pseudo-labels early in the evolution can bias the entire trajectory of models along a path that is resistant to self-correction. Recent work [26] shows that this danger is especially high with extreme class imbalance. This is exactly the situation in Hawk settings. To reduce this risk, Cascante-Bonilla et al [11] recommend curriculum learning [8]. This grades training examples from easy to hard, and performs training on the easy examples first. This is feasible in the cloud, but infeasible in the stream setting of Hawk.

Also relevant is the long history of work in active learning [45]. Shao et al's work on learning to sample [46] is a recent example. The primary focus of active learning is to optimize labeling effort by humans. This is in contrast to Hawk, where network bandwidth is the crucial resource. The cloud setting assumed by active learning permits its algorithms to preview the entire corpus of data at least once before they start presenting data for labeling. Hawk, in contrast, has to perform expensive data transfers over a low-bandwidth network before any preview at the cloud is possible.

Continuous learning [15, 31, 36, 48] has been an important ML theme over three decades. Hawk can be viewed as the first system to demonstrate continuous learning in a setting where obtaining labels for data dominates all other costs.

2.3 Learning with Highly Imbalanced Classes

Hawk focuses on extreme class imbalance (i.e., rare target classes). The ML community often addresses such tasks in the context of Long-Tailed Recognition [32, 53, 55], where the goal is to maximize accuracy across both common and rare classes. Such formalisms typically assume labeled data and ignore data transmission cost. Recent work has focused exclusively on rare class accuracy. For example, Background Splitting [37], focuses on assembling a good training set from labeled data with a small number of rare categories. Most related to Hawk is Imbalanced Active Learning [38], which focuses on the problem of selecting a small number of items to label from a large pool of unlabeled data on a tight budget. In contrast to Hawk, but consistent with classic active learning [45], this work assumes that: (a) cost is related only to the total number of items labeled, but not their transmission at low bandwidth; and, (b) asking for fewer labels is always better, even if it takes longer to assemble the training set.

Ad hoc solutions to this problem have also evolved. For example, Intel's CVAT tool [1] and related efforts [42] combined with crowd-sourcing can speed up training set assembly. However, such approaches do not address the problem of extreme low bandwidth to data. Further, crowd sourcing may often be infeasible due to privacy compliance, national security, domain expertise or other reasons.

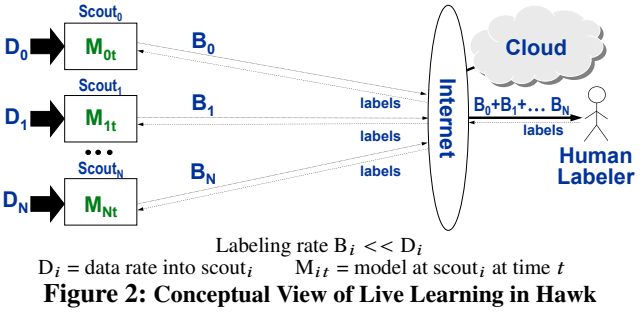


Figure 2: Conceptual View of Live Learning in Hawk

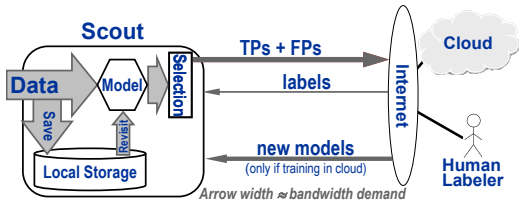


Figure 3: End-to-End Pipeline of One Scout

3 Hawk System Overview

Live Learning in Hawk is based on a simple insight. *Even a weak model, learned on-the-fly, is far more effective than random sampling in pruning away irrelevant data.* An intuitive analogy is the compounding of interest in financial settings. Continuous compounding can result in significant total returns over a modest length of time in spite of low interest rates. Similarly, continuous learning can accumulate many TPs over the course of a mission, even with a weak initial model, low base rate, and modest labeling rate. Hawk implements Live Learning by seamlessly pipelining semi-supervised learning, active learning, and transfer learning with asynchronous bandwidth-sensitive data transmission and human labeling.

Figure 2 shows a high-level view of Live Learning. Fresh data arrives continuously into each scout at a high rate, D_i. Only a tiny fraction of this incoming data can be transmitted for labeling, at rate B_i, because of the severely-limited backhaul bandwidth. At the start of a mission, only a few TPs of the desired class are available. These are used to train a weak ML model via transfer learning from a base model. This weak model is deployed at all scouts, and used to inference incoming data. Promising results are streamed to the user for labeling as they are generated (§5.1), and labels are sent back to scouts. Since the model is weak, both precision and recall are poor. However, as data is labeled, the training set grows. When sufficient new TPs have been discovered, a new model is trained. The training can be configured to occur at each scout, or in the cloud (§5.2). This improved model replaces the older model, and the process continues. Live Learning terminates when the desired number of TPs for the final training set have been collected. This newly-assembled training set

can be used to train a state-of-the-art model in the cloud, and also archived for training future models.

Concern for poor backhaul bandwidth pervades every aspect of Hawk. Both its absolute value and its extreme mismatch relative to the incoming data rate are problematic. Figure 3 shows a zoomed-in view of the pipeline at one scout. 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. Inferencing is done in small batches for two reasons. First, efficient use of GPUs typically requires batching. Second, the criteria for selective transmission (§5.1) are only meaningful on batches. After inferencing, a small subset of the tiles is selected for transmission and labeling (§5.1). A tunable data sourcing policy (§5.3) 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. Hawk assumes perfect labeling. Coping with imperfect labeling is an important problem that has been studied by others [9]. Labels are transmitted back to the scout as they are generated. The uplink bandwidth demand for this is small.

Hawk can choose between training in the cloud or on the scout (§5.2). 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. When training in the cloud, negatives are obtained from cloud archives.

In classic ML settings, only the model trained on a fully-assembled training set is of interest. However, some time-sensitive Hawk settings can benefit from "best so far" models. For example, if a new type of military threat emerges, survivability may be lowered until this threat 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 model that is based only on TPs discovered so far. A similar situation arises with programmable camouflage [4, 16] — the visual appearance of an object may morph into new forms that have not been seen before. In recognition of these time-sensitive use cases, our evaluation of Hawk (§7) presents the full timeline of TP discovery during a mission.

4 SSL and Active Learning in Hawk

Hawk makes three changes to classic pseudo-labeling. Asymmetric handling of negatives and positives is the essence of these changes. First, no data item is added to a training set as a positive unless confirmed by human labeling. Second, most data that is pseudo-labeled as negative is unlikely to be transmitted. Third, a fresh random subset of available pseudo-labeled negatives is used in the training set of each model.

The first rule is motivated by extreme class imbalance. When there are very few TPs, even a single erroneously pseudo-labeled positive can have devastating impact in training. It is therefore worth spending limited bandwidth to ensure that every single pseudo-labeled positive is verified as a TP.

The second rule is motivated by low bandwidth, and the desire to use this scarce resource for high value. Relative to the current model, a data item may be a positive ($\mathcal{P}$), a hard negative ($\mathcal{HN}$) or an easy negative ($\mathcal{EN}$). An $\mathcal{EN}$ is far from the decision boundary of the current model, while an $\mathcal{HN}$ is much closer. As learning proceeds, an item that used to be considered an $\mathcal{HN}$ may become a $\mathcal{P}$ or an $\mathcal{EN}$ because the decision boundary has shifted. Hawk prioritizes the transmission of $\mathcal{HN}$s over $\mathcal{EN}$s because their chances of misclassification are higher, and therefore obtaining a human label is more valuable. Hawk transmits $\mathcal{EN}$s only when bandwidth is underutilized because of too few $\mathcal{P}$s and $\mathcal{HN}$s.

The third rule is motivated by the danger of confirmation bias in SSL. A TP that is mislabeled as an $\mathcal{EN}$ in an early training set can wreak havoc on the entire evolution of models. There is no way to totally avoid this danger — it is inherent to pseudo-labeling. However, the risk can be reduced by choosing a random selection of $\mathcal{EN}$s afresh for the training of each new model. Because of the very small number of TPs and the extreme class imbalance, the chances that a misclassified TP will be selected again from the huge volume of $\mathcal{EN}$s is low. A specific model may be affected, but it is unlikely that this will harm the evolution of models over a longer term. $\mathcal{HN}$s pose less of a danger because they are prioritized over $\mathcal{EN}$s in selective transmission for human labeling, and are therefore likely to eventually receive a human label. The use of $\mathcal{EN}$s in training serves two purposes. First, it reduces the tendency for model drift to occur in training. Second, it tunes the model better to the specific attributes of the incoming data.

Formally, let $x \in \mathbb{R}^d$ be an unlabeled data item having d dimensions and let $y \in \{0,1\}^C$ is the corresponding one-hot label vector for C classes, then we aim to learn a model, $M(.;\theta_G) : \mathbb{R}^d \rightarrow \{0,1\}^C$, where θ_G are the parameters of the model M of generation Gen-G and $M(x;\theta_G) = [p_c]_{c=0}^{C}$ are the softmax probabilities ($p_c \in [0,1]$) produced by the model.

The data items, D_G are the unlabeled data processed on the scouts after the deployment of model Gen-G, but before the training of a new generation $G+1$. The items in D_G, can either belong to the human verified set D_{hG} or the unlabeled $\mathcal{EN}$s set D_{uG}. The number of items in the set D_{hG} (or D_uG) is given by N_{hG} (or N_{uG}). We can express D_{hG} and D_{uG} as:

$$D_{hG} = \{(x_i, y_i)\}_{i=0}^{N_{hG}}, \qquad D_{uG} = \{x_i\}_{i=0}^{N_{uG}},$$

where x_i is a data item and y_i is its one-hot label vector.

Suppose the ground-truth labels of unlabeled set D_{uG} were known, then we can denote the number of data items of class c as N_c, i.e., $\sum_{c=0}^{C} N_c = N_{uG}$. The degree of imbalance in

the unlabeled dataset may be given as $\gamma = \frac{N_{uG}}{min\ N_c}$. Under conditions of high imbalance ($\gamma \gg 1$), Kim et al [26] show that naively applying pseudo-labels can be detrimental to model training. Hawk has to deal with extreme imbalanced binary classification scenarios, where $\gamma \geq 1000$ and C is 2. Respecting Kim et al's guidance, we only pseudo-label a random subset of low-scoring $\mathcal{ENS}$, as given in Algorithm 1. We first select a fraction δ of low-scoring unlabeled data items in D_G (where the probability of x_i belonging to the target class (class 1) is small, i.e., $y_{1i} < 0.5$). As Li et al [29] and Mahajan et al [34] recommend, we then balance the number of items in positive and negative classes, such that $N_0 + N_{h0} = \beta N_{h1}$. The sampling ratio β is a hyperparameter, set to 50 in our experiments. The pseudo-labeled negatives are then added to the training set having human labeled items. We train and optimize the model parameters using cross entropy loss. In the presence of class imbalance, $\mathcal{ENS}$ being more frequent in the training set can cause instability during training. We use a weighted variant of Cross Entropy loss [13, 30] (specifically, a class-balanced variant of Focal Loss [30]) to mitigate this:

$$L(p) = \begin{cases} -\frac{1}{\beta}(1-p)^2 \log(p) & \text{if } c = 1 \\ -(1-\frac{1}{\beta})p^2 \log(1-p) & \text{otherwise,} \end{cases}$$

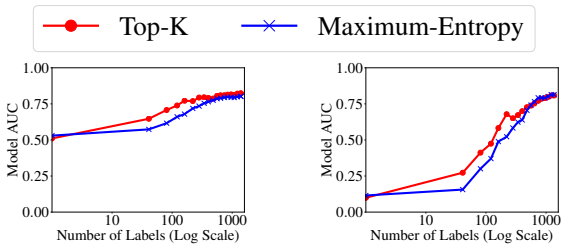
where β is the sampling ratio and p is the probability estimate given by the model.

Algorithm 1 Pseudo-Labeling Negatives in Hawk

Input: Model M_G, Data stream D_G

1: **for** x_i `in` D_G **do**
2: $[p_{ci}]_{c=0}^{2} \leftarrow M(x_i)$
3: **end for**
4: $\{x_0, \ldots, x_{|D_G|}\} \leftarrow$ **Sort**$(p_{1i}), \forall i \in |D_G|$
5: $D_{u'} = \{x_0, \ldots, x_M\}$, where $M < \delta |D_G|$
▷ Set of low scoring items
6: $D_{\tilde{u}} = \{x'_0, \ldots, x'_{N_0}\} \leftarrow$ **RandomSelection**$(D_{u'}, N_0)$
7: $\tilde{y}_{0i} = 1$ for i $\in D_{\tilde{u}}$
▷ Pseudo-labeling data as negatives

Hawk uses a Top-K selective transmission protocol to implement active learning. This runs counter to conventional wisdom in active learning, which advocates a maximum entropy (MaxEnt) strategy (i.e., favoring scores near 0.5). Our experiments (§5.1) show that Top-K is a better choice for Hawk. On each scout, data items that are inferenced by the current model are sorted by score. The highest-scoring items are likely to be $\mathcal{P}$s. Below them are likely to be $\mathcal{HNS}$. Further below are the $\mathcal{ENS}$. The value of "K" in Top-K is chosen to fill (but not overfill) the end-to-end pipeline. When bandwidth is scarce, very few $\mathcal{HNS}$ and $\mathcal{ENS}$ are transmitted. When bandwidth is plentiful, there is less need to be frugal. At all bandwidths, TPs are precious and the protocol aims to find and confirm every one of them via human labeling.



(a) YFCC-derived dataset (b) DVIDS-derived dataset

Symbols on curves indicate model switchover. Observed maximum standard deviation was 0.16 across three runs of each experiment.

Figure 4: Top-K vs. Maximum Entropy Selective Transmission

5 Key Design Decisions

5.1 Selective Transmission

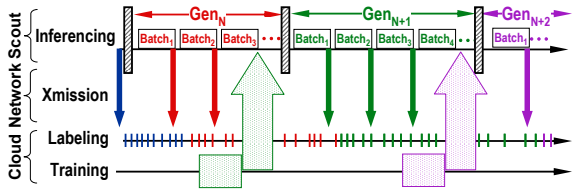
To study active learning alternatives in a Hawk setting we use the *YFCC-derived task* and the *DVIDS-derived task* that have been previously described [19]. These tasks are representative of Hawk missions. However, there is no danger of overfitting because their datasets are totally unrelated to the datasets used in this paper for experimental validation (§6.1).

Figures 4(a) and 4(b) compare Top-K versus MaxEnt for these tasks, as the number of labeled examples increases. Note that the X-axis in these graphs is in log scale. The Y axis is the area under the precision-recall curve (AUC). Both MaxEnt and Top-K converge to the same AUC asymptotically. However, Top-K improves much faster early in the evolution. Since the improved precision means that less bandwidth will be wasted on FPs early in the mission, Hawk uses Top-K. At high bandwidth, K is large to improve the chances of labeling all TPs. As bandwidth drops, K shrinks.

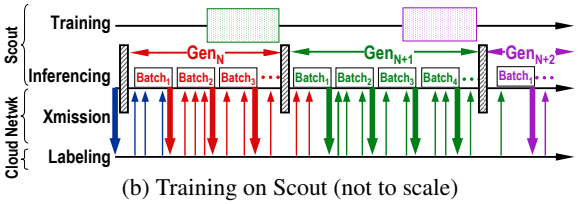
5.2 Training Location

Training in the cloud benefits from the unlimited hardware resources available there, and is standard ML practice. In the Hawk setting, all the TPs are available at the cloud for construction of a training set, since they are transmitted for labeling (Figure 3). The negatives for the training set can be assembled from all FPs that were transmitted, augmented by negatives from cloud archives. By definition, since the phenomenon being studied is new and rare, there are few TPs in the archives — almost everything there is negative. Figure 5(a) shows the timeline of events for cloud training.

Unfortunately, after training, the new model has to be transmitted from the cloud to the scout. Accurate models tend to be large. For example, in our experiments, the ResNet-50 model has a size of 95 MB and the YOLOv5-small model has a size of 14 MB. At 12 kbps, transmitting these models would take over 17 hours and over 2.5 hours respectively. Only with aggressive model compression can cloud-based training be practical for Hawk. We explore this in a later section (§5.4).



(a) Training in the Cloud (not to scale)



(b) Training on Scout (not to scale)

Time flows horizontally in these figures. If multiple scouts are participating, their transmissions are merged into a single stream for labeling.

Figure 5: Timelines at Scout and Cloud

Training on the scout is severely limited by the hardware resources available, and by the need to share these limited resources with concurrent inferencing of incoming data. However, it has the benefit of avoiding transmission of large models over an extremely low-bandwidth network. The lower the bandwidth, the more valuable this benefit. Hence, Hawk uses an adaptive training strategy. When bandwidth is high, training in the cloud is an obvious win. At very low bandwidth, however, training at the scout is the better approach. If an early decision is made to train on the scout, a further bandwidth optimization is possible. Only thumbnails are needed for labeling, while full-resolution data is needed for cloud-based training. This can reduce bandwidth demand by a small constant factor (roughly 4 in our experiments).

Figure 5(b) shows the timeline of events in the Hawk pipeline when training is done at the scout. Even though training is now much slower than cloud-based training, it is more than compensated by avoiding model transmission. The cross-over point is dependent on the model compression achievable. Our experiments (§7.5) explore these tradeoffs.

In both timelines, a sequence of improving models (Gen_{N-1}, Gen_N, Gen_{N+1}, ...) inferences data at the scout. A small, high-scoring subset of the data is then selected (§5.1) for transmission and labeling. Training of a new model is triggered when the number of TPs has increased by a certain (tunable) percentage. The new model replaces the old one as soon as training is completed. These details are completely invisible to the human labeler. She is only aware that, over time, more of the data sent to her for labeling are positives and fewer are trivial negatives. Due to transmission and think time delays, labeling is shifted to the right relative to inferencing.

5.3 Re-visiting Discards

A scout wastes little bandwidth on $\mathcal{EN}$s. Rigidly adhering to this principle would mean that a $\mathcal{P}$ that is misclassified as an $\mathcal{EN}$ may be lost forever. This would be especially unfortunate because of the low base rate — every positive is precious. The obvious solution of lowering the selectivity of transmission would overwhelm the limited bandwidth with FPs.

Re-visit Approach	Images Processed	TPs Found	Percentage of Ground-Truth Positives
YFCC-Derived Task (700 ground-truth positives)			
None	700,000	616 (11)	88%
Top 100	715,300	629 (5)	89%
All	5,460,460	630 (1)	90%
DVIDS-Derived Task (680 ground-truth positives)			
None	680,000	523 (13)	73%
Top 100	693,600	552 (3)	81%
All	4,751,840	575 (1)	85%

Figures in parentheses are standard deviations across three runs.

Table 1: Value of Re-visiting Discard Pile

We therefore keep selectivity high, but re-visit items that were discarded earlier. The better recall of an improved model may help to find a "lost" $\mathcal{P}$. For the two calibration tasks described earlier (§5.1), Table 1 shows the total number of positives discovered using three approaches: (a) no re-visit; (b) re-visit only the 100 top-scoring results in the discard pile of each previous batch; and (c) re-visit all previous discards.

Three observations follow from these results. First, there is indeed value in re-visiting the discard pile. For example, the bottom row for the YFCC-derived task in Table 1 shows that 14 additional positives were found, an increase of about 2.3%. Second, there is room for improvement: the last column shows that nearly 10% of the positives were missed. Third, just re-visiting the top 100 items in the discard piles of each batch is almost as good as re-visiting all discarded data. For a modest increase in the number of images processed (barely 2%), almost the full benefit is obtained (629 out of 630 positives). The results for the DVIDS-task in Table 1 confirm these observations. Based on these results, Hawk uses the policy of just re-visiting the 100 top-scoring items in discard piles. In effect, scores from an old model act as a result cache to guide the selection of items to rescore with a new model.

5.4 Model Compression

A key obstacle to training in the cloud is the size of the trained model that has to be shipped back to the scout. Classic compression and deduplication techniques (e.g., `bzip2` and `rsync`) only save 5-8 %. A more promising approach is model compression via DeepIoT [51], which reports size reductions of 90–99% on sequential model architectures such as LeNet and VGG. Unfortunately, DeepIoT is less effective in state-of-the-art DNNs such as ResNet-50 and YOLO. In contrast to older DNNs where each layer feeds into the next layer, more recent DNNs have residual blocks with skip connections, where each layer feeds into the next layer and also into layers 2–3 hops away. Our measurements (Table 2) only show

Model	Input (A) (MB)	Output (B) (MB)	Compression Ratio (A:B)
ResNet-50	97.69	38.61	2.53
YOLOv5-small	27.60	11.95	2.31
EfficientNet-B4	74.26	27.71	2.68

Table 2: Model Compression: DeepIoT

Model	Original Size	Freeze all but last		
		3 layers	2 layers	1 layer
ResNet-50	97.69	41.86	24.84	7.81
YOLOv5-small	27.60	8.77	7.64	5.39
EfficientNet-B4	74.26	21.55	9.92	6.84

All sizes above are in MB

Table 3: Model Compression: Freezing Layers

factors of two to three reduction in size. While substantial, this is not adequate for extreme low bandwidths. A different approach to model compression is to freeze most layers of a model, and only retrain the last few layers. However, as Table 3 shows, this only achieves modest compression. The smallest amount of data to be transmitted is ~5 MB — nearly one hour at 12 kbps. Anticipating future ML breakthroughs, our experiments (§7.5) explore the impact of 100X and 10X model compression if they were somehow attainable.

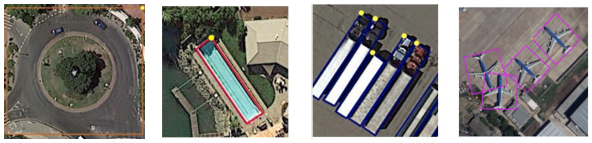
5.5 Prototype Implementation

Hawk is implemented open-source in Linux using C, Python, and PyTorch. The entire implementation is in user space, and Docker is used to encapsulate its ML components. Hawk works with diverse DNNs, and currently supports ResNet-50 [20], YOLOv5 [25], and EfficientNet-B4 [47]. Extending Hawk to support future DNNs is straightforward. For supported DNNs, Hawk provides a wizard that guides optimal hyperparameter settings. Hawk leverages the widely-supported ZeroMQ [2] mechanism for high latency, low-bandwidth networks. For additional details see [17, 18].

6 Evaluating Hawk

Our evaluation of Hawk answers four questions.

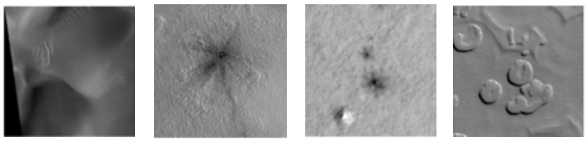
- *In spite of extreme low bandwidth, can a scout discover many of the TPs that it encounters during a mission?* Because of extreme class imbalance, the dominant influence on model accuracy for a given DNN architecture is the number of TPs used in training. Hence, the ratio of TPs discovered to the number of TPs in ground-truth is the most important system metric.
- *How close is Hawk to an ideal system?* We use two different definitions of "ideal." The first is an oracle that embodies ground truth. Every single TP is discovered and immediately queued for transmission, but no other data is transmitted. The second variant of "ideal" is called "BruteForce." It is a model with the same architecture as the Hawk DNN, but trained in advance on fully labeled incoming data. By definition, such a model is grossly overfitted to the data that will be seen during the mission. It requires all incoming data to be seen in advance, and transmitted to the cloud for labeling and training. It is not as good as the oracle, because it may not have perfect precision and recall. These two idealized baselines are the best available today for evaluating Hawk because no previous system has attempted to address the problem that it solves.
- *Can Hawk use additional bandwidth effectively?* To answer this question, we increase bandwidth to 30 and 100 kbps from its baseline value of 12 kbps, and measure the increase in number of TPs discovered.
- *Is Hawk DNN-agnostic?* To answer this question, we present results from experiments that use three different DNN architectures: ResNet50, YOLO-v5, and EfficientNet-B4.



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

Each of the above examples is a small part of a 256x256 tile from a large high-res DOTA image. Ground truth bounding boxes are also shown. There are 252,231 unlabeled tiles in the mission dataset.

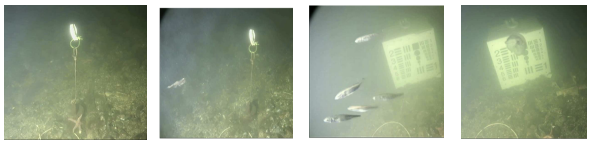
Figure 6: Examples of DOTA Target Classes



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

Each of the above examples is a 227x227 pixels in size. There are 73,031 unlabeled tiles in the mission dataset.

Figure 7: Examples of HiRISE Target Classes



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

Each of the above examples is 227x227 pixels in size. There are 563,829 unlabeled tiles in the mission dataset.

Figure 8: Examples of Brackish Target Classes

6.1 Validation Datasets

We validate Hawk on three very demanding publicly-available datasets from the domains of drone surveillance, planetary exploration, and underwater sensing. Each of these datasets was released within the past few years, and has been used in recent ML research publications in its domain.

Aerial Drone Surveillance (DOTA): The drone-captured 15-class DOTA v1.0 dataset [50] consists of 2806 fully-labeled images, half of which are used in the mission data. Image resolution ranges from 800x800 pixels to 4000x4000 pixels, with a bias towards higher resolution. All images are broken into tiles of size 256x256 pixels, yielding a mission dataset of 252,231 usable tiles. Our experimental results have been confirmed for all 15 classes. However, for space reasons we only show detailed results for the four representative classes shown in Figure 6. During a mission, all the tiles from a single image in the mission dataset are randomly delivered to one of 7 scouts at an average rate of one every 20 seconds.

Planetary Exploration (HiRISE): Images collected by the High Resolution Imaging Experiment of the Mars Reconnaissance Orbiter form the basis of the HiRISE dataset [14]. There are 73,031 labeled images, each cropped to 227x227 pixels. From the 8 target classes we have created a derived dataset by reducing the number of images of each class to achieve a 0.1% target base rate. We have confirmed our results on all classes, but show only those for four classes (Figure 7). During a mission, 100 tiles in 20 seconds are delivered at random to one of 7 scouts.

Underwater Sensing (Brackish): The Brackish dataset [43] is a publicly available dataset containing labeled images of marine animals in a brackish strait with varying visibility. There are 14,518 images of 1080p resolution. We split each image into tiles of size 256x256 pixels, with overlap of 50 pixels and remove tiles in which the size of the target instance is less than 20x20 pixels. The resulting derived dataset has 563,829 tiles. There are 6 target classes, all with base rates below 0.1%, and we have confirmed our results for all of them. To save space, we present results only for the four classes shown in Figure 8. During a mission, 100 tiles in 3 seconds are delivered at random to one of 7 scouts.

6.2 Hardware

Our evaluations use 7 physical machines as scouts, each with 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. We use the FireQoS traffic shaping tool [49] to emulate the following bandwidth and RTT on each scout's backhaul: 12 Kbps @4s, 30 Kbps @2s, and 100 Kbps @2s. Inter-scout connectivity (to enable sharing of TPs) is 1 Mbps @200ms. We emulate the human labeler by code that returns the ground-truth label for its input.

6.3 Choice of K for Selective Transmission

As discussed earlier (§5.1), we use a policy of only transmitting the K top-scoring items from inferencing a batch of tiles.

Gen	New model installed (seconds)	TPs Discovered	TPs Encountered	Tiles Xmitted	Tiles processed on scout
0	0	0	0	0	0
1	277 (54)	8 (1)	19 (3)	47 (16)	17759 (713)
2	530 (59)	19 (1)	39 (4)	103 (16)	32797 (556)
3	768 (55)	28 (3)	59 (4)	152 (17)	42859 (624)
4	1168 (115)	50 (2)	95 (5)	241 (34)	65116 (703)
5	1579 (63)	72 (2)	126 (8)	335 (12)	94186 (433)
6	2201 (57)	97 (5)	165 (5)	455 (15)	124783 (542)
7	3010 (44)	137 (3)	218 (5)	604 (14)	166576 (366)
8	3529 (97)	187 (5)	283 (8)	772 (31)	207074 (676)
	4149 (32)	219 (6)	336 (0)	923 (3)	252231 (0)

Number of bootstrapping TPs = 20 DOTA Class: Roundabout
Figures in parentheses are standard deviations from three runs.

Table 4: Model Evolution (12 kbps, DOTA, Scout Training)

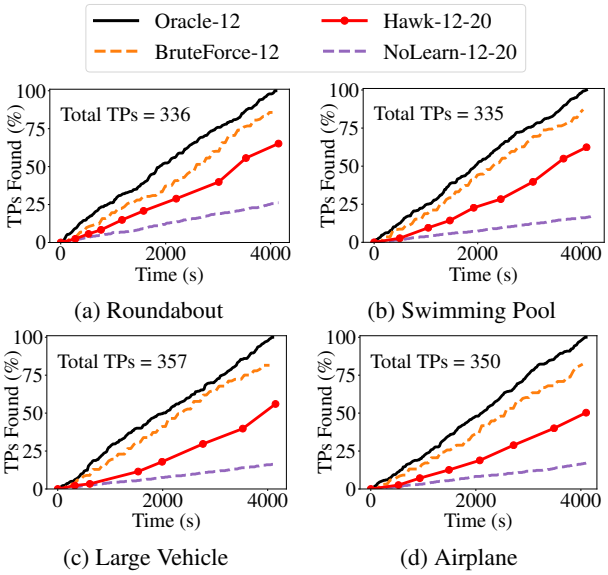
The goal is to fill, but not overfill, the end-to-end pipeline (Figure 3) from incoming data to cloud. The bottleneck of this pipeline is the smallest of three values: (a) the ingest rate at a scout; (b) the inferencing rate on a scout; and, (c) the transmission rate at current bandwidth of an average-sized tile. To allow for some spare bandwidth, we set K to be significantly lower than the bottleneck value: K=4 at 12 kbps; K=10 at 30 kbps; K=30 at 100 kbps.

6.4 Experimental Procedure

All our experiments begin with a model pre-trained on ImageNet and then trained via transfer learning on a certain number of bootstrapping TPs of the target class. This crude model is referred to as "Gen-0." The unlabeled mission dataset is striped across scouts at the specified input rate for the dataset, and the experiment continues until the entire mission data has been delivered. This takes roughly an hour to an hour and a half, depending on the dataset. We conduct three runs of each experiment, using different seeds for random number generation. The results use the following notation:

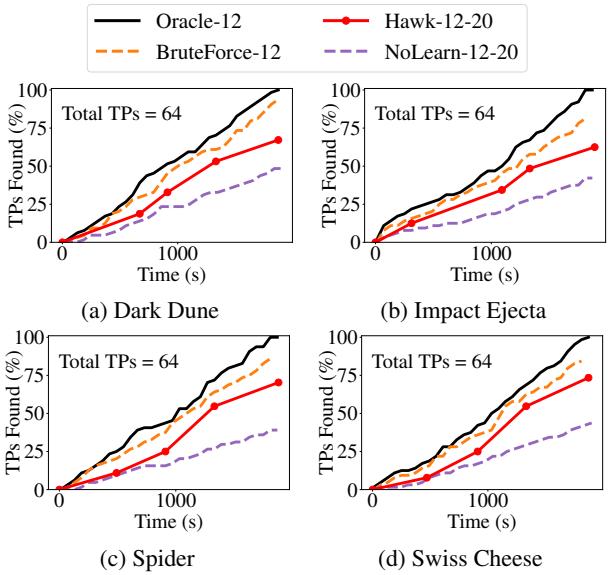
- *Hawk-xx-yy:* Hawk at a bandwidth of *xx* kbps, starting from a Gen-0 that was bootstrapped from *yy* TPs.
- *NoLearn-xx-yy:* same as *Hawk-xx-yy*, except that no learning is done to improve Gen-0.
- *BruteForce-xx:* at *xx* kbps, a DNN that uses the same architecture as Hawk but is trained in the cloud using unlimited resources and a fully-labeled version of the same dataset used in the mission.
- *Oracle-xx:* an oracle embodying ground truth at *xx* kbps.

The ML task in most of our experiments is object classification using ResNet-50. The sole exception is in Section 7.7, where model-agnostic Live Learning is demonstrated on an object detection task using YOLO and EfficientNet.



(a) Roundabout (b) Swimming Pool (c) Large Vehicle (d) Airplane

Figure 9: Model Evolution (12 kbps, DOTA, Scout Training)



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

Figure 10: Model Evolution (12 kbps, HiRISE, Scout Training)

7 Results

7.1 Bandwidth-Frugal Model Evolution

With a backhaul bandwidth of 12 kbps and training done on the scout, Table 4 shows the evolution of models for the class "Roundabout" in the DOTA dataset. This corresponds to the timeline shown earlier in Figure 5(b). Column 2 shows that Gen-0 is replaced by Gen-1 at 277 seconds into the mission. Since the model for each scout is trained independently on that scout, the exact moment of replacement may vary slightly across scouts. That variance, plus the use of a different random seed for each run, leads to the standard deviation shown (54 seconds). Gen-1 is replaced by Gen-2 at 530 seconds, and so on. The mission ends when mission data is exhausted.

By the time Gen-1 replaces Gen-0, Hawk scouts have discovered 8 TPs (Column 3) out of the 19 that they have encountered so far (Column 4). The 10 TPs they have missed is indicative of the poor recall of Gen-0. The last two columns show how frugal Hawk is in bandwidth usage. Across all scouts, only 47 tiles (Column 5) have been transmitted for labeling, out of 17,759 encountered so far (Column 6). These general observations continue throughout the mission. By the end, 219 TPs out of 336 are discovered, with the scouts only transmitting 923 tiles out of 252,231 (less than 0.4%).

In a more compact format, the curve for Hawk-12-20 (red) in Figure 9(a) summarizes the data in Table 4. The symbols correspond to models Gen-1, Gen-2, etc. in Table 4. The X axis value of a symbol gives the time into the mission when that model was installed (Column 2 of Table 4), while its Y axis value gives the pecentage of TPs discovered by that point in the mission (Column 3 of Table 4). The curve for Oracle (black) in Figure 9(a) gives the ground truth: i.e., the percentage of TPs encountered by the 7 scouts until that point in time (i.e., Column 4). Relative to Table 4, Figure 9(a) omits (a) the observed variance in the results, (b) tiles transmitted for labeling (Column 5), and (c) total tiles encountered so far (Column 6). To save space, we will present most results in this format, rather than the tabular form of Table 4.

Figure 9(b), (c) and (d) confirm similar model evolution for other DOTA classes. Figures 10 and 11 confirm that it also holds for the HiRISE and Brackish data sets. Across all datasets and classes, Hawk is typically able to find 50-60% of the TPs encountered. Its lowest success rate is 49% (class Starfish in dataset Brackish) and its highest is 73% (class Swiss Cheese in dataset HiRISE). Since these results were all obtained at 12 kbps, it is clear that Hawk is very bandwidth-frugal. Our results give a strongly positive answer to the question *"In spite of extreme low bandwidth, can a scout discover many of the TPs that it encounters during a mission?"*

7.2 Value of Self-Improvement

The results in Figures 9, 10, and 11 also illustrate the importance of learning during a mission. In each of these graphs, the curve labeled "NoLearn-12-20" shows the number of TPs that would have been discovered if Hawk was not self-improving. The gap between the red and purple lines shows the importance of learning. In almost all cases, the number of TPs discovered would have been 50% or fewer than Hawk is actually able to discover. This justifies the need for Hawk. It also justifies the weight, size, power, and cost overhead of equipping scouts with hardware that is capable of doing training. If inferencing using a static model was all that was needed, much weaker scout hardware would have sufficed. Training in the cloud allows use of weaker scout hardware,

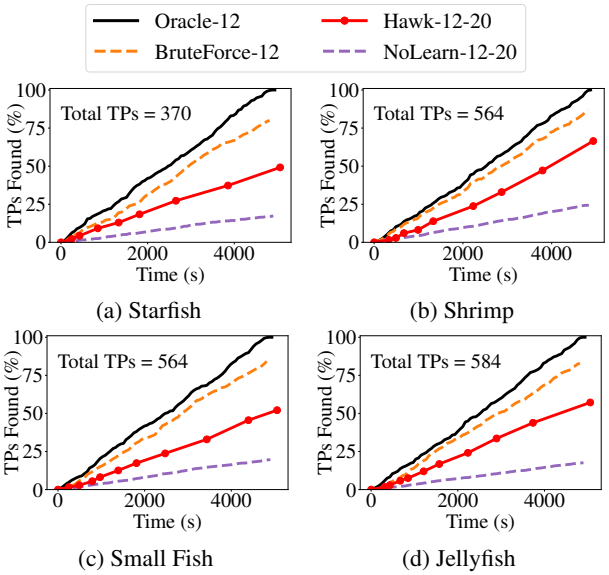


Figure 11: Model Evolution (12 kbps, Brackish, Scout Training)

but Section 7.5 shows that it is much less bandwidth-frugal than training on the scout. Self-improvement is worth it!

7.3 Comparison with Brute Force Approach

Figures 9, 10, and 11 also compare Hawk with a "Brute Force" model. As explained earlier (§6.4), such a model cannot be implemented in practice because it assumes that (a) all mission data can be previewed in advance of mission start, and (b) there is sufficient bandwidth to transmit all mission data to the cloud for labeling and training. Yet, since it is a model that uses the same DNN architecture as Hawk, a comparison offers insights into how much improvement is possible in Hawk without improvement to the underlying DNN.

The gap between the curves labeled "Hawk-12-20" and "BruteForce-12-20" in Figures 9–11 shows the difference between the number of TPs found by the two approaches. By mission end, the gap is roughly 15-40% across all datasets and classes. The smallest gap is 13% for class Swiss Cheese of dataset HiRISE, and the largest gap is 39% for class Airplane of dataset DOTA. This gap can be attributed to the difference between fully supervised learning and Live Learning in Hawk. We show how to bridge this gap later in the paper (§7.8).

7.4 Ability to Use Additional Bandwidth

A scout can periodically estimate bandwidth during a mission, and use it to modify K as discussed earlier (§6.3). For four classes of DOTA, Table 5 shows the impact of increasing backhaul bandwidth from 12 kbps to 30 kbps, and then to 100 kbps. In all cases, increased bandwidth enables more TPs to be discovered during the mission. For class Roundabout, for example, 4.7% additional TPs are discovered at 30 kbps (69.9% rather than 65.2%, out of 336). Because bandwidth

		12 kbps		30 kbps		100 kbps	
	Total TPs	TPs Discovered	Tiles Xmitted	TPs Discovered	Tiles Xmitted	TPs Discovered	Tiles Xmitted
Roundabout	336	65.2%	923	69.9%	2451	73.2%	7501
Swim. Pool	335	62.4%	924	74.6%	2339	81.2%	7479
Lg. Vehicle	357	56.0%	912	60.1%	2280	74.0%	7350
Airplane	350	50.3%	924	60.9%	2410	72.3%	7530

Table 5: Using Additional Bandwidth (DOTA, Scout Training)

is more plentiful, more tiles can be transmitted for labeling (2451 rather than 923 out of 252,231, less than 1.0%). At 100 kbps, a further 3.3% TPs are discovered by the end of the mission (73.2% rather than 69.9%), with 7501 tiles transmitted out of a total of 252,231 (less than 3.0%). The results for HiRISE and Brackish are consistent with Table 5, but omitted for space reasons. These results confirm that Hawk is able to effectively use additional bandwidth effectively.

At high enough bandwidths, the network may no longer be the bottleneck in the Hawk pipeline — it will shift to the human labeler. To address this, it will be necessary to support multiple human labelers working concurrently on a labeling work queue. Hawk will support this capability in the future.

7.5 Cloud versus Scout Training

As discussed earlier (§5.2), Hawk can be configured to train on the scout or in the cloud. The major disadvantage of cloud training is the need to transmit the resulting large model over low bandwidth. We have shown earlier in this paper (§5.4) that model compression is not very effective on the kinds of DNNs typically used in Hawk. The achievable compression of 2–3 is helpful, but hardly adequate at 12 kbps for transmitting models that are tens of MB in size in a timely manner.

Our experiments examine what the impact on Hawk would be if 100X and 10X model compression were somehow achievable. The results overstate the benefits of cloud-based training since they ignore the time required for model compression, and any associated loss of model accuracy. Yet, this deliberate handicapping of scout-based training is useful as an upper bound on the benefit of cloud-based training in Hawk.

Table 6 presents the results of cloud-based training followed by 100X model compression for a mission that is

Gen	New model installed (seconds)	TPs Discovered	TPs Encountered	Tiles Xmitted	Tiles processed on scout
0	0	0	0	0	0
1	973 (94)	10 (3)	22 (11)	54 (30)	15780 (6947)
2	1771 (108)	32 (7)	76 (5)	298 (84)	64574 (3698)
3	2531 (164)	66 (4)	124 (34)	450 (44)	93972 (9455)
4	3298 (156)	102 (13)	201 (13)	649 (47)	152379 (7041)
	4010 (15)	151 (14)	336 (0)	1009 (6)	252231 (0)

Number of bootstrapping TPs = 20 DOTA Class: Roundabout
Format identical to Table 4

Table 6: Model Evolution (12 kbps, DOTA, Cloud-100X Training)

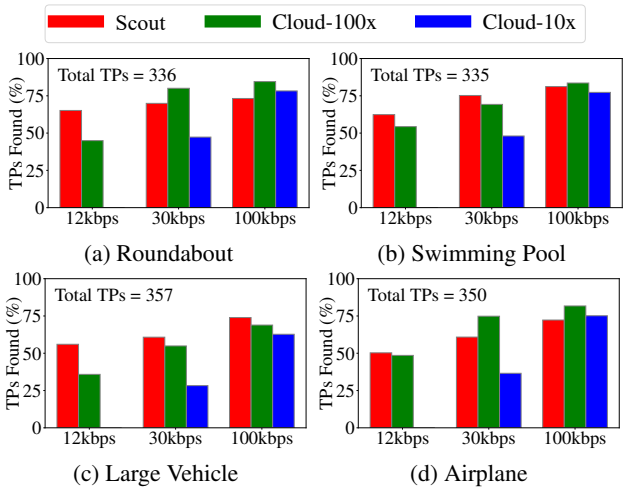


Figure 12: Cloud vs. Scout Training (DOTA)

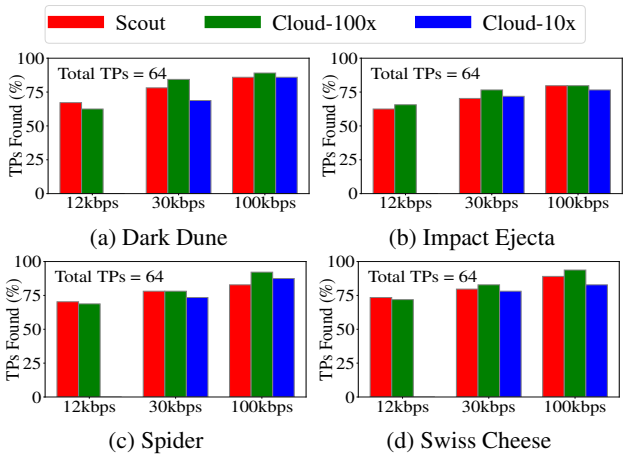


Figure 13: Cloud vs. Scout Training (HiRISE)

identical in all other respects to the mission presented earlier in Table 4. The bandwidth (12 kbps), dataset (DOTA) and class (Roundabout) are unchanged across the two experiments — only the training site is different. With cloud training, 0.95 MB has to be transmitted at 12 kbps even after 100X compression. Since this consumes over 10 minutes, fewer models are installed during a mission — only 4 models in Table 6, in contrast to 8 models in Table 4. On the plus side, inferencing is faster because scout hardware does not have to be used for training. Processing of mission data hence completes roughly 100 seconds sooner in Table 6 than in Table 4. Unfortunately, the net impact of these factors is not favorable for cloud training. Out of 336 TPs, Hawk only finds 151 in contrast to 219 (bottom rows of Tables 6 and 4).

When bandwidth rises, cloud training at 100X wins over scout training. Figure 12(a) summarizes the results for class Roundabout in dataset DOTA. Out of a total of 336 TPs, 80.1% are found at 30 kbps (versus 69.9% with scout training); 84.5% are found at 100 kbps (versus 73.2%). With 10X

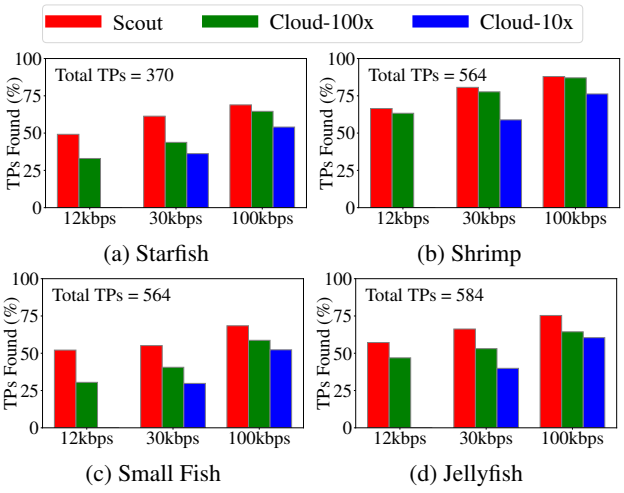


Figure 14: Cloud vs. Scout Training (Brackish)

compression, cloud training fares much worse. At 12 kbps, model transmission takes so long that the results are meaningless. Matters improve as bandwidth rises. At 30 kbps, 47.3% of the TPs are found (versus 69.9% with scout training). At 100 kbps 78.3% are found (versus 73.2%).

The results for three other classes of DOTA are shown in Figures 12(b)–(d), and the results for the HiRISE and Brackish datasets are shown in Figures 13 and 14.

Our results show that the cloud is not always the right place for training, even if privacy is not an issue. When data is unlabeled, bandwidth is very low, and the target class is rare, the reality is more complex. The optimal training site may be the cloud or the scout. Only a dynamic, bandwidth-adaptive approach is likely to be successful.

7.6 Novelty of Phenomenon

From an ML viewpoint, the size of Hawk's bootstrap set implicitly defines the novelty of the phenomenon being explored. All results reported so far were obtained with a bootstrap set size of 20. We ask *"Is Hawk effective with even smaller bootstrap sets, implying an even weaker Gen-0 model?"*

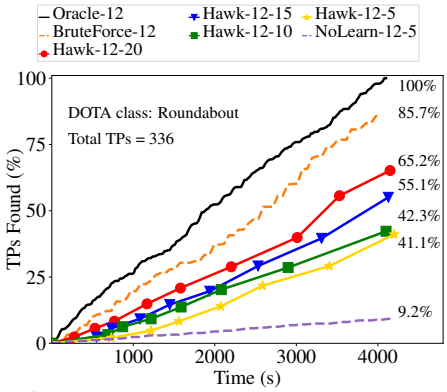


Figure 15: Impact of Bootstrap Set Size (12 kbps, DOTA)

		12 kbps		30 kbps		100 kbps	
	Total TPs	TPs Discovered	Tiles Xmitted	TPs Discovered	Tiles Xmitted	TPs Discovered	Tiles Xmitted
Roundabout	214	65.9%	570	77.1%	1427	81.8%	4280
Swim. Pool	214	69.6%	571	78.5%	1430	84.6%	4285
Lg. Vehicle	225	54.7%	546	70.2%	1367	75.6%	4100
Airplane	200	60.0%	633	71.0%	1433	84.0%	4390

Compare with Table 5 for ResNet-50.

Table 7: YOLO as Bandwidth Varies (DOTA, Scout Training)

		12 kbps		30 kbps		100 kbps	
	Total TPs	TPs Discovered	Tiles Xmitted	TPs Discovered	Tiles Xmitted	TPs Discovered	Tiles Xmitted
Roundabout	336	55.1%	948	64.3%	2570	86.0%	7860
Swim. Pool	335	60.6%	940	82.1%	2590	91.0%	7650
Lg. Vehicle	357	48.7%	916	57.7%	2470	69.2%	7620
Airplane	350	55.4%	958	77.7%	2430	84.9%	7560

Compare with Table 5 for ResNet-50.

Table 8: EfficientNet as Bandwidth Varies (DOTA, Scout Training)

Figure 15 shows how Hawk model evolution varies with bootstrap set size as it shrinks from 20 down to 15, 10 and 5. These results were obtained at 12 kbps for the Roundabout class of the DOTA dataset. The figure shows that bootstrap set size clearly impacts Hawk's effectiveness. As this size shrinks, Hawk's model evolution has a shallower slope and ends with discovery of fewer TPs. However, it is impressive that even with a bootstrap set size of 5, Hawk finds 41.1% of the TPs by the end of the mission. This is in contrast to just 9.2% TPs found by Gen-0 without learning at the same bootstrap set size. As bootstrap set size is increased to 10, 15, and 20, Hawk finds 42.3%, 55.1%, and 65.2% TPs respectively. We have confirmed that this pattern applies across all classes and datasets. We omit those graphs because of limited space. In summary, Hawk is effective even with very novel phenomena.

7.7 DNN-Agnostic Model Evolution

Hawk cleanly separates and encapsulates the DNN used for selective transmission from the rest of its machinery. Adding a new DNN to Hawk is relatively straightforward. Hawk's modularity thus positions it well to benefit from future DNN architectural improvements from the ML community.

Hawk currently supports ResNet-50 [20] and EfficientNet-B4 [47] using a tile size of 256x256. It also supports YOLOv5--small [25], using a tile size of 600x600. Mirroring Table 5, which showed the bandwidth sensitivity of ResNet-50 for dataset DOTA using scout training, Table 7 and Table 8 show the results for YOLO and EfficientNet respectively. As in the case of ResNet-50, many of the TPs that are encountered during a mission are discovered even at 12 kbps. The DNN-dependence of these results is clearly visible. Since automating the choice of DNN for a mission is difficult, Hawk defers this important decision to mission personnel.

		12 kbps		30 kbps		100 kbps	
	Total TPs	Top-K Only	+Diversity	Top-K Only	+Diversity	Top-K Only	+Diversity
Roundabout	336	65.2%	60.4%	69.9%	70.5%	73.2%	80.1%
Swim. Pool	335	62.4%	57.9%	74.6%	78.2%	81.2%	93.4%
Lg. Vehicle	357	56.0%	51.0%	60.8%	60.8%	74.0%	75.9%
Airplane	350	50.3%	52.3%	60.9%	63.1%	72.3%	81.4%

The mean of three runs of each experiments are shown. Standard deviations are not shown since they are small (less than 5.54%).

Table 9: Using Diversity Sampling (DOTA, Scout Training)

7.8 Closing the Gap to BruteForce

As Figures 9, 10, and 11 show, there is a persistent gap between Hawk and BruteForce. The gap is as small as 13% in the best case, but higher in other cases. It can be viewed as the price of doing Live Learning rather than supervised learning. Although Hawk is doing very well, can we do even better?

The confirmation bias inherent to SSL is a significant contributor to this gap. The early TPs guide learning along a path that gives low scores to TPs that look very different. These are the proverbial "black swans," relative to what has been learned so far. Once SSL proceeds down this path, the TPs transmitted for labeling will merely reinforce the existing bias. This will lead to stagnation of model quality.

To overcome confirmation bias in selective transmission, we build on recent work in *diversity sampling* [46, 54]. Similar to the approach taken by Zhdanov [54], we use clustering to select diverse samples. Unlike Zhdanov, we cannot make assumptions on the number of clusters. We therefore use a density-based clustering scheme [10], rather than K-means clustering. We project the high-dimensional DNN-extracted features to a low-dimensional space using principal component analysis (PCA) before applying the clustering algorithm.

How much of the available bandwidth should we allocate to diversity-based samples, as opposed to Top-K samples? This is a tricky question to answer, because it is highly dependent on the data, and the order in which it is encountered. Our experiments show that deviating from strict Top-K at very low bandwidth (12 kbps) hurts rather than helps. Since K=4 at 12 kbps, one diversity sample and three Top-K samples per batch is the smallest diversity allocation possible. The columns labeled "12 kbps" in Table 9 show that even using this smallest possible allocation for diversity hurts Hawk for most classes of the DOTA dataset.

When bandwidth rises, there is additional headroom for diversity. At 30 kbps, K=10; we transmit 7 Top-K samples and 3 diversity samples. At 100 kbps, K=30; we transmit 20 Top-K samples and 10 diversity samples. As seen from Table 9, there is a significant improvement in the number of TPs found at these higher bandwidths. For the target "Swimming-Pool" at 100 kbps, we discover 12.2% more positives using Top-K+Diversity as compared to a pure Top-K strategy.

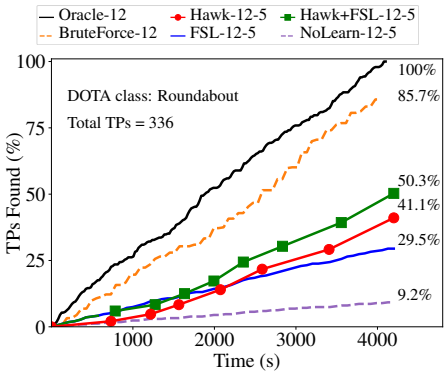


Figure 16: Hawk with Few-Shot Learning (12 kbps, DOTA)

Hawk's default configuration optimizes for extreme low bandwidths by using a pure Top-K selection strategy. However, just as K is chosen adaptively to match current bandwidth, the fraction of K allocated to diversity can also be adapted. Also, later in the mission, when many TPs have already been found, the value of diversity may be higher.

7.9 Few-Shot Learning

As mentioned earlier in the paper (§1), FSL is a complement to Hawk rather than a competitor. Its goal is to create the best model possible when only a few TPs are available. This is indeed the situation at the start of a Hawk mission. However, Hawk's goal is to go well beyond the few-shot stage by discovering new TPs and adding them to the training set. This leads to two questions: (1) Can Live Learning in Hawk help it to catch up with FSL? (2) Can the strengths of FSL and Hawk be combined? We answer these questions using SnaTCHer [24], a state-of-the-art open-set recognition algorithm for FSL. SnaTCHer supports 1-shot and 5-shot FSL; we use the 5-shot version in our experiments.

At 12 kbps and a bootstrap set size of 5, Figure 16 compares Hawk-12-5 and FSL-12-5 for class Roundabout of the DOTA dataset. For reference, the Oracle-12, BruteForce-12, and NoLearn-12-5 lines are also shown. Two observations are salient. First, FSL-12-5 does much better than NoLearn-12-5. In other words, if no learning is done beyond the bootstrap stage, using FSL to create Gen-0 is definitely better than using transfer learning. Second, although Hawk-12-5 is noticeably worse than FSL-12-5 at the start of the mission, it is noticeably better by the end. This positively answers the first question: Live Learning can indeed overcome the advantage of a superior initial model that does not evolve.

To answer the second question, we compare Hawk-12-5 with a modified version of Hawk whose Gen-0 model is FSL-12-5. Once sufficient TPs have been collected to train Gen-1, Hawk switches to using transfer learning for Gen-1 and all further models. This hybrid approach is shown by the curve Hawk+FSL-12-5 in Figure 16. From mission start to Gen-1, the superiority of FSL over simple transfer learning is operative. As mentioned earlier (§2.2), the influence of early training examples is especially significant. This leads to 9.2% more positives being found, relative to Hawk-12-5.

In effect, FSL provides Hawk with a higher starting point from which to benefit from improvements via Live Learning. Since FSL and diversity sampling (§7.8) apply to different phases of a Hawk mission, we can easily combine the two. This combination can further close the gap relative to BruteForce. We omit the results due to lack of space.

8 Closing Thoughts

Autonomous unmanned robotic missions play an increasingly important role in scientific, military, disaster recovery and other settings. In many cases, these missions involve poor network connectivity. For such missions, this paper addresses the severe mismatch between the incoming sensor data rate on scouts, and much lower backhaul bandwidth from scouts to the Internet. This mismatch ratio can easily be one to four orders of magnitude. We have described the design, implementation and evaluation of a system called Hawk that is extremely bandwidth-frugal in collecting data for training sets of rare phenomena. Using an efficient semi-supervised approach, Hawk continuously improves its selective transmission capability during a mission. Our experimental results show that even on very challenging datasets and at bandwidths as low as 12 kbps, Hawk approaches what is achievable through brute force. As ML technology advances, so will Hawk's effectiveness in bandwidth-challenged environments.

Acknowledgments

We thank the anonymous MobiCom shepherd and reviewers for their thoughtful and constructive feedback that improved the presentation of this paper. We also thank Jonathan M. Smith of the University of Pennsylvania for inspiring this work during his exemplary leadership of the DARPA DCOMP program. This research was sponsored by the Defense Advanced Research Projects Agency under award number HR001117C0051, the Lockheed Martin Corporation under award number MRA19-001-RPS006, by the United States Navy under award number N00174-23-1-0001, and by the National Science Foundation under award number CNS-2106862. 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, and the Conklin Kistler family fund. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of any sponsoring entity or the U.S. government.

References

[1] CVAT: Open Data Annotation Platform. https://www.cvat.ai/. Last accessed on July 13, 2023.

[2] Zeromq: An open-source universal messaging library. (https://zeromq.org). Last accessed: May 20, 2022.

[3] Energy Efficiency and Fuel Efficiency: Fuel Cell Technology. (https://www.energy.gov/eere/fuelcells/fuel-cells), 2015.

[4] Better camouflage is needed to hide from new electronic sensors. *The Economist* (March 29, 2023).

[5] ADAMY, D. *EW 103: Tactical Battlefield Communications Electronic Warfare*. Artech House, 2008.

[6] AHMAD, T., DHAMIJA, A. R., JAFARZADEH, M., CRUZ, S., RABINOWITZ, R., LI, C., AND BOULT, T. E. Variable Few Shot Class Incremental and Open World Learning. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)* (2022).

[7] ARAZO, E., ORTEGO, D., ALBERT, P., O'CONNOR, N. E., AND MCGUINNESS, K. Pseudo-Labeling and Confirmation Bias in Deep Semi-Supervised Learning. (https://doi.org/10.48550/arXiv.1908.02983), 2020.

[8] BENGIO, Y., LOURADOUR, J., COLLOBERT, R., AND WESTON, J. Curriculum Learning. In *Proceedings of the 26th Annual International Conference on Machine Learning* (Montreal, Canada, 2009).

[9] BOUGUELIA, M.-R., BELAID, Y., AND BELAID, A. Identifying and Mitigating Labelling Errors in Active Learning. In *Pattern Recognition: Applications and Methods* (2015).

[10] CAO, F., ESTERT, M., QIAN, W., AND ZHOU, A. Density-based clustering over an evolving data stream with noise. In *Proceedings of the 2006 SIAM International Conference on Data Mining* (2006).

[11] CASCANTE-BONILLA, P., TAN, F., QI, Y., AND ORDONEZ, V. Curriculum Labeling: Revisiting Pseudo-Labeling for Semi-Supervised Learning. In *Proceedings of the 34th AAAI Conference on Artificial Intelligence (AAAI-21)* (2021).

[12] CHAO, X., AND ZHANG, L. Few-shot imbalanced classification based on data augmentation. *Multimedia Systems* (2021), 1–9.

[13] CUI, Y., JIA, M., LIN, T.-Y., SONG, Y., AND BELONGIE, S. Class-balanced loss based on effective number of samples. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2019).

[14] DORAN, G., LU, S., MANDRAKE, L., AND WAGSTAFF, K. Mars orbital image (hirise) labeled data set version 3. *NASA: Washington, DC, USA* (2019).

[15] FEI, G., WANG, S., AND LIU, B. Learning cumulatively to become more knowledgeable. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (2016).

[16] FENG, C., MAO, M., ZHANG, X., LIAO, Y., XIAO, X., LIU, H., AND LIU, K. Programmable microfluidics for dynamic multiband camouflage. *Microsystems & Nanoengineering 9*, 1 (April 2023). (https://doi.org/10.1038/s41378-023-00494-3).

[17] GEORGE, S. *Low-Bandwidth Remote Sensing of Rare Events*. PhD thesis, Carnegie Mellon University, Computer Science Department, March 2023. Technical Report CMU-CS-23-104.

[18] GEORGE, S., HARKES, J., EISZLER, T., AND STURZINGER, E. Hawk Source Code. (https://github.com/cmusatyalab/hawk). Last accessed July 22, 2023.

[19] GEORGE, S., TURKI, H., FENG, Z., RAMANAN, D., PILLAI, P., AND SATYANARAYANAN, M. Edge-Based Privacy-Sensitive Live Learning for Discovery of Training Data. In *NetAISys '23: Proceedings of the 1st International Workshop on Networked AI Systems* (Helsinki, Finland, June 2023).

[20] HE, K., ZHANG, X., REN, S., AND SUN, J. Deep residual learning for image recognition. In *Proceedings of IEEE Computer Vision and Pattern Recognition* (2016).

[21] HEIDEMANN, J., STOJANOVIC, M., AND ZORZI, M. Underwater sensor networks: applications, advances and challenges. *Philosophical Transactions of the Royal Society A 370* (2012).

[22] HP. OMEN Transcend Laptop 16-u0097nr. (https://www.hp.com/us-en/shop/pdp/omen-transcend-laptop-16-u0097nr). Last accessed July 12, 2023.

[23] INSITU. ScanEagle: The UAS that invented the agile ISR category. (https://www.insitu.com/products/scaneagle). Last accessed July 12, 2023.

[24] JEONG, M., CHOI, S., AND KIM, C. Few-shot open-set recognition by transformation consistency. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2021).

[25] JOCHER, G., CHAURASIA, A., STOKEN, A., BOROVEC, J., NANOCODE012, KWON, Y., TAOXIE, FANG, J., IMYHXY, MICHAEL, K., LORNA, V, A., MONTES, D., NADAR, J., LAUGHING, TKIANAI, YXNONG, SKALSKI, P., WANG, Z., HOGAN, A., MAMMANA, L., ALEXWANG1900, PATEL, D., YIWEI, D., YOU, F., HAJEK, J., DIACONU, L., AND MINH, M. T. ultralytics/yolov5: v6.1 - TensorRT, TensorFlow Edge TPU and OpenVINO Export and Inference. (https://doi.org/10.5281/zenodo.6222936), Feb. 2022.

[26] KIM, J., HUR, Y., PARK, S., YANG, E., HWANG, S. J., AND SHIN, J. Distribution Aligning Refinery of Pseudo-label for Imbalanced Semi-supervised Learning. In *Proceedings of the 34th Conference on Neural Information Processing Systems (NeurIPS)* (Vancouver, Canada, 2020).

[27] LEE, D.-H. Pseudo-Label : The Simple and Efficient Semi-Supervised Learning Method for Deep Neural Networks. In *ICML 2013 Workshop : Challenges in Representation Learning (WREPL)* (Atlanta, GA, 2013).

[28] LI, T., SAHU, A. K., TALWALKAR, A., AND SMITH, V. Federated Learning: Challenges, Methods, and Future Directions. *IEEE Signal Processing Magazine 37*, 3 (May 2020).

[29] LI, Y., WANG, T., KANG, B., TANG, S., WANG, C., LI, J., AND FENG, J. Overcoming classifier imbalance for long-tail object detection with balanced group softmax. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (2020).

[30] LIN, T.-Y., GOYAL, P., GIRSHICK, R., HE, K., AND DOLLÁR, P. Focal loss for dense object detection. In *Proceedings of the IEEE International Conference on Computer Vision* (2017).

[31] LIU, B. Lifelong machine learning: a paradigm for continuous learning. *Front. Comput. Sci. 11*, 3 (2017).

[32] LIU, Z., MIAO, Z., ZHAN, X., WANG, J., GONG, B., AND YU, S. X. Large-Scale Long-Tailed Recognition in an Open World. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2019).

[33] LYNN, S. K., AND BARRETT, L. F. 'Utilizing' signal detection theory. *Psychological science* (2014).

[34] MAHAJAN, D., GIRSHICK, R., RAMANATHAN, V., HE, K., PALURI, M., LI, Y., BHARAMBE, A., AND MAATEN, L. V. D. Exploring the limits of weakly supervised pretraining. In *Proceedings of the European Conference on Computer Vision (ECCV)* (2018).

[35] MEKKI, K., BAJIC, E., CHAXEL, F., AND MEYER, F. A comparative study of LPWAN technologies for large-scale IoT deployment. *ICT Express 5*, 1 (2019).

[36] MITCHELL, T., COHEN, W., HRUSCHKA, E., TALUKDAR, P., YANG, B., BETTERIDGE, J., CARLSON, A., DALVI, B., GARDNER, M., KISIEL, B., KRISHNAMURTHY, J., LAO, N., MAZAITIS, K., MOHAMED, T., NAKASHOLE, N., PLATANIOS, E., RITTER, A., SAMADI, M., SETTLES, B., WANG, R., WIJAYA, D., GUPTA, A., CHEN, X., SAPAROV, A., GREAVES, M., AND WELLING, J. Never-Ending Learning. *Communications of the ACM 61*, 5 (May 2018).

[37] MULLAPUDI, R. T., POMS, F., MARK, W. R., RAMANAN, D., AND FATAHALIAN, K. Background Splitting: Finding Rare Classes in a Sea of Background. In *Proc. of the 2021 IEEE/CVF Conference on*

Computer Vision and Pattern Recognition (CVPR) (2021).

[38] MULLAPUDI, R. T., POMS, F., MARK, W. R., RAMANAN, D., AND FATAHALIAN, K. Learning Rare Category Classifiers on a Tight Labeling Budget. In *Proc. of the 2021 IEEE/CVF International Conference on Computer Vision (ICCV)* (Montreal, Canada, 2021).

[39] NASA. Communications with Earth. (https://mars.nasa.gov/msl/mission/communications/). Last accessed July 12, 2023.

[40] NASA. Mars Curiosity Rover. (https://mars.nasa.gov/msl/spacecraft/rover/summary/). Last accessed July 12, 2023.

[41] OCHAL, M., PATACCHIOL, M., STORKEY, A., VAZQUEZ, J., AND WANG, S. Few-shot learning with class imbalance. arXiv preprint arXiv:2101.02523, 2021.

[42] PATTERSON, G., HORN, G. V., BELONGIE, S., PERONA, P., AND HAYS, J. Tropel: Crowdsourcing detectors with minimal training. In *Proceedings of the AAAI Conference on Human Computation and Crowdsourcing* (2015), vol. 3.

[43] PEDERSEN, M., BRUSLUND, HAURUM, J., GADE, R., AND MOESLUND, T. Detection of marine animals in a new underwater dataset with varying visibility. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops* (2019).

[44] SATYANARAYANAN, M., GAO, W., AND LUCIA, B. The Computing Landscape of the 21st Century. In *Proceedings of the 20th International Workshop on Mobile Computing Systems and Applications (HotMobile '19)* (Santa Cruz, CA, 2019).

[45] SETTLES, B. *Active Learning*. Morgan & Claypool Synthesis Series on Machine Learning, 2012.

[46] SHAO, J., WANG, Q., AND LIU, F. Learning to Sample: an Active Learning Framework. In *Proceedings of the IEEE International Conference on Data Mining (ICDM)* (2019).

[47] TAN, M., AND LE, Q. Efficientnet: Rethinking model scaling for convolutional neural networks. In *International Conference on Machine Learning* (2019).

[48] THRUN, S., AND MITCHELL, T. Lifelong robot learning. In *The Biology and Technology of Intelligent Autonomous Agents* (1995), L. Steels, Ed., Springer, pp. 165–196.

[49] TSAOUSIS, C. FireQOS Reference. (https://firehol.org/fireqos-manual.html).

[50] XIA, G.-S., BAI, X., DING, J., ZHU, Z., BELONGIE, S., LUO, J., DATCU, M., PELILLO, M., AND ZHANG, L. DOTA: A Large-Scale Dataset for Object Detection in Aerial Images. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2018).

[51] YAO, S., ZHAO, Y., ZHANG, A., SU, L., AND ABDELZAHER, T. DeepIoT: Compressing Deep Neural Network Structures for Sensing Systems with a Compressor-Critic Framework. In *Proceedings of SenSys '17* (Delft, Netherlands, 2017).

[52] ZHANG, C., SONG, N., LIN, G., ZHENG, Y., PAN, P., AND XU, Y. Few-Shot Incremental Learning with Continually Evolved Classifiers. In *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (2021).

[53] ZHANG, Y., KANG, B., HOOI, B., YAN, S., AND FENG, J. Deep long-tailed learning: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2023). (https://doi.org/10.1109/TPAMI.2023.3268118).

[54] ZHDANOV, F. Diverse mini-batch active learning. *arXiv preprint arXiv:1901.05954* (2019).

[55] ZHU, X., ANGUELOV, D., AND RAMANAN, D. Capturing Long-Tail Distributions of Object Subcategories. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition* (2014).

[56] ZIKO, I., DOLZ, J., GRANGER, E., AND AYED, I. B. Laplacian regularized few-shot learning. In *Proceedings of the 37th International Conference on Machine Learning* (2020), vol. 119.