Uṣás: A Sustainable Continuous-Learning Framework for Edge Servers

Cyan Subhra Mishra, Jack Sampson, Mahmut Taylan Kandemir, Vijaykrishnan Narayanan and Chita R Das
Department of Computer Science and Engineering, The Pennsylvania State University, University Park, USA.
{cyan, jms1257, mtk2, vijaykrishnan.narayanan, cxd12}@psu.edu

2024 IEEE International Symposium on High-Performance Computer Architecture (HPCA) | 979-8-3503-9313-2/24/$31.00 ©2024 IEEE | DOI: 10.1109/HPCA57654.2024.00073

***Abstract*—Edge servers have recently become very popular for performing localized analytics, especially on video, as they reduce data traffic and protect privacy. However, due to their resource constraints, these servers often employ compressed models, which are typically prone to data drift. Consequently, for edge servers to provide cloud-comparable quality, they must also perform continuous learning to mitigate this drift. However, at expected deployment scales, performing continuous training on every edge server is not sustainable due to their aggregate power demands on grid supply and associated sustainability footprints.**

To address these challenges, we propose Uṣás, an approach combining algorithmic adjustments, hardware-software co-design, and morphable acceleration hardware to enable the training of workloads on these edge servers to be powered by renewable, but intermittent, solar power that can sustainably scale alongside data sources. Our evaluation of Uṣás on a real-world traffic dataset indicates that our continuous learning approach simultaneously improves both accuracy and efficiency: Uṣás offers a 4.96% greater mean accuracy than prior approaches while our morphable accelerator that adapts to solar variance can save up to {234.95kWH, 2.63MWH}/year/edge-server compared to a {DNN accelerator, data center scale GPU}, respectively.

I. Introduction

The rampant growth, and anticipated sustained expansion of data collection and consumption are currently driving data-driven analytics using trained inference models, with significant economic impact. Amidst the myriad of data-driven domains, urban mobility, smart cities, autonomous driving, and the Internet of Things (IoT) emerge as some of the most rapidly expanding fields contributing to the global economy, amounting to more than 4 trillion US dollars [1], [54], [76], [98]. These statistics underscore the profound significance and transformative potential of these data-driven realms, delineating their pivotal role in shaping the landscape of computing technology, from algorithms to architecture.

What distinguishes these data is their diverse origin, spanning from IoT devices to wearables, and their acquisition from challenging environments, including autonomous driving and urban mobility scenarios. Consequently, they frequently exhibit a phenomenon known as "data drift", where the incoming data deviates from the distribution of the originally trained model, leading to degradation in inference accuracy.

Mitigating Data Drift: Dealing with data drift in edge compute nodes presents a significant challenge. While larger models with more parameters may exhibit limited data drift due to their increased capacity to generalize, deploying such large models on edge compute nodes can be difficult due to inherent limitations in form factor, energy efficiency, thermal constraints, and compute resources. To accommodate these constraints, it is a common practice to employ compressed Deep Neural Network (DNN) models, that are quantized, distilled, or otherwise reduced in size. However, while compressed models are essential for meeting resource limitations, they are more sensitive to data drift because they may not generalize as effectively.

Traditionally, data drift has been handled by cloud-based periodic re-training using continuous learning algorithms [20], [74]. However, there are challenges in resources, privacy, and sustainability to utilize existing techniques at envisioned scales. As these applications become more ubiquitous, particularly in urban deployments for tasks like traffic surveillance, autonomous driving, and health analytics [18], [77], [90], demands on communication bandwidth and network reliability limit the direct streaming of diverse data (e.g., video, 3D point cloud, sensor, voice) from numerous sensor-compute nodes to the cloud. Moreover, recent changes in privacy regulations across multiple countries [2], [100] call for preserving the privacy of citizens [12] and may preclude streaming personal data to third-party cloud services. As a result, "on-premise" *edge servers* [7], [8] have become prime choices for local inference and prediction [6], [39], [85], [86], necessitating the handling of both learning and inference tasks to meet application needs, including privacy preservation, reduced data communication, and disaggregated computing. Finally, although recent studies have suggested co-locating training and inference [12] to tackle privacy concerns without significantly affecting the inference service, the power demand associated with equipping multiple commercial edge servers [7], [8] for both tasks hinders sustainable scaling.

The Problem Space: To address the multi-faceted challenges of sustainable, scalable and privacy-preserving continuous learning at edge servers, several crucial problem spaces must be explored. Firstly, the issue of **(non-)supervision** arises, demanding the ability to label data without human intervention to preserve privacy during the learning process. While recent works [12], [46] have attempted to tackle this concern through student-teacher paradigms, efficiently deploying such approaches in complex data modalities (e.g., multi-class video, 3D point cloud) remains a formidable challenge. Ensuring adherence to Service Level Agreements (SLAs), where inference typically utilizes a lower-resource model [51], [75] and labeling is performed using a larger teacher model [44],

2378-203X/24/$31.00 ©2024 IEEE
DOI 10.1109/HPCA57654.2024.00073

[50], [73] at a much lower rate, necessitates informed decisions regarding deployment placement and sampling rates.

Secondly, the issue of **functionality** comes to the fore, requiring effective continuous learning from often non-Independently and Identically Distributed (non-IID) data. Such non-IID data distributions, evident in tasks like standard traffic monitoring with varied class observations (e.g., more cars than buses, all frames having `STOP` signs), may introduce *sampling bias* [70], [74] in the network. This challenge can be addressed through proper *exemplar selection* algorithms employing representation learning techniques [31], [74], capable of learning new classes in real-time. However, these compute-intensive algorithms can be optimized further through dedicated hardware acceleration.

Thirdly, the aspect of **sustainability** poses a critical question of deploying such systems, ideally with minimal reliance on the power grid for learning tasks. Designing a learning platform that can adapt to intermittent renewable energy sources (e.g., solar power) and maintain a minimal operational carbon footprint [29] is paramount. Such a platform should continuously make progress on unsupervised labeling, exemplar building, and continuous learning, and maximize *drift mitigation* while minimizing power consumption. Moreover, the system must accommodate **support for intermittency** inherent in sustainable power sources like solar and wind. While incorporating conventional battery storage can mitigate intermittency, it introduces environmental and sustainability challenges associated with resource extraction, production, and replacement [3], [5], [10], [13], [53], [66], [69]. An ideal solution would entail a battery-free system (*not* energy storage-free, i.e., still with some capacitive storage), circumventing these concerns and aligning with the objectives of sustainable and reliable continuous learning at the edge.

To these ends, we propose Uṣás, [1] a HW-SW co-design approach to building sustainable, scalable, drift-mitigating edge analytics platforms using harvested power to support continuous learning. Uṣás, unlike prior edge-focused analytics approaches (e.g., Ekya [12]), detaches the inference and training hardware, as the training task is the major source of the compute, power, and time consumption. Uṣás introduces an algorithmic framework for data labeling using a teacher-student model, designing the exemplar selection using representation learning and determining the right set of hyperparameters using micro profiling to energy-efficiently continuously train the DNNs with the selected exemplar sets. Uṣás also employs a dynamically morphable systolic array for enabling energy-efficient computing within the harvested power envelope. **Key contributions** of the work include:

- We propose algorithmic enhancements of *continuous learning* for mitigating data drift and design a *student-teacher based automated data labelling algorithm*, to prepare training exemplars from input data. We use a two-level data annotation mechanism: exemplar identification based on the confidence matrix of the student model, followed by a representation learning based exemplar selection by ensembling multiple teacher models. Our policy updates *both* the teacher and student models for robust unsupervised learning.
- We implement a *micro-profiler*, which predicts the right set of hyper-parameters to efficiently perform the training tasks on an energy-harvesting edge server while operating within its power budget and minimizing data drift.
- We design a *morphable hardware accelerator* that efficiently maps training tasks, is suitable for intermittent computing, and can adapt its capabilities to reduce power emergencies without devolving to grid operation. We discuss how the proposed hardware techniques can be adapted by many of the current DNN training accelerators to add similar dynamism in sustainability-sensitive environments.
- Finally, we evaluate Uṣás in depth on a *real-world traffic data set* [97] and perform sensitivity studies on other classes (audio, IMU) of data. Our algorithmic framework for performing continuous learning has a 4.96% greater mean accuracy than a naïve continuous learner. Power estimations of our hardware design, modeled by Design Compiler [93], indicate that the proposed morphable accelerator approach can save up to 234.95kWH/year/edge-server, compared to running continuous learning on a state of the art DNN accelerator and 2.63MWH/year/edge-server, compared to utilizing a datacenter-scale GPU for learning on the edge.

[1] Vedic goddess of dawn in Hinduism [36]; emphasizing the dawn of sustainable continuous learning and significance of solar power in our design.

II. Background and Motivation

Edge servers often leverage the convenience and flexibility of cloud interfaces, granting access to the same APIs, tools, and functionalities [60]. However, due to their inherent limitations in resources, such as weak GPUs and smaller memory capacities [83], these servers often resort to "customized" analytics services to maximize throughput and meet SLAs, including specialized DNN models tailored for edge deployments [51], [75], which are compressed, quantized, and optimized for the targeted hardware [30], [103], [109]. These tailored models enable accurate inference with high throughput and reduced resource footprint, with some compressed models having approximately 50× fewer parameters [30], but with a greater susceptibility to data drift [42], [55].

Data drift emerges as a significant concern in real-world systems as the live data diverges from the original training data, and the environment undergoes rapid changes [12]. Fig. 1 depicts our experimental investigations on data drift, encompassing training and testing multiple DNNs on diverse datasets such as Urban Traffic [12], [97], 3D Point Cloud [14], [24], and audio [78]. The similar trends across these results highlight the impact of varying time windows and encountering diverse scene changes, leading to degradation in network accuracy by up to 30%. These findings underscore the critical challenge posed by data drift and the need for continuous learning on edge servers.

Continuous Learning at the Edge: Continuous learning, wherein the model continually learns from new samples over time, adapting to seen and previously unseen classes, has

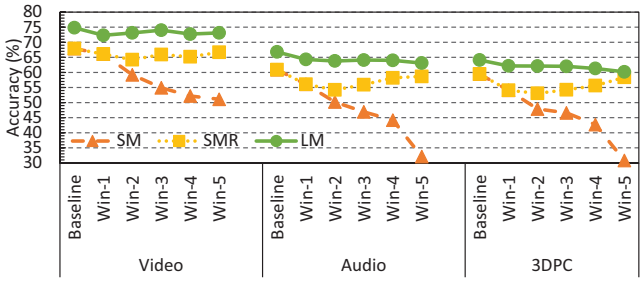


Fig. 1: Data drift on different data modalities. Sampling window size: 4hours for video, 20 minutes for audio for urban traffic video and audio data. 1hour for 3D Point Cloud simulated data. [SM:Small Model (smaller model or larger model pruned and quantized using energy aware pruning [102] and NetAdapat [103]), LM:Large Model (no pruning or quantization), SMR:Small Model with Retraining].

emerged as a preferred approach to mitigate data drift [20], [44], [50], [74]. The temporal locality of (video like) data has shown models to effectively learn from recent data. Although, multiple task-dedicated models are typically deployed to enhance accuracy and reduce sampling bias [70], particularly in scenarios like traffic monitoring, where different time periods exhibit distinct traffic patterns, they are not immune to data drift. As depicted in Fig. 1, our experiments, on different modalities, shows the accuracy degradation due to data drift. Specifically focuing on video data, we observe that: using quantized MobileNet-v2 (14M paramters, 71.3% accuracy) as the small model and ResNet-101 (171M parameters, 76.4% accuracy) as the large model, the accuracy of the smaller model has degraded $> 20\%$ over 5 sampling windows (of 4 hours each), where as the effect is minimal in the larger model. However, with a proper retraining, the smaller model could keep up with the original accuracy. We also observe a similar trend over other modalities, making the importance of continuous learning clear for multiple domains.

However, in a continuous learning paradigm, training becomes an essential, repeatedly scheduled task whose computational and time costs cannot be considered a one-time overhead freely delegated to the cloud. A recent work, Ekya [12], has demonstrated that edge servers equipped with GPUs are capable of performing the necessary tasks for continuous learning within their form-factor-imposed resource constraints, provided that those resources are intelligently managed.

***Sustainable* Continuous Learning at the Edge:** Even given such advancements in continuous learning on edge servers, provisioning training resources at the edge for every sensing-to-analytics application entails sustainability questions. For example, a popular AWS outpost, a g4dn.12xlarge instance [83], consists of a 24 core Intel Xeon CPU (150W TDP) [71] with 192GB of memory and 4 NVIDIA T4 (with tensor cores, 70W TDP) [65] with 64GB GPU memory. A standard offering with $2\times$g4dn.12xlarge instances need 4kW power [7] (the compute units have a TDP of $\approx 1kW$ [65], [71]) for performing analytics. With state-of-the-art learning APIs [60] and intelligent co-location and scheduling of inference and continuous learning [12], these edge servers can support about 8 videos streams [12], resulting in $\approx 120W$ (just for compute) per video stream. Scaling this to crowded cities with 30-50+kilo-cameras like Beverly Hills ($> 35k$ [11]), Los Angeles ($\approx 35k$ [92]), New York ($\approx$56k [92]), or Chicago ($\approx$30k) will need a lot of power. In fact, it will take $\geq$3Million cameras (assuming $\approx$9 cameras/1000 people, similar to LA, and scaled to US population) to just enable autonomous urban mobility in the USA, which may consume $360MW$ power ($1296GWh$ energy, 0.03% of US power) for video analytics alone. Clearly, the current solution is *not* sustainable, neither in terms of the load on the power grid, nor in terms of the CO_2 footprint (1.1×10^9lbs); reducing the power budget for continuous learning is essential, as the carbon footprint of DNN training has emerged as a prominent concern [21], [57], [67], [89], demanding careful consideration as a primary design metric.

Although green data centers [58], [59] provide partial mitigation, they fail to address data privacy and communication bandwidth challenges in the current context. Similarly, other applications with diverse data modalities, such as LiDAR and Camera for autonomous driving, IMU, bio-sensors, and Speech for IoT, face similar issues. Thus, *attaining a sustainable solution for privacy-preserving, distributed continuous learning remains an ongoing pursuit.*

Exploiting Intermittent Computing: An obvious solution to the power problem is to run the training in a self-sustained way, i.e., without depending on the power grid and by relying on a renewable energy source like solar power; opportunities for harvesting renewables naturally scale alongside a greater number of deployment locations and solar power, even though not always available, is in abundance. In the United States, a typical 12% efficient solar panel [91], can provide an annual average of $50W/m^2 - 150W/m^2$ of power [64]. Furthermore, solar power has reasonably predictability characteristics. Typically, inference tasks have significantly less compute time and power requirement, and commercial off the shelf devices, like edgeTPU [19] can perform object detection using the aforementioned compressed models at a reasonable frame rate (at times $\geq 71fps$). Therefore, *designing a training platform to perform continuous learning with the intermittent solar power and within the typical harvested budget* would be the best solution. The power sustainability consequently reduces the cost of deployment as the publicly available edge server, like AWS outpost offering (one of the cheaper and lower power consuming ones) for performing edge inference costs $5,134.92/month [83].

Our Approach (and its Novelty): *Uṣás* introduces several novel contributions in the domain of *sustainable* continuous learning at edge servers using harvested energy, setting it apart from prior works examining on-edge learning.

Battery-Free Operation: A key highlight of *Uṣás* lies in its battery-free operation, which aligns with the current global push for sustainable computing. The scaling up via mil-

lions of additional battery-supported analytics platforms would introduce severe environmental challenges due to resource extraction, production, and replacement of batteries [3], [5], [10], [13], [53], [66], [69]. By demonstrating the viability of a battery-less edge server for video analytics, Uṣás spearheads the adoption of similarly sustainable systems for other domains. While the initial scope is limited to urban mobility applications, the concept's adaptability extends to various domains, including autonomous driving, smart industries, and remote sensing: Section V-D performs an initial exploration of how techniques from Uṣás will apply to other domains.

Algorithmic Advancements: *Uṣás* extends the frontier of representation learning for continuous learning by implementing it at a large scale and addressing related challenges. Prior works relied on supervised learning or K-means clustering, unsuitable for *Uṣás* due to its need for unsupervised data annotation and the inability to handle large-scale datasets with numerous classes. To overcome these limitations, *Uṣás* employs an ensembled teacher-student method, wherein multiple teachers annotate student data. A hierarchical K-means+ (or DBSCAN) clustering approach learns representations for exemplar selection. Additionally, a novel power-aware micro-profiling policy is adapted to determine optimal hyper-parameters for a variable-power environment. The robust exemplar selection and micro-profiling mechanisms are discussed and evaluated in §III-B and §III-C, respectively.

Hardware Innovation: Uṣás embraces the intermittency entailed by harvesting and advocates for hardware adaptation (resizing) to efficiently manage variable power income and avoid power emergencies. While previous works have designed energy-efficient training hardware with support for variable precision training, none have adapted to variable energy income. *Uṣás* optimizes the entire solution space, maximizing hardware reuse for exemplar selection and micro-profiling while addressing the training task. The system can turn off individual compute-tiles to accommodate runtime power variability (see §IV-B) and enable seamless operation during power reductions.

Overall, Uṣás demonstrates the viability of sustainable continuous learning at edge servers, encompassing advancements in energy harvesting, algorithmic techniques, and hardware adaptation.

III. Continuous Learning

The first step to any data-driven learning algorithm is data collection and annotation. Since *Uṣás* is a continuous learning framework and learns from the live data that the camera(s) capture, data collection is simply storing the live video feed. However, data annotation or labeling is more challenging. Classically, once data is collected, it is classified, labeled, and bounded by borders (bounding box) mostly using manual labor (at times with software assistance) or crowd sourcing [33], [82], [88]. This requires the data to be present at a central location for manual inspection, both of which are not possible because of communication and privacy constraints. Therefore, we adapt a "student-teacher paradigm" [46], where a more general, robust and larger model (typically with hundreds of millions of parameters [43], [99]) helps in annotating the data. However, because of the heavy compute requirements, the teacher model runs with a much slower frame rate and annotates only some (important) frames. There has been a significant body of work on frame similarity and saliency [45], [84], [101], [105], [107], [108], and those details remain beyond the scope of this work.

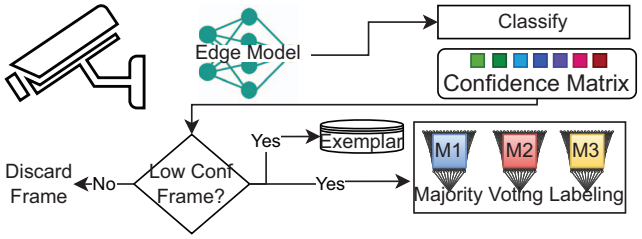


Fig. 2: Auto-labeling in *Uṣás*: Select frames only with low confidence as they might contain potentially new information, and use ensemble learning to improve the labeling.

A. Data Annotation

Picking the Important Ones: Typically, edge models are capable of inferring at the frame rate of the camera (at times, 30fps to 60fps) [19]. However, the teacher model used to label the incoming data cannot match this in a resource-constrained environment where performing training is going to be even more resource consuming. Therefore, we employ an intelligent "data sampling mechanism" to select the frames that might contain new information and a potential candidate for learning. Fig. 2 shows the different components of the student-teacher data annotation model adapted in *Uṣás*, where the edge model is the "student" (continuously retrained), and larger models are the "teachers" (the ones teaching the student about what-is-what). The students models are typically optimized for edge, i.e. with optimizations like quantization, pruning etc. or by developing an application specific model from scratch along with the said optimizations. These student models, thanks to their lack of robustness (which is often, but not always, related to the smaller footprint they have, and thereby lacking the parameter space to generalize better), are susceptible to data drift and hence are continuously retrained. However, the teacher models are typically large, and with a wide parameter space can generalize the learning process better than the students. These teacher models are often factory trained. As they are less prone to drift, they need occasional updates. For each sampled frame, the classification results and the confidence matrix (output of the last layer) are sent for annotation. If the student (or the edge model) is confident about the classification (e.g. a clear frame with no new objects, or a frame similar to one of the training samples), then that frame is discarded as it potentially contains little to no new information. However, if the student is not confident on the classification, the frame is then saved as a potential exemplar (we will further refine this in §III-B). The potential exemplars

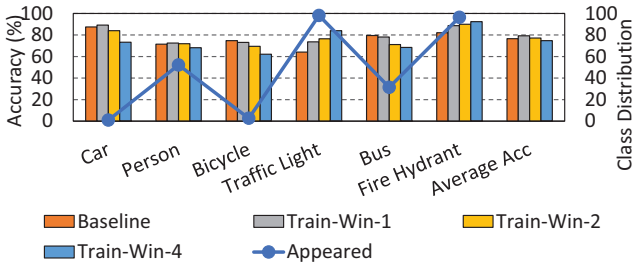


Fig. 3: Distribution of different classes on a typical traffic pattern and the impact of training on the sampling bias. The "appeared" line represents the percentage of the frames in which the corresponding class is present, e.g. Fire hydrant, in the taken scene, is present in 100% of the frames. Incorrect exemplar selection might lead to non-IID training data distribution, leading to catastrophic forgetting or over-fitting.

are then further refined and classified by the teacher models. To improve the confidence of the teacher models, we employ an ensemble learning based weighted majority voting policy [28]. Each of the teacher models infers on the exemplar frame. Furthermore, each teacher model has its private confidence matrix on different object classes. This confidence matrix serves as a weight for performing the ensemble of multiple teachers, and helps exploiting the expertise of each of the teacher models for each of the individual classes, significantly boosting the accuracy and robustness of the data annotation. This maximizes the accuracy of the teacher, and consequently minimizes the chance of the student model learning wrong labels. Note that the limited parameters of the student make it more sensitive to data fidelity and hence ensuring an accurate data labeling is very important for end to end classification accuracy. The impact of wrong labeling is discussed in §V.

The Problem: However, this exemplar section mechanism has an inherent flaw. Consider a traffic camera looking at a busy street with a traffic signal. Due to the traffic distribution (e.g., more cars than buses), the camera typically sees a varied distribution of different classes, which might reflect in the exemplar set. Moreover, some static objects (traffic light, stop sign, etc.) might be present in all frames. This creates a sampling "bias" [70] while performing the training, and often leads to catastrophic forgetting. Fig. 3 shows a typical traffic distribution from Urban Traffic data [97]) and the impact of sampling bias on class distribution. Note that, as some of the classes (e.g., bicycles) are barely present in the exemplar, the model tend to lose accuracy (because of catastrophic forgetting) on them, whereas the model rapidly over-fits for the classes with more examples (e.g., traffic light).

B. Proper Exemplar Selection

To tackle the sampling bias [70], we adapt a representation learning [74] framework for designing the proper exemplar selection. The fundamental issue with the previous approach is the inability to select correct numbers of IID data for training. In addition to that, just DNN training cannot learn new classes if there is no way to annotate and label new classes. Representation learning solves both these issues.

The learner (here the teacher models) need to properly classify the data, learn if the data is a new type of one of the older classes, and identify if it encounters a new class. We achieve this by clustering the feature vector of the Large DNN model. Fundamentally, we use the larger DNN models as feature extractors which turn the data into a feature vector. In the original training phase, these feature vectors are separated using K-means [74] or other clustering. The cluster centers for each data (μ_y for class y) are calculated as $\mu_y = \frac{1}{P_y}\sum_{p \in P_y} \Phi(p)$, where P_y is the number of samples belonging to class (or cluster y), and Φ is the feature extraction function working on the data p. These clusters represent the classes in the high dimensional feature space. When the classifier sees new data (x), it calculates its distance from all the cluster centers as $y^* = \min_{y=1\ldots t} ||\Phi(x) - \mu_y||$. There are three cases:

Case-1: If the data is close to one of the cluster centers and belongs to its cluster boundary, then it falls into the bucket of that particular class. This typically happens if the data are very similar to the training samples.

Case-2: If the data belongs to a known class, but is significantly different from the training samples, it falls not too far from one of the clusters. This distance of the new data from the cluster center is called the "distillation loss" [74]. An encounter of a new example of the existing class is followed by an update to the clustering by minimizing the classification loss of the newly-seen data.

Case-3: Finally, if the classifier sees an example of a new class then the feature vector of the data sits far from all the cluster centers indicating an unknown class. The distance of this feature vector from the other cluster center is called "classification loss" [74], and this re-triggers clustering with an updated number of clusters.

Over multiple time windows, the representation learner goes through all the possible exemplars selected by using the confidence matrix and creates an exemplar set with same number of examples from each possible class. Since we have multiple teacher models, each of them contributes to the exemplar set, making it robust and removing bias. To efficiently implement the exemplar selection algorithm, *Uṣás* implements the major portions using "custom hardware" (discussed in §IV-A). The annotations on the new exemplar set created by the representation learner is compared against the confidence matrix of the edge model to calculate the "drift". Consequently, this exemplar set becomes the training data for the continuous learning, which consequently minimizes the drift. Once the student model is trained with the exemplar set, the data is discarded and the feature space for the teacher models is updated. By doing this, *Uṣás* keeps *both* the student and the teacher models "updated." Since the feature space of the teacher model is updated using K-means+, the major computation is the training of the student model using the exemplar data. Although efficient hardware accelerators [16], [27], [80] have been developed to do the same, these accelerators are typically designed with a "throughput-first"

approach and are neither configured nor capable of operating with an intermittent power source. Deploying sufficient battery resources to allow intermittency-unaware designs to operate on solar power is neither efficient nor sustainable.

C. Hyperparameters: The Right Way to Learn

After finalizing the training set for continuous learning, the next challenge is to learn within the power and time budget. Given enough time even naïve low power hardware can finish training, but will have longer periods where the drift is exposed. A more preferable solution is to get rid of drift as quickly as possible, i.e. finish training on the exemplars (described in §III-B) as soon as possible and also reach the desired accuracy – but to do this within the harvested budget. Prior works [34], [48], [68] suggest that selecting the right hyper-parameters (like batch size, learning rate, number of layers to train etc.) have a huge impact on the convergence and accuracy of the models. For each edge servers to handle multiple steams with multiple drifts, we need to jointly optimize the hyper-parameters for maximizing accuracy with minimum power and resource budget.

To achieve this, we design a "micro-profiler" that can look into the drift of the models as well as the power availability and decide the right hyperparameters to train the models. Prior works [12], [38], [68] have designed hyperparameter micro-profilers. However, they never considered an intermittent power source, nor explored jointly optimizing multiple models with power, accuracy and latency constraints. Furthermore, each model might contribute differently to the overall accuracy. Observing this, we propose a "weighted accuracy metric", where the weight of each of the model is a function of the accuracy, time needed and power availability. Furthermore, we allow some slack to the weighted accuracy so that the optimizer can choose a better set of hyperparameters if we can reach **close to** the weighted accuracy with much lower resource (power or compute) consumption. Typically, there is an inverse correlation of the convergence of the stochastic gradient descent (SGD) algorithm, the most popular training algorithm for DNNs, over the number of iterations (n_i) [68]: $l \propto \mathcal{O}(1/n_i)$ and $l = \frac{1}{\beta_0 . n_i + \beta_1} + \beta_2$, where l is the loss of the SGD and β_i is an non-negative real number. Therefore, by running a few iterations of the SGD algorithms with various other hyperparameters, we can easily *predict* the convergence of the models. Note that this needs to be done every time one of the constraints (accuracy, power etc.) changes. The micro-profiler optimizes the weighted accuracy ($A_w = \frac{W_i}{A_i} / \sum W_i; \forall i \leq \#models; W_i = f(time, drift, compute)$ with a user-defined slack value of δ), with respect to available power (P_{av}): $\max A_w$; $s.t. P \leq P_{av}$.

Energy Buffering and Power-Predictor: To regulate, manage and ensure a stable power supply to the circuitry, *Ușás* uses a super-capacitor assisted voltage regulation circuit. To properly model the energy harvesting, losses during conversion, and leakage, we built a rectification circuit with $4 \times 5.5V$, $2.2F$ super-capacitors connected in parallel to a voltage regulator circuit. The harvested power is given as an input to a moving average power predictor [61], [72] to predict the future available power. Note that the power predictors used in prior works are meant for fickle energy harvesting scenarios like piezoelectric (movement), or RF (WiFi). We have adjusted the time window size. We took a history (years 2019 and 2020; from Seattle, WA; Sterling, VA; and Oak Ridge, TN) of solar energy traces from SOLRAD [25], [91] and built a weight matrix which looks into a window of 1 hour at 1 minute (average power) intervals to predict the power for next 10 minutes (1 minute granularity). We use regression to find the weights (exponents and coefficients) to the prediction curve followed by exponential smoothing to decay the weights. The rate of exponential smoothing depends on the scheduler used - while for the conservative scheduler the predictor always underestimated the power (shallow smoothing), the eager scheduling uses the direct output of the predictor (steeper smoothing). In either case, the predictor predicts the power with $\approx 95\%$ (peak of 98.72 (with real solar power trace) and minimum of 89.14 (with synthetic power trace) accuracy. The micro-profiler, having run multiple sweeps, returns a set of hyper-parameters (Ψ_i) for each model which is then stored in a history table. This helps us avoid unnecessary profiling (up to 41%). When introduced to a new set of constraints (change of power availability, accuracy etc.), the micro-profiler first looks in the history table to find a configuration and runs profiling if and only if it could not find one.

IV. The Morphable Hardware

Why Not Commercial GP-GPUs? DNN training is massively parallel, fairly compute intensive, time consuming, and needs a lot of (albeit structured) data movements [16], [37]. Therefore, GP-GPUs have classically been used to train DNN models. However, as mentioned in §I, the commercial GPUs used for DNN training are typically power hungry (typically in 100s of Watts TDP; We exprimented with multiple GPUs, server class A6000: 300W TDP, server class A100: 250W – 400W TDP, client class TRX3090: 350W TDP, and client class T4: 70W TDP), and are *not* equipped to handle intermittent power emergencies. However, these GPUs are often equipped with dynamic voltage and frequency scaling (DVFS). [2] To understand the impact of DVFS on energy savings and dynamic compute scaling, we implemented a simple multi-arm bandit algorithm to select the right bucket of compute frequencies (SM frequency for NVIDIA GPUs), and memory frequencies to match the power-demands of the intermittent solar source. As shown in Fig. 4 even with DVFS, commercial off the shelf GPUs could only finish $< 50\%$ of the scheduled training task. However, hardware is not the only limitation, as even with custom hardware [16] enabled with the state-of-the-art continuous learning algorithm [12] could only finish $\approx 75\%$

[2]NVIDIA provides the list of supported clocks through the API "`nvidia--smi --q --d SUPPORTED_CLOCKS`"; We did not creport the results from A100 for this, as it does not offer multiple memory clocks, significantly impacting its DVFS capabilities. T4, thanks to its limited compute capabilities, could not finish training tasks on time.

of the scheduled training without any intermittency support. It is clear that we *can neither* use the commercial GPUs *nor* rely on the standard software and algorithmic approach for intermittent training purpose as they *cannot* finish the compute given the intermittent power budget.

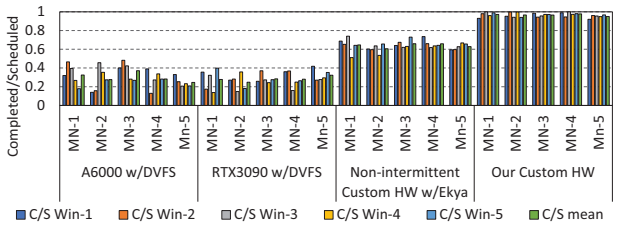
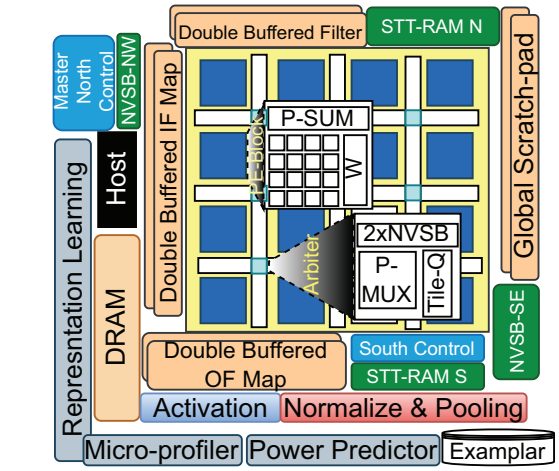


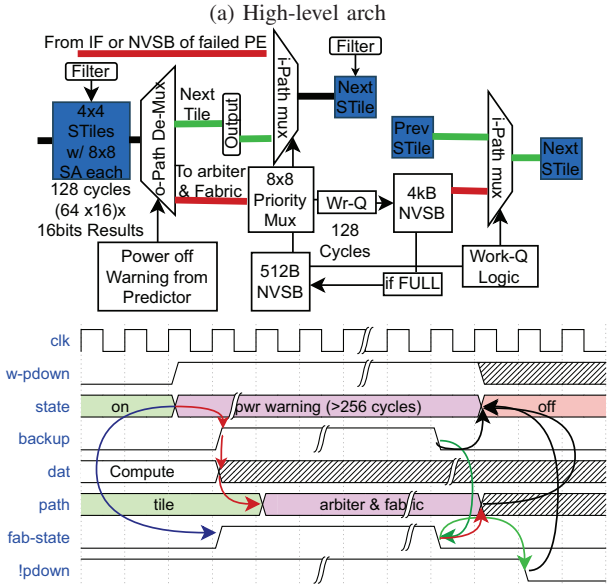
Fig. 4: Impact of DVFS on completion (average power budget 70W). Note that, even with DVFS, most scheduled compute could not be finished. This includes the intermittent failures ($\leq$20W where no compute could be done); we included checkpointing to ensure that progress is saved in power-failures. C/S is the ratio of Completed over the Scheduled training tasks over multiple time windows of 4hours. Our custom HW runs with intermittent support both by hardware and software.

There have also been significant efforts in designing and optimizing specialized DNN training accelerators [16], [27], [81], and many commercial organizations have already developed their own accelerators [37], [95] as well. Considering the compute mapping of the DNN training, almost all of these designs are based on a "systolic architecture", performing chains of multiplication and accumulations (MACs). However, these devices take a "throughput-first" approach, to minimize the time consumption and seldom optimize power consumption first. This has lead to a global concern of the energy and consequently carbon-footprint of the DNN training [21], [57], [67], [89]. Furthermore, these accelerators have been designed to operate under constantly available power. Although our proposed representation learning (§III) and micro-profiler (§III-C) help us find a better training configuration that can minimize the compute if deployed in the aforementioned accelerators, it does not solve sustainability: That is, with variable solar power, can we scale compute alongside power to continue to make "forward progress", even when minimum amount of power is available. The systolic array structure of the DNN accelerators is well suited for this as we can change the compute size, as well as the number of memory channels feeding to those compute units as per the power availability. However, we need to be innovative in terms of designing and placing the compute hierarchy to ensure minimum data movement and re-computations when compute scaling. The hardware design of *Uṣás* (Fig. 5a) incorporates all the aforementioned points. Note that, *Uṣás* introduces a design philosophy for building a morphable hardware, and it can easily be adapted by any of the systolic array based commercial off the shelf (or research prototype) DNN training accelerators.

DNN Compute Mapping: Typically there are three ways of mapping DNN compute into a systolic array, namely, 1. output



(a) High-level arch



(b) Power-down/ Failure handling

Fig. 5: Overall architecture with the components and the power failure handle sequence of *Uṣás*.

stationary; 2. input stationary; and 3. weight stationary [79]. Most large-scale accelerators use the output stationary implementations to minimize the output feature map movement [81], and some of available hardware even supports multiple types of mappings [15], [37]. However, our design objective is to *minimize* data movements in the case of compute reconfiguration. In an output stationary mapping, *both* input and weights are dynamic and any power failure or reconfiguration will need to save and restore a lot of current context (partial sums, indices of weights and inputs etc.) to resume and remap the compute. This problem reduces in both input stationary and weight stationary, but at the cost of throughput [80]. Typically, the input feature maps are larger than the (individual) weights, and more importantly large weights can easily be represented

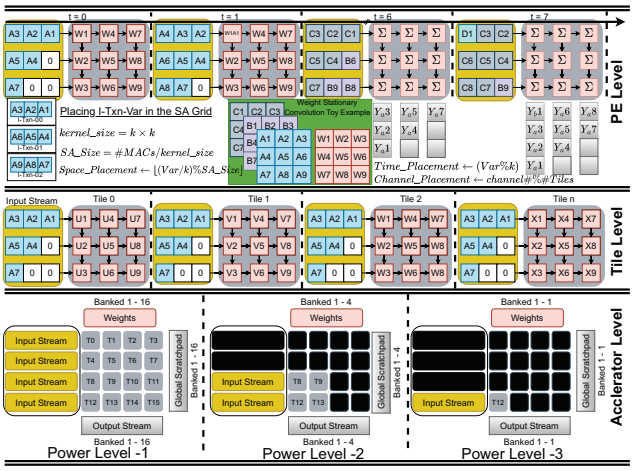


Fig. 6: Weight stationary compute mapping. The PE-level shows how the input flows and the convolutions are computed with a 3x3 convolution toy example. The tile-level shows how each tile consists of multiple such PEs and will be working on one kernel at a time. The accelerator-level shows that the entire accelerator is made of multiple such tiles (4x4 in the toy example). Inputs are broadcast into each tile so that each tile can work on a kernel. Computation is *redistributed* when there is a change in the power availability, and multiple tiles are shutdown (redacted) without impacting the data flow.

or decomposed as multiple units called "kernels" (or "filters"). In a typical convolutional neural network (CNN), each kernel is convoluted over the entire input feature map, and hence there is an "inter-kernel parallelism" (all kernels of a single layer can be executed in parallel) and "intra-kernel parallelism" (multiple computes in a convolution can happen in parallel). This property is true both for the forward pass and the backward pass of the standard CNN training. The modular nature of the weight stationary mapping makes it a strong candidate for use in a re-configurable or morphable systolic structure as turning off some compute is the same as not computing a kernel and scheduling it for later. Therefore, *Uṣás* employs a weight stationary compute mapping for executing the training tasks on the morphable hardware.

A. Design Description of the DNN Hardware Augmentations

Compute Mapping: Fig. 5a shows the high level design, architecture and different components present in our proposed accelerator. The accelerator encompasses 256 tiles, structured in a 4×4 configuration of 16 super-tiles, each harboring 4×4 tiles. These super-tiles Each tile, individually switchable ON or OFF based on power availability, houses 64 16-bit floating point MAC units configured in an 8×8 systolic array for convolution operations. A modular computational approach is adopted where each tile is accountable for one CNN kernel, necessitating $\lceil [C \times H \times W]/64 \rceil$ iterations for a kernel of size $[C \times H \times W]$. Data streaming and partial compute storage are facilitated by four double buffered SRAM structures, with the weights residing in a double buffered multi-banked SRAM. The filter SRAM has 256 banks (one per tile), each with a size of 1kB (double buffered, 512B per buffer per bank). Input data broadcast to all tiles is managed by a 64kB double-buffered input feature map SRAM (32kB each), requiring $\lceil [X \times Y \times Z]/1024 \rceil$ iterations for full input loading, with each buffer loaded $[X \times Y \times Z]/2048$ times. The convolution map transforms $[X \times Y \times Z] \xrightarrow{M \times C \times W \times H} [M \times U \times V]$ to yield an output tensor of dimensions $[M \times U \times V]$, supported by a 256 banked double buffered output feature map SRAM, each bank of size 8kB (4kB/buffer). A 128kB SRAM serves as a scratchpad for storing activations, transposes, and intermediate differentials during the backward pass. The accelerator also houses 256×256 compactor-mux combinational logic units (256 units per tile) for ReLU activation (forward pass) and inverse activation (backward pass). For smaller DNNs without 256 kernels in any layer, a batching mode is operational with a batch size of $B = \lfloor A/L \rfloor$ images, where L denotes the layer with the fewest channels, and A the number of active tiles. This generic design is adaptable for various workloads.

In DNN training, meticulous compute mapping, memory access strategies, and operational formulas are instrumental for the forward and backward passes. The forward pass computes activations via the formula $A_{muv} = \sum_{c=0}^{C-1} \sum_{i=0}^{H-1} \sum_{j=0}^{W-1} X_{(u+i)(v+j)c} \cdot K_{mijc}$, with results stored in the double-buffered output feature map SRAM. The backward pass emphasizes gradient computation through backpropagation, which is crucial for weight updates. The gradient of the loss function concerning the weights is computed through the formula $\frac{\partial L}{\partial K_{mijc}} = \sum_{u=0}^{U-1} \sum_{v=0}^{V-1} \frac{\partial L}{\partial A_{muv}} \cdot X_{(u+i)(v+j)c}$. This gradient computation, fundamental for learning, is meticulously mapped across the systolic array, ensuring precise and efficient backpropagation. Memory accesses are optimally managed via the double-buffered SRAM structures, providing timely data availability for the MAC units. The 8×8 systolic array in each tile executes multiply-accumulate operations in a pipelined and parallel fashion, abiding by the Weight Stationary approach, thereby optimizing the throughput and efficiency of the training operations within this hardware architecture.

Power Control Logic: Power emergency prediction in *Uṣás* is always conservative, and the solar power predictor has a mean accuracy of 92%, limiting false positives and helping the control unit select appropriate tile counts. The system needs at least 512 cycles of advanced notice to flush compute and enable a compute migration. Fig. 5a shows the block diagram of the mesh interconnect, and Fig. 5b shows the power-down sequence and signal states. The network works at a super-tile (STile) granularity and each arbiter node uses an 8x8 priority-mux. The network only gets activated when it gets a *w-pdown* warning signal from the predictor. This signal starts a graceful power-down sequence for the required number of tiles. The *w-pdown* triggers the *backup* signal and the system goes into *pwr-warning* state (other states being on, off, invalid and X). In the *pwr-warning* phase the system finishes the remaining compute of the systolic arrays (which can take up to 64 cycles),

and starts flushing the results for backup.

Buffer Management: *Uṣás* uses non-volatile state buffers (NVSBs, 18 count, 1 per 4x4 tiles, each of 1kB, and 2 of 4kB each) for state saving and data backup. The control logic prioritizes writing data into the local NVSB for the arbiter (each arbiter caters to 2 STiles). If those get full because of continuous power failures, the control directs the data to the global NVSBS (NVSB-NW and NVSB-SE in Fig. 5a). The NVSB stores the global/asynchronous work queue and the shuffling configuration (mini-batch arrangement).

B. Power Failure and Compute Scheduling

Central to the *Uṣás* accelerator's operational efficiency is the work queue—an intricately designed, hierarchical structure that meticulously catalogues pending computational tasks. Each task, represented in the queue, corresponds to the execution of specific CNN kernels, feature tiles and operations. As deep neural network models often have a complex interplay of layers, each with distinct computational needs, the work queue ensures a systematic, prioritized approach to handle these operations. Two distinct scheduling strategies, each complemented by its own type of work queue, govern the computational flow: Conservative Scheduling and Eager Scheduling. The dual-scheduling mechanism, bolstered by the work queue's flexible architecture, not only optimizes compute performance but also offers resilience against power uncertainties. The work queue schedule, intermediate result, network and layer information are saved on predicted power failure, and data from the DRAM (the working set of IF/OF/filter and model state) are moved to an NV-RAM using STT-RAM based buffers in the memory hierarchy (parallel to the IF/OF/filter). We do *not* replace the DRAM buffers with NVM because of limited lifetimes [17]. The host writes the latest copy of the completed iteration (in epoch granularity) into the STT-RAMs (STT-RAM-N for the upper 128 SAs, and STT-RAM-S for the lower 128SAs, Fig. 5a). In case of a complete power failure, the compute in flight are rejected and, once the system starts working, the work queues get invalidated and the host starts the compute again from the last checkpoint. Along with that, the most common intermittent software libraries and software designs [26], [52] (and most DNN training libraries like PyTorch, TensorFlow) also offer periodic checkpoints. Note that the power-up sequence for a tile runs in the exact opposite order of the *powerdown* sequence (a tile becomes computationally active 512 cycles after it gets the power up signal). *Uṣás* uses two kinds of scheduling policies to handle the graceful *powerdown* and work queue rearrangement.

Conservative Scheduling: The most important part of the *Uṣás* accelerator design is to ensure proper "compute placement" even under a power emergency or power scaling. Fig. 6:**Accelerator level** provides a high-level overview of the compute scheduling (where the redacted part of the hardware is turned off because of the lack of power). The key components of the scheduler are the "moving average power predictor" and the "micro-profiler". In the i^{th} kernel scheduling iteration, given the power budget and power prediction, the micro-profiler decides the required training configuration, and the control logic (conservatively) enables suitable number of tiles (say t_i tiles of the 256 tiles). Those t_i tiles fetch t_i unique kernels from the 1Byte wide, 256 deep global kernel dispatch queue (GKDQ, t_i kernels scheduled in parallel). Note that the power requirement of each tile is known in advance (please refer to §V, TABLE I for details). Once the scheduled (t_i) tiles are completed, the micro-profiler again finds the right configuration for the $i+1^{th}$ iteration and the scheduler again conservatively enables t_{i+1} number of tiles suitable for the power budget. The t_{i+1} tiles fetch the next t_{i+1} kernels from the GKDQ and the process continues. This conservative compute and power estimation ensures that none of the kernel computes (the lowest decomposed level of compute unit for the hardware) ever fails and hence there is no need for any partial data movement. The GKDQ always points to the next available kernel location. The control fetches the right number of kernels and all of them are synchronously executed in the active tiles.

Eager Scheduling: A weight stationary implementation with a conservative scheduling will always run synchronously. However, in the middle of an kernel execution iteration, if the hardware gains access to more power which in turn can enable more tiles, it cannot do so without breaking synchrony (i.e. when some of the tiles are half way through the compute, some other tiles can just start execution). Facilitating such scheduling will provide us less idle time, more forward progress and more efficient use of the incoming power but at the expense of more control overheads. We call this *Eager Scheduling*. To enable eager scheduling, we decentralized the global kernel dispatch queue and equipped each tile with a local kernel dispatch queue (1Byte wide 16 deep). At the beginning of each kernel scheduling iteration, the micro-profiler decides the right configuration, and the control distributes equal number of kernels to each active tile (given A active kernel, and K total kernels, each tile gets $\lfloor K/A \rfloor$ kernels to execute). The conservative scheduler ensures that no tile loses power before finishing the current scheduled kernel. However, in the middle of the execution if any new tiles becomes alive (because of an increase in harvested power), the scheduler immediately marks it ready to start working and the tile fetches a kernel (currently not scheduled in any of the tiles) and starts working on it. We face three issues here: 1. How does the new tile get any kernel to work on? 2. Over multiple iterations of such asynchronous scheduling, the kernel queue for each tile will be of different size creating a load imbalance; how to tackle this? 3. How do we know when to stop executing?

To address the first two issues, we developed a work-stealing mechanism for each tile. When any of the active tiles are marked ready by the scheduler, the tile employees a state machine to decide where to get work from. Each time the tile finishes some work, if its remaining work queue (the local work queue size) is less than the average of all other active tiles, it seeks a new kernel to work on. Considering the global control always enqueues any idle tile with work, whenever the tile has no work left, it steals a kernel from the most

heavily loaded tiles. We implemented a counter (local kernel counter) to keep track of the size of the remaining local work queue of each of the tile. We also implemented a counter (layer kernel counter) which keeps track of the total kernels to be scheduled for each layer. Whenever all the local work queue counter hits zero along with the layer kernel counter, the control moves to schedule the next layer (or previous layer in backward propagation) for computation.

Note that we do not delve into the details of the computational primitives involved in training the DNN as several prior works [15], [16], [80] provide a very detailed accounting of it (for both forward and backward pass) along with the hardware and control requirements. We treat the convolution scheduling (using input stationary and at a kernel level) in a morphable systolic hardware to be the main challenge and explain it.

V. Implementation and Evaluation

We focus our evaluation on urban mobility, i.e. performing *single shot* object detection for traffic monitoring using the **MobileNetV2** [51] model on the urban traffic data set [97]. This is a traffic video dataset containing 62GB of videos recorded from five pole-mounted fish-eye cameras in the city of Bellevue, WA, USA. Each video stream is recorded with a resolution of 1280×720 at 30fps. This contains a total of 101 hour of video across all cameras of which 30 hours of video is used to fine-tune the teacher models and the rest 71 hours of data is used to evaluate our continuous learning solution. For annotating the incoming video stream we use three teachers models, namely, ResNet101 [32], YOLOV2 [22], and VGG16 [87]. For sustainability, we use solar power to perform our compute. Since our dataset is from Bellevue, WA, we took the SOLRAD solar radiation data [25] (managed and published by National Oceanic and Atmospheric Administration, NOAA) of Seattle, WA (the SOLRAD center closest to Bellevue and hence we believe is a good approximation). In our experiments we assume the hardware to be powered by a solar panel of one square-meter, and the powers are scaled accordingly (data is available as W/m^2). Finally, we assume the exact same setup of the Urban traffic dataset and hence have 5 different MobileNetV2 models trying to classify the traffic they are facing, and learning from the streaming data. We vary the training intervals to see the effect of frequency of retraining.

Existing Approaches: Although there has been significant research [40], [41], [47], [52], [56], [61], [72], [104] on enabling machine learning in intermittently powered devices, a majority of it focuses on performing inference. Only intermittent learning [47] focuses on performing on-device training, but with very small workloads and models. Considering the scale, scope and workload of our problem, limits direct comparisons, except for comparing their exemplar selection method (refer Fig. 9). Similarly, Ekya [12] only focuses on co-location of computation, and it's efficiency on finishing compute even on custom hardware is shown in Fig. 4.

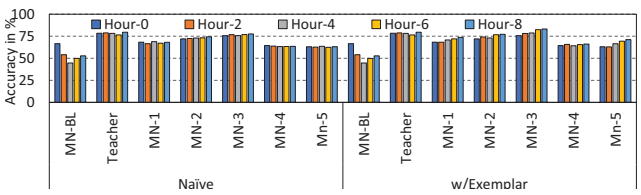


Fig. 7: Accuracy boost due to proper exemplar selection over 8 hours of time window. Labels: MN – MobileNet-V2, BL – Baseline, Teacher – the ensemble of teacher models, MN–#: targeted MobileNet-V2 model for the particular time of day.

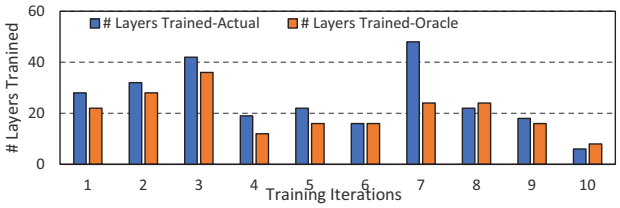
A. Continuous Learning: Accuracy

Fig. 7 shows the accuracy improvement over a time window of 8 hours by using the continuous learning algorithm. We compare against a baseline using naïve continuous learning algorithm with no representation learning. In contrast, *Ụṣás* uses a 2 level exemplar selection algorithm (one using the confidence matrix, and then further refined by the representation learning). We observe that, with representation learning, *Ụṣás* is ≈4.94% (maximum ≈8.03%, and minimum ≈2.62%) more accurate than the naïve learner. Further, *Ụṣás* converges closer to the accuracy of the teacher model. This was possible by restricting the training space and by using the superior exemplar set construction by using representation learning.

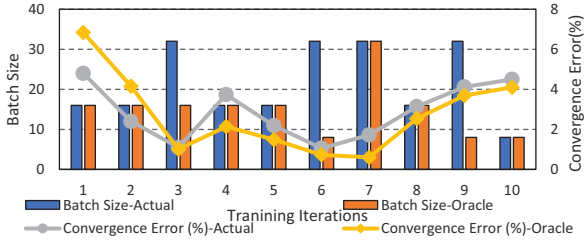
Fig. 8 shows the impact of micro-profiling on the hyper parameter selection. Due to the drift- and weighted accuracy-aware micro-profiler, the suggested configuration is almost every time the same as an oracular selection. Fig. 8a shows the number of layers trained for a DNN, in contrast to the ideal number of layers to achieve maximum accuracy. Over 10 training iterations, we observed the micro-profiler to be consistent with the oracle (except for one case of iteration 7). Note that the hyperparameter selected in iteration 7 by the micro-profiler performs as good as the the oracle model in terms of achieving accuracy, albeit by performing more computation. A deep dive into 7^{th} iteration reveals that the micro-profiler chose a higher learning rate (compared to the oracle), which biased the convergence curve fitting and extrapolation (as discussed in §III-C) and hence suggested a larger number of layers to be trained to achieve the required convergence. Similarly, the micro-profiler shows consistent behaviour while choosing the right number of batches. Fig. 8b also shows the error rate of retraining performed by choosing the hyperparameters given by the micro-profiler vs an oracle selection. Observation over 40 hours of continuous learning on the dataset suggest that the micro-profiler has, on average, an accuracy deviation of 2.46%, compared to an oracle parameter selection. Along with that, the micro-profiler selects correct batch size 82.64% of the time and the correct number of layers for 87.06% of the time.

B. Impact on Exemplar Selection

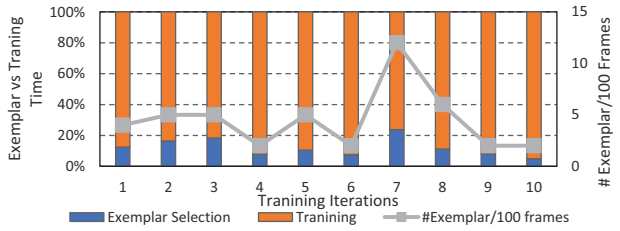
Ụṣás benefits from the use of *multiple* teacher models for data annotation and exemplar selection. Prior works on intermittent learning have either chosen one teacher model



(a) Number of layers trained.



(b) Batch-size and convergence.



(c) Exemplar selection w.r.t. training.

Fig. 8: Algorithmic performance of *Uṣás*: the benifits of the exemplar selection and μ-profiler.

Component	Spec	Power	Area(mm^2)
SRAM Buffers	1kB*256+8kB*256+64kB+16*256kB	10.372W	117.164
MAC Unit	(8*8)*256	8.46W	32.72
Adder Tree and Comparator	16*16bit + 256	2.4W	21.556
Control	–	0.96W	12.2
Host	∼Cortex A78 series	11W	–
Design at 592MHz with Synopsys AED 32nm library			
Total	256 tiles	33.192W	183.64

TABLE I: Area and power estimation of our design.

(e.g. Ekya [12] using ResNeXt101) to annotate the data or used a heuristic on top of the teacher model (e.g. intermittent learning [47] using randomized selection, K–Last List, or round-robin policy). As shown in Fig. 9, a single teacher, even with the augmented heuristics, typically fails to select the right exemplar set. The exemplar set significantly impacts the accuracy in two ways: 1. missing valid exemplars will result in the student model missing out in learning vital information, increasing its drift, and 2. a wrong annotation by the teacher can also result in the student learning wrong labels, resulting in increased mis-predictions. To avoid this, in *Uṣás*, the teacher models perform majority voting to decide the right exemplar, which significantly reduces false positives and true negatives (refer to the top bar in Fig. 9: with the ensemble, the best case annotation is the ideal one with only 2 false positives). Furthermore, the feature extraction for each of the potential exemplars for the teacher model is hardware-assisted (§V-C), and hence poses no overhead to the inference task.

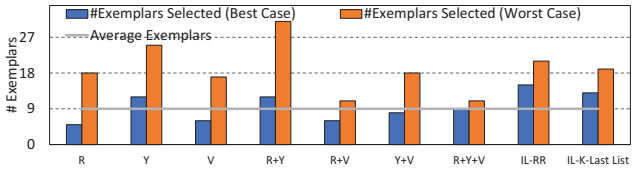


Fig. 9: Impact of multiple teachers on exemplar selection. X-axis shows #exemplars/100 inferred frames over a 2hr window. Having an ensemble provides robust exemplar selection and improves accuracy over a single teacher. The X-Axis has different DNNs , R: ResNeXt101, T: YOLO-V3, V: VGG-16, IL: Intermittent Learning, RR: Round Robin.

C. Hardware Implementation and Evaluation

The proposed morphable hardware was simulated using an in-house simulator based on ScaleSim [79]. We included a wrapper around ScaleSim to *dynamically* change the configuration of the systolic array. Further, the simulator was integrated with CACTI [62] and DRAMSIM3 [49] to estimate access latency, power, and simulate the memory access pattern. Rather than including a cycle accurate CPU (host) simulator to orchestrate the compute, we used a simple program to act as proxy for the host CPU and send control signals to schedule and orchestrate the compute on the systolic array. To correctly estimate accelerator power and area, we implemented a register-transfer level model using System Verilog and synthesized using Synopsys Design Compiler [93] with a 32nm library [94]. Table I lists the estimated power consumption and area of the major components. Instead of simulating the CPU, we tested the K-means clustering and cluster optimization on a mobile SoC with 8× ARM Cortex A78 series CPU. Table II gives the key attributes of the implemented hardware against some of the prior accelerators. Note that *Uṣás* hardware is not outperforming any of them as the goal was greater *scheduling flexibility* for power tracking rather than performance or area.

Platform	Freq. (MHz)	Area (mm^2)	Power (W)	Peak Thpt. (GOps)	Energy Eff. (GOps/W)
DaDianNao [16]	606	67.3	16.3	4964	304.54
CNVLUTIN [4]	606	70.1	17.4	4964	285.29
Activation Sparse [80]	667	292	19.2	5466	284.69
EyerissV2 [15]	200MHz	N/A	N/A	153.6G 8b fixed pt/s	193.7
FlexBlock [63]	333MHz	160.3 (65nm)	34.4 (when same #PEs)	4504	131.03
Uṣás	592	168.2	22.7 (17.2 if only train)	4016	159.42
	Fully powered, DNN Compute only				**287.44**
	Fully powered, DNN $\mu-$profiler				255.39
	EH + $\mu-$profile + NV-mems + resizing RAM + Host				159.40

TABLE II: Comparison with prior accelerator-based platforms.

The systolic array accelerator time multiplexes between performing feature extraction for exemplar selection and running the training. Fig. 8c shows the time distribution of the accelerator between performing exemplar selection and training. It also shows the number of exemplar frames per 100 frame, i.e., of any 100 frame encountered, how many of those will contain a relatively new data. Over 10 iterations of retraining,

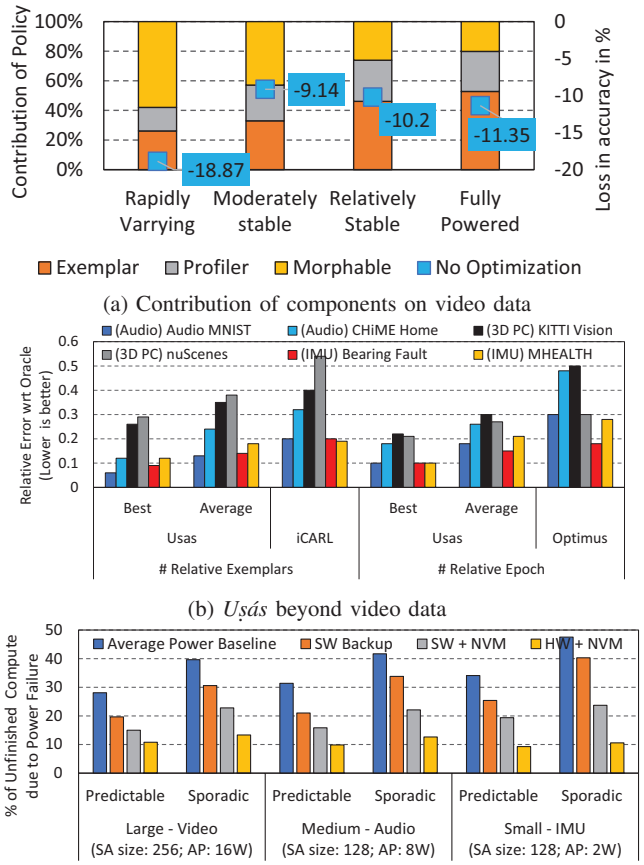


Fig. 10: Contribution of different components of *Usás* on other applications, data modalities, and power environments.

the learner classified ≈4.5 frames/100-frames (on an average) as exemplar data. And, over 40 hours of continuous learning, we get ≈5.02 frames/100-frames as exemplar data (resulting in ≈17.4% of total accelerator time).

Performance-Power Trade-offs: As Table II suggest, *Usás* does not deliver the highest throughput and also consumes more power compared to the other accelerators. *Usás* was designed on a intermittency friendly approach, and was never designed to hit the best throughput. The unit compute (only a 3×3 convolution per tile) that *Usás* can perform is much smaller than the other accelerators, limiting its throughput but increasing its modularity of handling intermittent power failures (or power changes). *Usás* also consumes more power than the other accelerators since it also performs the exemplar selection along with the DNN training, and also houses NV-SRAM buffers for hardware check-pointing. While fully powered, *Usás* is competitive in terms of energy efficiency for training-only tasks. We include details on the energy efficiency of *Usás* under different of operation configurations in Table II. Note that the energy inefficiency arises primarily from i) multiple saves and restores, ii) use of NV memories and iii) reconfiguring the DRAM (along with a commercial ARM based host CPU). Note that most prior works ignore memory and host overheads while reporting the throughput, efficiency and power numbers. Moreover, *Usás* needs more I/O operations to store the streaming data to compute the exemplars. We believe it will not be fair to compare the energy efficiency and throughput of a system like ours, which inherently has more memory, I/O and reconfiguration operation with a pure compute based systems mentioned in the hardware baseline. Further, we are the first of a kind system to imagine sustainability first and design a morphable hardware which can facilitate multiple functionality. We also compare our work against two reconfigurable platforms [15], [63].

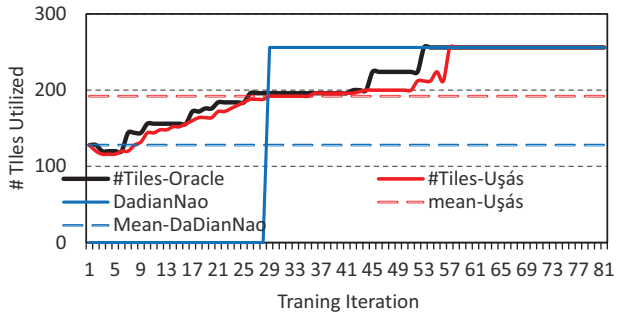
Power Aware Scaling: The *Usás* hardware's most important feature is its ability to *morph* according to power availability. Fig. 11 shows its ability to maximize the instantaneous power utilization and scale the number of tiles. This allows *Usás* to effectively perform more computation with an intermittent power source. As shown in Fig. 11b, *Usás* maintains a high duty cycle across power variance, whereas DaDianNao [16] could not be active for all the power cycles. Considering the power profile of Fig. 11b, *Usás* can finish about 50 cycles of retraining (50 complete training cycles) and DaDianNao can only finish 22 training cycles, even assuming a zero overhead, seamless save-restore of the partial computes of DaDianNao during a power failure/emergency.

Setup	Effective Training	Accuracy Degradation	Replacement Cycle*
Battery Backed Custom HW[5000mAH]	93.17	2.48	2 - 3 years
Battery Backed Mobile GPU	78.55	7.43	18 - 24 months
Fixed Power [15W]	67.54	12.6	NA
Fixed Power [35W]	100	1.87	
Usás	95.3	1.92	7 - 10 years

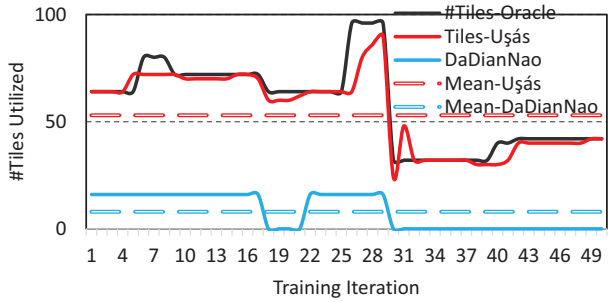
TABLE III: Comparing *Usás* with other possible solutions.

Sustainability: To ensure sustainable and continuous learning at the edge, *Usás* operates independently of the power grid or cloud dependency. We evaluated *Usás* against DaDianNao, a power-efficient DNN training accelerator, with some modifications for comparison. Using the Seattle SOLRAD power trace for January 1, 2022, we simulated 40 hours of continuous learning with 5 different models on Urban Traffic data [97] and *Usás* hardware. The results, summarized in Table IV, demonstrate the effectiveness of *Usás* in achieving continuous forward progress compared to other approaches. It completed more training tasks while consuming less power and minimizing wastage. In contrast, cloud-based solutions exhibited poor sustainability, relying on high-power-consuming GP-GPUs, and edge servers without power availability struggled to perform any compute. *Usás* emerges as a promising solution, effectively achieving sustainable and carbon-neutral continuous learning at the edge, addressing critical challenges related to power constraints and environmental impact.

Alternate Solutions: Although *Usás* works completely using intermittent power, it is imperative to compare and contrast it with other possible solutions. TABLE III depicts some of such possible comparison points. The possible alternate solutions being battery-backed custom HW [16], battery-backed commercial GPU and fixed power budget with store and



(a) Monotonically increasing



(b) Rapidly varying

Fig. 11: Tile utilization against available power: *Uṣás* with eager scheduling vs an oracle scheduler. *Uṣás* closely tracks oracle, where as DaDianNao [16] falls short.

execute (using a capacitor/ battery). We quantitatively compare the effective training, i.e. the ratio of number of scheduled training to the number of completed training, loss of accuracy compared to the baseline. It is clear that even with intermittent power availability *Uṣás* is objectively finishing more tasks (except compared to a system with a consistently high power availability). Furthermore, we also present a qualitative comparison on the maintenance cycle needed for these solutions. While a completely grid based solution is best in terms of reliability, it is not feasible because of the power demands. Any battery backed system will be limited to the charging cycle of the batteries (≈ 500 cycles for Li-ion batteries) which leads to a typical 18 to 24 months of life for such devices (compared to this, a super capacitor have a life of more than 100 years). We believe that the lifetime of *Uṣás* will be limited either by the live of the harvesting source ($\approx$ 20 – 30 years for solar, $\approx$ 10 – 12 years for portable wind turbines), or the training hardware (typical life cycle of embedded devices are of range of 7 – 10 years). We agree that a limitation of our work comes from the choice of solar energy: unavailability during night and bad weather makes the deployment harder. However, there has been significant recent development in portable wind turbines [96], which can be deployed on rooftops, can work with $\geq 5mph$ wind speed, and can provide power equivalent of 15 solar cells. Therefore, similar technologies can be used to augment the harvesting mechanism.

Deployment	Training Completed	Mean Power Consumed (W)	Mean Power Wasted (W)	Carbon Footprint (lbs/yr)
Uṣás	11	17.2	3.54	8.33
DaDianNao (persistent)	6	6.4	10.8	128.0712
DaDianNao (Software Only)	4	5.8	12.46	135.964
DaDianNao (actual)	2	2.09	24.73	199.70
Edge Cloud	0	0	–	
Cloud	1	200	0	2233.8
Max Power = 32W; Min Power = 12W; Training Scheduled = 12				

TABLE IV: Comparing *Uṣás* hardware with other state of the art offerings for both performance and sustainability.

D. Towards Other Applications and Domains

The morphable hardware design of *Uṣás* plays a crucial role in efficiently handling varying energy income and workloads. As the energy income becomes more sporadic, hardware-assisted scheduling seamlessly transfers work to active processing elements (PEs), maximizing the completion of tasks that could have otherwise been lost. This hardware-driven adaptive scheduling significantly impacts different data modalities, from large-scale to small-scale, and various magnitudes of energy income, as depicted in Fig. 10c. For scenarios with larger and predictable energy income, software-based backup and restore mechanisms can offer significant benefits, as the energy consumed for such operations is typically a small fraction of the overall energy income. Predictive actions for saving the system state can be easily taken. However, in situations with sporadic energy income, the hardware-assisted scheduling becomes paramount. It ensures that active PEs efficiently utilize available power to complete work, preventing potential losses and eliminating the need to restart tasks from the beginning. *Uṣás* excels as a candidate for continuous learning at all scales due to the hardware's adaptability to varying data and model sizes. As data and model dimensions decrease, the hardware assistance's impact becomes more pronounced, making *Uṣás* an excellent solution for continuous learning across diverse application sizes.

Along with morphable hardware, the exemplar selection and the micro-profiler play an important role for the success of *Uṣás*. When power is highly uncertain, the morphable hardware also strongly contributes, however, as the power profile becomes stable, the algorithmic contributions dominate. Fig. 10a shows the contribution of the different components of *Uṣás* under different power profiles. Moreover, the algorithmic contributions can be extended into any classification based application or data modality. If the learning has to be unsupervised, one needs to experiment with known clustering techniques to decide the right classification approach. We demonstrate this by testing the exemplar selection and the $\mu-$profiler with different modalities of data. Our workloads included Audio [23], [35](speech classification), 3D Point Clouds [14], [24](object classification) and Inertial Measurement Unit sensor data [9], [106](fault and activity detection). Observe that, as *Uṣás* is designed to handle dense and noisy data, it outperforms the respective state-of-the-arts (which were tuned for small, clean benchmark data).

VI. Discussion

Key insights: Compared to other systems, the ratio of energy requirement of task vs the harvested energy is much higher here. Added with time constraints, designing such a system becomes tricky. While many of the prior works have designed their systems around inference using intermittent systems, we are one of the few works which focuses on learning, and the only work which does it on a large scale of data. The proposed system is not only energy intermittent, but also memory intermittent, interconnect intermittent and most importantly data intermittent (we don't know how much data and what data). This gives us a unique platform to think of intermittency beyond embedded systems and energy.

Related Works: Although there has been significant research [40], [41], [47], [52], [56], [61], [72], [104] on enabling machine learning in intermittently powered devices, a majority of it focuses on performing inference. Only intermittent learning [47] focuses on performing on-device training, but with very small workloads and models. Considering the scale, scope and workload of our problem, limits direct comparisons, except for comparing their exemplar selection method. Similarly, Ekya [12] only focuses on co-location of computation, and it's efficiency on finishing compute even on a custom hardware is shown in Fig. 4.

Green Data Centers: As sustainability gains traction, industry has worked towards building green data centers [58], [59]. Although using these data centers for computation can be an alternative, it will not solve the bandwidth and the privacy issues mentioned in §I. Moreover, communicating and storing such high volume data will also require energy. Our solution decentralizes this massive compute using a sustainable approach and hence has its own merits. Further, this can help build future solutions using these decentralised nodes for other applications. We do encourage the use of green data centers for other centralized compute applications.

VII. Concluding Remarks

The growth of smart cities and urban mobility applications, along with reformations in privacy laws, have produced a need for pervasive, DNN based continuous learning at the edge. Although current commercial devices are capable of handling inference at the edge, the power and resource requirements of training make it impractical and unsustainable for all edge nodes to also perform continuous training off of grid power. In this work, we design *Uṣás*, a sustainable continuous learning platform, which can perform video analytics by using an intermittent power source like solar power. The learning algorithm of *Uṣás* delivers 4.96% more accurate classification compared to a naïve learner, and the morphable hardware design uses intermittent computing to maintain forward progress even while running on lower power budget. Together, *Uṣás* can save up to 200lbs of CO_2 per year compared to a state of the art accelerator running on the grid.

VIII. Acknowledgments

We would like to offer our thanks to the anonymous reviewers for their detailed feedback, which has greatly helped to improve and refine this paper. This work was supported in part by Semiconductor Research Corporation (SRC), Center for Brain-inspired Computing (C-BRIC) and NSF Grant #1822923 (SPX: SOPHIA). We acknowledge that all product names used are for identification purposes only and may be trademarks of their respective companies.

References

[1] S. M. Abhay S, "Autonomous vehicle market by level of automation," https://www.alliedmarketresearch.com/autonomous-vehicle-market, Feb 2022, (Accessed on 08/04/2023).

[2] "Achieving Compliant Data Residency and Security with Azure," https://azure.microsoft.com/mediahandler/files/resourcefiles/achieving-compliant-data-residency-and-security-with-azure/Achieving_Compliant_Data_Residency_and_Security_with_Azure.pdf.

[3] D. B. Agusdinata, W. Liu, H. Eakin, and H. Romero, "Socio-environmental impacts of lithium mineral extraction: towards a research agenda," *Environmental Research Letters*, vol. 13, no. 12, p. 123001, 2018.

[4] J. Albericio, P. Judd, T. Hetherington, T. Aamodt, N. E. Jerger, and A. Moshovos, "Cnvlutin: Ineffectual-neuron-free deep neural network computing," *ACM SIGARCH Computer Architecture News*, vol. 44, no. 3, pp. 1–13, 2016.

[5] W. Amit Katwala, "The spiralling environmental cost of our lithium battery addiction," https://www.wired.co.uk/article/lithium-batteries-environment-impact, May 2018, (Accessed on 07/08/2023).

[6] G. Ananthanarayanan, V. Bahl, P. Bodík, K. Chintalapudi, M. Philipose, L. R. Sivalingam, and S. Sinha, "Real-time Video Analytics – the killer app for edge computing," *IEEE Computer*, 2017.

[7] AWS Outposts, "https://aws.amazon.com/outposts/rack/hardware-specs/?nc=sn&loc=4."

[8] Azure Stack Edge, "https://azure.microsoft.com/en-us/services/databox/edge/."

[9] O. Banos, C. Villalonga, R. García, A. Saez, M. Damas, J. Holgado-Terriza, S. Lee, H. Pomares, and I. Rojas, "Design, implementation and validation of a novel open framework for agile development of mobile health applications," *BioMedical Engineering OnLine*, 2015.

[10] S. Bauer, "Explainer: The opportunities and challenges of the lithium industry," *Diálogo Chino*, 2020.

[11] Beverly Hills has thousands of surveillance cameras, "https://bit.ly/BeverlyHillsCamera."

[12] R. Bhardwaj, Z. Xia, G. Ananthanarayanan, J. Jiang, Y. Shu, N. Karianakis, K. Hsieh, P. Bahl, and I. Stoica, "Ekya: Continuous learning of video analytics models on edge compute servers," in *19th USENIX Symposium on Networked Systems Design and Implementation (NSDI 22)*, 2022, pp. 119–135.

[13] R. Bird, Z. J. Baum, X. Yu, and J. Ma, "The regulatory environment for lithium-ion battery recycling," 2022.

[14] H. Caesar, V. Bankiti, A. H. Lang, S. Vora, V. E. Liong, Q. Xu, A. Krishnan, Y. Pan, G. Baldan, and O. Beijbom, "nuscenes: A multimodal dataset for autonomous driving," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 11 621–11 631.

[15] Y.-H. Chen, T.-J. Yang, J. Emer, and V. Sze, "Eyeriss v2: A flexible accelerator for emerging deep neural networks on mobile devices," *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, vol. 9, no. 2, pp. 292–308, 2019.

[16] Y. Chen, T. Luo, S. Liu, S. Zhang, L. He, J. Wang, L. Li, T. Chen, Z. Xu, N. Sun *et al.*, "Dadiannao: A machine-learning supercomputer," in *2014 47th Annual IEEE/ACM International Symposium on Microarchitecture*. IEEE, 2014, pp. 609–622.

[17] P. Chi, S. Li, Y. Cheng, Y. Lu, S. H. Kang, and Y. Xie, "Architecture design with stt-ram: Opportunities and challenges," in *2016 21st Asia and South Pacific design automation conference (ASP-DAC)*. IEEE, 2016, pp. 109–114.

[18] A. F. CNBC, "How traffic sensors and cameras are transforming city streets," https://www.cnbc.com/2021/02/22/how-traffic-sensors-and-cameras-are-transforming-city-streets.html, (Accessed on 04/28/2023).

[19] Coral.ai, "Edge tpu performance benchmarks = https://coral.ai/docs/edgetpu/benchmarks/," (Accessed on 11/21/2022).

[20] D Maltoni, V Lomonaco, "Continuous learning in single-incremental-task scenarios," in *Neural Networks*, 2019.

[21] J. Dodge, T. Prewitt, R. Tachet des Combes, E. Odmark, R. Schwartz, E. Strubell, A. S. Luccioni, N. A. Smith, N. DeCario, and W. Buchanan, "Measuring the carbon intensity of ai in cloud instances," in *2022 ACM Conference on Fairness, Accountability, and Transparency*, 2022, pp. 1877–1894.

[22] E. Dong, Y. Zhu, Y. Ji, and S. Du, "An improved convolution neural network for object detection using yolov2," in *2018 IEEE International Conference on Mechatronics and Automation (ICMA)*. IEEE, 2018, pp. 1184–1188.

[23] P. Foster, S. Sigtia, S. Krstulovic, J. Barker, and M. D. Plumbley, "Chime-home: A dataset for sound source recognition in a domestic environment," in *2015 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*. IEEE, 2015, pp. 1–5.

[24] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? the kitti vision benchmark suite," in *2012 IEEE conference on computer vision and pattern recognition*. IEEE, 2012, pp. 3354–3361.

[25] N. Global Monitoring Laboratory, "Solrad network," https://gml.noaa.gov/grad/solrad/index.html, (Accessed on 11/21/2022).

[26] G. Gobieski, B. Lucia, and N. Beckmann, "Intelligence beyond the edge: Inference on intermittent embedded systems," in *ASPLOS*. ACM, 2019.

[27] Z. Gong, H. Ji, C. W. Fletcher, C. J. Hughes, and J. Torrellas, "Sparsetrain: Leveraging dynamic sparsity in software for training dnns on general-purpose simd processors," in *Proceedings of the ACM International Conference on Parallel Architectures and Compilation Techniques*, 2020, pp. 279–292.

[28] J. R. Gunasekaran, C. S. Mishra, P. Thinakaran, B. Sharma, M. T. Kandemir, and C. R. Das, "Cocktail: A multidimensional optimization for model serving in cloud," in *19th USENIX Symposium on Networked Systems Design and Implementation (NSDI 22)*, 2022, pp. 1041–1057.

[29] U. Gupta, Y. G. Kim, S. Lee, J. Tse, H.-H. S. Lee, G.-Y. Wei, D. Brooks, and C.-J. Wu, "Chasing carbon: The elusive environmental footprint of computing," *IEEE Micro*, vol. 42, no. 4, pp. 37–47, 2022.

[30] S. Han, H. Mao, and W. J. Dally, "Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding," *arXiv preprint arXiv:1510.00149*, 2015.

[31] J. He and F. Zhu, "Online continual learning for visual food classification," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 2337–2346.

[32] K. He, X. Zhang, S. Ren, and J. Sun, "Identity mappings in deep residual networks," in *European conference on computer vision*. Springer, 2016, pp. 630–645.

[33] IBM, "Data labeling," https://www.ibm.com/cloud/learn/data-labeling, (Accessed on 11/21/2022).

[34] I. Ilievski, T. Akhtar, J. Feng, and C. Shoemaker, "Efficient hyperparameter optimization for deep learning algorithms using deterministic rbf surrogates," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 31, no. 1, 2017.

[35] Z. Jackson, "Free spoken digit dataset (fsdd)," https://github.com/Jakobovski/free-spoken-digit-dataset, July 2022, (Accessed on 07/08/2023).

[36] C. Jones and J. D. Ryan, *Encyclopedia of hinduism*. Infobase publishing, 2006.

[37] N. P. Jouppi, C. Young, N. Patil, D. Patterson, G. Agrawal, R. Bajwa, S. Bates, S. Bhatia, N. Boden, A. Borchers *et al.*, "In-datacenter performance analysis of a tensor processing unit," in *Proceedings of the 44th annual international symposium on computer architecture*, 2017, pp. 1–12.

[38] Junchen Jiang, Ganesh Ananthanarayanan, Peter Bodík, Siddhartha Sen, Ion Stoica, "Chameleon: Scalable adaptation of video analytics," in *ACM SIGCOMM*, 2018.

[39] Junjue Wang, Ziqiang Feng, Shilpa George, Roger Iyengar, Pillai Padmanabhan, Mahadev Satyanarayanan, "Towards scalable edge-native applications," in *ACM/IEEE Symposium on Edge Computing*, 2019.

[40] C.-K. Kang, H. R. Mendis, C.-H. Lin, M.-S. Chen, and P.-C. Hsiu, "Everything leaves footprints: Hardware accelerated intermittent deep inference," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 39, no. 11, pp. 3479–3491, 2020.

[41] C.-K. Kang, H. R. Mendis, C.-H. Lin, M.-S. Chen, and P.-C. Hsiu, "More is less: Model augmentation for intermittent deep inference," *ACM Transactions on Embedded Computing Systems (TECS)*, vol. 21, no. 5, pp. 1–26, 2022.

[42] D. Kang, J. Emmons, F. Abuzaid, P. Bailis, and M. Zaharia, "Noscope: Optimizing deep cnn-based queries over video streams at scale." *Proc. VLDB Endow.*, vol. 10, no. 11, pp. 1586–1597, 2017.

[43] Keras, "Keras applications," https://keras.io/api/applications/, (Accessed on 11/21/2022).

[44] Konstantin Shmelkov, Cordelia Schmid, Karteek Alahari , "Incremental learning of object detectors without catastrophic forgetting," in *ICCV*, 2017.

[45] G. Kordopatis-Zilos, S. Papadopoulos, I. Patras, and I. Kompatsiaris, "Visil: Fine-grained spatio-temporal video similarity learning," in *Proceedings of the IEEE/CVF international conference on computer vision*, 2019, pp. 6351–6360.

[46] S. Lee, S. Goldt, and A. Saxe, "Continual learning in the teacher-student setup: Impact of task similarity," in *International Conference on Machine Learning*. PMLR, 2021, pp. 6109–6119.

[47] S. Lee, B. Islam, Y. Luo, and S. Nirjon, "Intermittent learning: On-device machine learning on intermittently powered system," *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, vol. 3, no. 4, pp. 1–30, 2019.

[48] L. Li, K. Jamieson, G. DeSalvo, A. Rostamizadeh, and A. Talwalkar, "Hyperband: A novel bandit-based approach to hyperparameter optimization," *The Journal of Machine Learning Research*, vol. 18, no. 1, pp. 6765–6816, 2017.

[49] S. Li, Z. Yang, D. Reddy, A. Srivastava, and B. Jacob, "Dramsim3: a cycle-accurate, thermal-capable dram simulator," *IEEE Computer Architecture Letters*, vol. 19, no. 2, pp. 106–109, 2020.

[50] Z. Li and D. Hoiem, "Learning without forgetting," *IEEE transactions on pattern analysis and machine intelligence*, vol. 40, no. 12, pp. 2935–2947, 2017.

[51] M Sandler, A Howard, Menglong Zhu, Andrey Zhmoginov, Liang-Chieh Chen , "Mobilenetv2: Inverted residuals and linear bottlenecks," in *CVPR*, 2018.

[52] K. Maeng and B. Lucia, "Adaptive dynamic checkpointing for safe efficient intermittent computing," in *OSDI*. USENIX Association, 2018.

[53] B. Makuza, Q. Tian, X. Guo, K. Chattopadhyay, and D. Yu, "Pyrometallurgical options for recycling spent lithium-ion batteries: A comprehensive review," *Journal of Power Sources*, vol. 491, p. 229622, 2021.

[54] Markets and Markets, "Smart cities market analysis, industry size and forecast," https://www.marketsandmarkets.com/Market-Reports/smart-cities-market-542.html#:~:text=The%20global%20Smart%20Cities%20Market,drive%20the%20smart%20cities%20market, 2022, (Accessed on 08/04/2023).

[55] M. McCloskey and N. J. Cohen, "Catastrophic interference in connectionist networks: The sequential learning problem," in *Psychology of learning and motivation*. Elsevier, 1989, vol. 24, pp. 109–165.

[56] H. R. Mendis, C.-K. Kang, and P.-c. Hsiu, "Intermittent-aware neural architecture search," *ACM Transactions on Embedded Computing Systems (TECS)*, vol. 20, no. 5s, pp. 1–27, 2021.

[57] Meta, "Sharing our progress on combating climate change = https://about.fb.com/news/2022/11/metas-progress-on-combating-climate-change/," (Accessed on 11/21/2022).

[58] Meta, "Sustainability: Data centers," https://sustainability.fb.com/data-centers/, (Accessed on 04/28/2023).

[59] Microsoft, "Building world-class sustainable datacenters and investing in solar power in arizona," https://blogs.microsoft.com/on-the-issues/2019/07/30/building-world-class-sustainable-datacenters-and-investing-in-solar-power-in-arizona/, (Accessed on 04/28/2023).

[60] Microsoft-Rocket-Video-Analytics-Platform, "https://github.com/microsoft/Microsoft-Rocket-Video-Analytics-Platform."

[61] C. S. Mishra, J. Sampson, M. T. Kandemir, and V. Narayanan, "Origin: Enabling on-device intelligence for human activity recognition using energy harvesting wireless sensor networks," in *DATE*, 2021.

[62] N. Muralimanohar, R. Balasubramonian, and N. P. Jouppi, "Cacti 6.0: A tool to model large caches," *HP laboratories*, vol. 27, p. 28, 2009.

[63] S.-H. Noh, J. Koo, S. Lee, J. Park, and J. Kung, "Flexblock: A flexible dnn training accelerator with multi-mode block floating point support," *IEEE Transactions on Computers*, 2023.

[64] T. N. R. E. L. (NREL), "Solar resource maps and data," https://www.nrel.gov/gis/solar-resource-maps.html, (Accessed on 11/21/2022).

[65] NVIDIA T4 for Virtualization, "http://bit.ly/3EfiuWg."

[66] I. of Energy Research, "The environmental impact of lithium batteries," https://www.instituteforenergyresearch.org/renewable/the-environmental-impact-of-lithium-batteries/, November 2020, (Accessed on 07/08/2023).

[67] D. Patterson, J. Gonzalez, Q. Le, C. Liang, L.-M. Munguia, D. Rothchild, D. So, M. Texier, and J. Dean, "Carbon emissions and large neural network training," *arXiv preprint arXiv:2104.10350*, 2021.

[68] Y. Peng, Y. Bao, Y. Chen, C. Wu, and C. Guo, "Optimus: an efficient dynamic resource scheduler for deep learning clusters," in *Proceedings of the Thirteenth EuroSys Conference*, 2018, pp. 1–14.

[69] J. F. Peters, M. Baumann, B. Zimmermann, J. Braun, and M. Weil, "The environmental impact of li-ion batteries and the role of key parameters–a review," *Renewable and Sustainable Energy Reviews*, vol. 67, pp. 491–506, 2017.

[70] A. Prabhu, C. Dognin, and M. Singh, "Sampling bias in deep active classification: An empirical study," *arXiv preprint arXiv:1909.09389*, 2019.

[71] I. X. G. Processors, "Rankings about energy in the world," https://www.intel.com/content/www/us/en/products/details/processors/xeon/scalable/gold/products.html, (Accessed on 11/21/2022).

[72] K. Qiu, N. Jao, M. Zhao, C. S. Mishra, G. Gudukbay, S. Jose, J. Sampson, M. T. Kandemir, and V. Narayanan, "Resirca: A resilient energy harvesting reram crossbar-based accelerator for intelligent embedded processors," in *2020 IEEE International Symposium on High Performance Computer Architecture (HPCA)*. IEEE, 2020, pp. 315–327.

[73] Ravi Teja Mullapudi, Steven Chen, Keyi Zhang, Deva Ramanan, Kayvon Fatahalian, "Online model distillation for efficient video inference," in *ICCV*, 2019.

[74] S.-A. Rebuffi, A. Kolesnikov, G. Sperl, and C. H. Lampert, "icarl: Incremental classifier and representation learning," in *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, 2017, pp. 2001–2010.

[75] J. Redmon and A. Farhadi, "Yolov3: An incremental improvement," *arXiv preprint arXiv:1804.02767*, 2018.

[76] G. V. Research, "Consumer iot market size, share and trends analysis report by component (hardware, services), by connectivity technology (wired, wireless), by application (healthcare, wearable devices), and segment forecasts, 2023 - 2030," https://www.grandviewresearch.com/industry-analysis/consumer-iot-market-report#:~:text=The%20global%20consumer%20IoT%20market,advanced%20devices%20and%20home%20appliances., July 2022, (Accessed on 08/04/2023).

[77] M. S. rutgers.edu, "Support for traffic cameras increases if used as a tool to limit interactions with police," https://www.rutgers.edu/news/support-traffic-cameras-increases-if-used-tool-limit-interactions-police, (Accessed on 04/28/2023).

[78] J. Salamon, C. Jacoby, and J. P. Bello, "A dataset and taxonomy for urban sound research," in *22nd ACM International Conference on Multimedia (ACM-MM'14)*, Orlando, FL, USA, Nov. 2014, pp. 1041–1044.

[79] A. Samajdar, Y. Zhu, P. Whatmough, M. Mattina, and T. Krishna, "Scale-sim: Systolic cnn accelerator simulator," *arXiv preprint arXiv:1811.02883*, 2018.

[80] A. Sarma, S. Singh, H. Jiang, A. Pattnaik, A. K. Mishra, V. Narayanan, M. T. Kandemir, and C. R. Das, "Exploiting activation based gradient output sparsity to accelerate backpropagation in cnns," *arXiv preprint arXiv:2109.07710*, 2021.

[81] A. Sarma, S. Singh, H. Jiang, A. Pattnaik, A. K. Mishra, V. Narayanan, M. T. Kandemir, and C. R. Das, "Exploiting activation based gradient output sparsity to accelerate backpropagation in cnns," *arXiv preprint arXiv:2109.07710*, 2021.

[82] scale.com, "Data labeling: The authoritative guide," https://scale.com/guides/data-labeling-annotation-guide#data-labeling-for-computer-vision, (Accessed on 11/21/2022).

[83] A. W. Services, "Aws outposts rack pricing," https://aws.amazon.com/outposts/rack/pricing/, (Accessed on 11/21/2022).

[84] K. Seyerlehner, G. Widmer, and P. Knees, "Frame level audio similarity-a codebook approach," in *Proc. of the 11th Int. Conf. on Digital Audio Effects (DAFx-08)*, 2008, p. 31.

[85] Shadi Noghabi, Landon Cox, Sharad Agarwal, Ganesh Ananthanarayanan, "The emerging landscape of edge-computing," in *ACM SIGMOBILE GetMobile*, 2020.

[86] Si Young Jang, Yoonhyung Lee, Byoungheon Shin, Dongman Lee, Dionisio Vendrell Jacinto , "Application-aware iot camera virtualization for video analytics edge computing," in *ACM/IEEE SEC*, 2018.

[87] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[88] snorkel.ai, "Making automated data labeling a reality in modern ai," https://snorkel.ai/automated-data-labeling/, (Accessed on 11/21/2022).

[89] E. Strubell, A. Ganesh, and A. McCallum, "Energy and policy considerations for deep learning in nlp," *arXiv preprint arXiv:1906.02243*, 2019.

[90] J. S.-M. studyfinds.org, "Traffic cameras become more popular when they cut down on interactions with police," https://studyfinds.org/traffic-cameras-interactions-police/, (Accessed on 04/28/2023).

[91] N. A. W. . Sun, "Us solar insolation maps," https://www.solar-electric.com/learning-center/solar-insolation-maps.html/#Map1, (Accessed on 11/21/2022).

[92] "Surveillance camera statistics: which cities have the most cctv cameras?" https://www.comparitech.com/vpn-privacy/the-worlds-most-surveilled-cities/, (Accessed on 11/21/2022).

[93] Synopsys, "Design compiler," https://www.synopsys.com/implementation-and-signoff/rtl-synthesis-test/dc-ultra.html, (Accessed on 04/28/2023).

[94] Synopsys, "Standard cell libraries," https://www.synopsys.com/dw/ipdir.php?ds=dwc_standard_cell, (Accessed on 04/28/2023).

[95] E. Talpes, D. Williams, and D. D. Sarma, "Dojo: The microarchitecture of tesla's exa-scale computer," in *2022 IEEE Hot Chips 34 Symposium (HCS)*. IEEE Computer Society, 2022, pp. 1–28.

[96] A. Technologies, "Aeromine technologies," https://www.aerominetechnologies.com/, (Accessed on 04/28/2023).

[97] Urban Traffic Dataset, "https://github.com/edge-video-services/ekya#urban-traffic-dataset."

[98] O. Wayman, "How urban mobility will change by 2030," https://www.oliverwyman.com/our-expertise/insights/2022/jun/how-urban-mobility-will-change-by-2030.html, 2022, (Accessed on 08/04/2023).

[99] P. with Code, "Object detection on coco test-dev," https://paperswithcode.com/sota/object-detection-on-coco, (Accessed on 11/21/2022).

[100] www.dlapiperdataprotection.com, "Sweden data collection & processing," https://www.dlapiperdataprotection.com/index.html?t=collection-and-processing&c=SE, (Accessed on 11/21/2022).

[101] J. Xu and X. Wang, "Rethinking self-supervised correspondence learning: A video frame-level similarity perspective," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 10 075–10 085.

[102] T.-J. Yang, Y.-H. Chen, and V. Sze, "Designing energy-efficient convolutional neural networks using energy-aware pruning," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 5687–5695.

[103] T.-J. Yang, A. Howard, B. Chen, X. Zhang, A. Go, M. Sandler, V. Sze, and H. Adam, "Netadapt: Platform-aware neural network adaptation for mobile applications," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 285–300.

[104] C.-H. Yen, H. R. Mendis, T.-W. Kuo, and P.-C. Hsiu, "Stateful neural networks for intermittent systems," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 41, no. 11, pp. 4229–4240, 2022.

[105] Z. Ying, S. Zhao, H. Zhang, C. S. Mishra, S. Bhuyan, M. T. Kandemir, A. Sivasubramaniam, and C. R. Das, "Exploiting frame similarity for efficient inference on edge devices," in *2022 IEEE 42nd International Conference on Distributed Computing Systems (ICDCS)*. IEEE, 2022, pp. 1073–1084.

[106] R. Zhang, H. Tao, L. Wu, and Y. Guan, "Transfer learning with neural networks for bearing fault diagnosis in changing working conditions," *Ieee Access*, vol. 5, pp. 14 347–14 357, 2017.

[107] S. Zhao, H. Zhang, S. Bhuyan, C. S. Mishra, Z. Ying, M. T. Kandemir, A. Sivasubramaniam, and C. R. Das, "Déja view: Spatio-temporal compute reuse for 'energy-efficient 360 vr video streaming," in *2020 ACM/IEEE 47th Annual International Symposium on Computer Architecture (ISCA)*. IEEE, 2020, pp. 241–253.

[108] S. Zhao, H. Zhang, C. S. Mishra, S. Bhuyan, Z. Ying, M. T. Kandemir, A. Sivasubramaniam, and C. Das, "Holoar: On-the-fly optimization of 3d holographic processing for augmented reality," in *MICRO-54: 54th Annual IEEE/ACM International Symposium on Microarchitecture*, 2021, pp. 494–506.

[109] B. Zoph, V. Vasudevan, J. Shlens, and Q. V. Le, "Learning transferable architectures for scalable image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 8697–8710.