Failure-Resilient ML Inference at the Edge through Graceful Service Degradation

Walid A. Hanafy[1], Li Wu[1], Tarek Abdelzaher[2], Suhas Diggavi[3], and Prashant Shenoy[1]

[1]University of Massachusetts Amherst
[2]University of Illinois Urbana Champaign
[3]University of California Los Angeles

***Abstract*—With recent innovations in machine learning (ML) technologies, especially deep learning, many IoT applications have increasingly relied on ML models for various tasks, such as classification, detection, and decision-making. Most of these tasks are latency-sensitive and depend on models deployed at the edge of the network. Network and edge devices are prone to various kinds of failures, such as transient, crash, or Byzantine failures. Such failures can impact the IoT device's ability to offload tasks, affecting the system's reliability. A traditional solution involves replicating the underlying resources and deploying a failover replica of the ML model. However, edge resources are typically limited, and increasing their size incurs significant computational and infrastructure cost overheads.**

This paper proposes a range of failover strategies for resource-constrained edge environments, leveraging the flexibility offered by ML models. We explore various approaches for graceful service degradation, such as degraded accuracy, latency, and sampling rate, and highlight their potential benefits and trade-offs. Furthermore, we discuss the challenges associated with these techniques and outline future directions.

***Index Terms*—Edge computing, ML inference, Resilience, Replication, Graceful degradation**

I. Introduction

Machine learning (ML) models, especially deep neural networks (DNNs), have significantly improved the performance of IoT applications across various tasks, such as classification, detection, and decision-making. However, despite advances in the architecture and design of ML models, they still require significant processing capacity and power, which is typically not available in IoT devices. To overcome this limitation, computational tasks are offloaded to nearby edge sites that offer higher computing capacity.

Figure 1 shows a typical edge architecture where sensors connect to edge sites and offload their data to a single or a pipeline of ML models. These models are placed on edge sites in a resource-aware manner while aiming to achieve minimal latency or meet the required service level objective (SLO) [1], [2]. However, failures, such as transient, crash, and Byzantine failures, are more frequently observed at the edge due to the increasing system complexity and growing environmental uncertainties. Especially in the adversarial environment of IoBTs, disruptions in communication, attacks on battlefield intelligent devices, and injections of malicious inputs are possible [3]. These failures hinder model availability and can have significant implications. As shown in Figure 1, a motion sensor might fail to detect passengers in a restricted area if the edge site it runs on crashes.

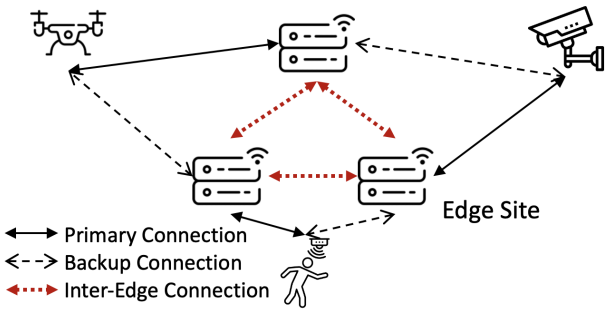


Fig. 1: Edge Architecture.

Designing a failure-resilient system can be challenging, especially for edge sites that are typically resource-constrained. Traditional solutions involve introducing redundancy in hardware and placed ML models. In this case, each edge server will have one or more failover backups that are activated in case of failures. However, this can be difficult to implement at edge sites with limited space, power, and resources. Edge sites typically span only a few servers and accelerators, making it infeasible to replicate the number of ML models. Additionally, increasing the number of servers or edge sites is not always viable due to significant cost increases and feasibility constraints.

Deep learning models are known for their design and runtime trade-offs between latency, resource requirements, and accuracy. Researchers have pointed out that picking the right-sized DNN model [4], [5], and compression techniques, such as quantization and pruning [6], [7], can significantly decrease resource requirements and latency with a minor reduction in accuracy. This flexibility has encouraged researchers to design systems that adjust their execution and model selection to handle various workload dynamics by trading latency requirement guarantees for a minor accuracy loss [5], [8]–[10]. We ask the following question: *How can this flexibility be leveraged to enable failure-resilient execution in resource-constrained edge environments?*

This paper addresses the issue of failure-resilient ML inference at resource-constrained edge environments by leveraging the flexibility offered by ML models. We explore a range of strategies for ensuring the resilient execution of the ML

inference tasks, which involve introducing a failover replica with a degraded quality of service (known as *graceful service degradation* [11]). For instance, in the event of an edge node crash, the affected ML models would fail over to their backup counterparts. Still, they would operate with reduced accuracy, a lower sampling rate, or increased latency. Furthermore, we describe the key system components for failure-resilient ML inference incorporating these graceful service degradation techniques. Lastly, we present the preliminary results of replicating ML models with the degradation techniques and outline the challenges and future directions. In summary, our contributions are threefold:

- We define the notion of graceful service degradation for resilient ML inference, encompassing accuracy, latency, and sampling rate degradation, and explore key mechanisms and their trade-offs (Section III).
- We describe key system components for failure-resilient ML inference that incorporates the explored graceful service degradation techniques (Section IV), and demonstrate the practicality of graceful service degradation (Section V).
- We discuss the challenges associated with deploying such techniques and outline future directions (Section VI).

II. Background

This section provides background information on edge inference, resilient execution via failover backups, and graceful degradation of ML inference services.

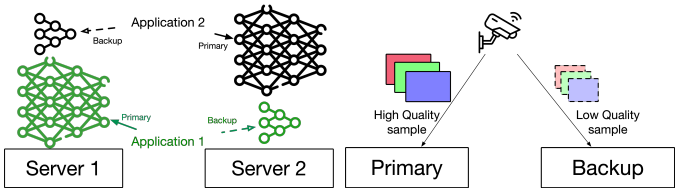
A. Edge ML Inference

Edge computing enables the deployment of computing and storage resources at the network's edge to provide users with low-latency services. Initially, edge computing focused on storage services, such as CDNs, which serve frequently used data, reduce latency, and conserve network bandwidth. More recently, edge computing has expanded to provide computing services [12], particularly to support resource-limited IoT sensors. Edge computing also plays a crucial role in deploying sensors in remote areas and for privacy-sensitive services where public cloud deployments are not feasible.

The overall edge offloading procedure can be described as follows: An IoT sensor sends input data x_i to an edge facility for a specific service, and the service responds with output data y_i, where $y_i = f(x_i)$. For ML applications, the function $f()$ may represent a single model inference or a pipeline of different ML models typically placed to minimize latency [1], [13]. For example, as shown in Figure 1, a camera sends its current view to the nearest edge site. The edge site uses advanced machine learning algorithms to identify and classify objects before sending the results back to the sensor.

B. Failover Replication

Resilient execution focuses on ensuring service availability in the event of faults and aims to minimize metrics such as downtime or mean time to recovery (MTTR). One key technique for implementing resilient execution is introducing a failover replica, where one server acts as the primary and others as backups. Clients send requests to the backup node if the primary fails or becomes inaccessible [14]. However, implementing failover replication can be challenging in edge computing due to several constraints. Edge resources are typically limited, making it challenging to deploy full-fledged replicas. Moreover, adding extra servers to edge sites may not be feasible due to space and power constraints. Even when adding additional servers is possible, it can significantly increase costs. The next section introduces strategies for graceful service degradation in failover replicas as a solution for resource-constrained environments.



(a) Heterogeneous replication (b) Communication flexibility

Fig. 2: Accuracy degradation mechanisms.

TABLE I: Speedup and memory savings of pruned and quantized models.

Model	DataSet	Pruning%	Speedup	Mem. Saving	Accuracy loss
Yolov 8 (Large)	COCO	85%	2.6×	11.4×	2.4 mAP@0.5
ResNet 50	ImageNet	90%	4.2×	8.35×	0.1%
oBERTa (Medium)	SST-2	90%	6.7×	8.6×	2%

C. Graceful Degradation

Graceful degradation is a commonly used technique in scenarios with resource contention. For example, internet applications may lower streaming quality or use progressive JPEG images to serve clients with poor network conditions. Graceful degradation is also employed to handle workload dynamics, such as sudden demand spikes. To prevent higher-order failures that are challenging to recover from, the system can gracefully reduce the available, less-critical features or quality of service for a subset of users [15]. Additionally, graceful degradation is a common technique for achieving fault tolerance, allowing a system to continue operating, typically at a reduced quality of service, even when experiencing a partial failure. For instance, networks often have a primary high-speed route and a slower backup if the primary route becomes unreliable [11].

III. ML Inference Graceful Degradation

The definition of graceful degradation can vary depending on the application. In ML inference applications, we define three primary types of degradation to maintain operation in the event of a failure, namely: *1) Accuracy degradation* where the failover replica provides less accuracy than the primary replica, *2) Latency degradation* where the failover replica is executed at a higher latency SLO, and *3) Sampling rate reduction* where a failover replica accepts a lower sampling rate than the primary one. In this section, we explore key mechanisms and tradeoffs for each type of degradation.

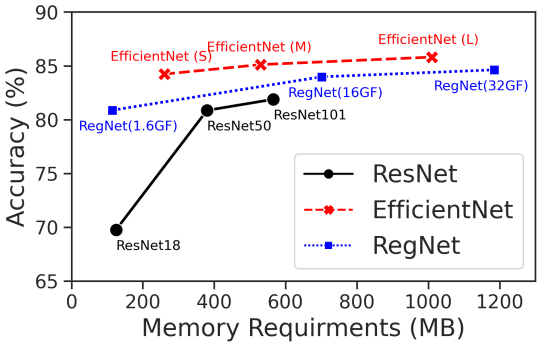


Fig. 3: Memory and Accuracy tradeoffs across DNN families.

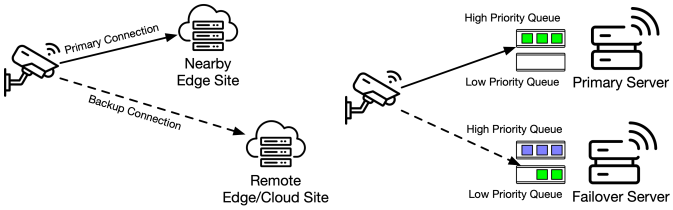
A. Accuracy Degradation

Figure 2 depicts key mechanisms to achieve accuracy degradation by running a smaller/compressed model trained on the same task, which we denote as heterogeneous replication, or by adjusting the sample quality, which we denote as communication flexibility. As shown in Figure 2a, a server can act as a failover replica for a peer node by deploying a compressed or smaller version of the initial model. The failover model can be selected by training smaller models or applying post-training methods such as pruning and quantization [5]–[7].

Figure 3 shows the tradeoff between memory requirements and the accuracy of different models trained on the ImageNet classification from Pytroch model library[1]. The figure shows that deploying a small failover backup that is an order of magnitude smaller can come at the cost of a few percent accuracy loss. Compression techniques can also decrease the computational requirements by an order of magnitude for limited accuracy loss. Table I shows the speedup and memory savings, as well as the accuracy loss of different ML models collected from sparsezoo[2] and executed using the DeepSparse machine learning framework on a C6i.12xlarge AWS instance. As shown, pruning and quantization can speed up ML models by 6.7× and reduce the memory requirement to 11.4× with minor accuracy loss.

In addition to the limitations imposed by the resource capacity of edge sites, the bandwidth of the edge network can also become a bottleneck due to network failures. In such case, as shown in Figure 2b, the client might only be allowed to send low-quality or highly compressed samples. Fortunately, ML models have shown robustness against different compression methods where they tolerate information loss with minor accuracy loss [16], [17]. For example, the authors of [16] showed that compression and channel selection techniques can reduce the transmitted data by 2.4× without accuracy reduction and by 7× for 0.03 mAP loss.

Key Insight: Utilizing less accurate failover techniques with low resource requirements, can increase the reliability of ML inference systems without the need for additional resources.



(a) Distant failover replica (b) Low-priority failover replica

Fig. 4: Latency degradation mechanisms.

B. Latency Degradation

Another degradation dimension is the application latency, where the latency requirements are relaxed for a subset of the affected applications due to failures, e.g., non-critical machine learning tasks. Figure 4 highlights key techniques to ensure resilient execution in cases of failure. Figure 4a shows a scenario where the camera feed is offloaded to the closest edge site. The camera also connects to an edge site further away, serving as a failover backup. The scheme has another benefit of reduced correlation between failures at distant sites, thereby enhancing overall resiliency. Figure 4b depicts another way of degrading the latency by deploying failover replicas at a low priority. This allows the system to harvest available free processing cycles with negligible effects on high-priority tasks [2].

Key Insight: Relaxing the latency of low criticality tasks can increase the reliability of ML inference systems without the need for additional resources.

C. Sampling Rate Degradation

The sampling rate determines the ML system quality and its ability to promptly detect incidents or uncommon events. Typically, the sampling rate is adjusted dynamically based on the current situation or operation objectives. It defines the system utilization and expected latency, whereas the worst-case sampling rate determines the maximum number of deployed ML applications [1]. The relation between utilization ρ and sampling rate λ can be described by Little's theorem, $\rho = \lambda/\mu$, where μ is the service rate, i.e., maximum throughput. To ensure the system stability $(\rho < 1)$, operators usually place applications in a way that guarantees the service rate (throughput) exceeds the sampling rates. Although an edge server might not accommodate the full load of another server, it might be able to accommodate a partial load by accepting a lower sampling rate that suits its utilization. Applications can also distribute their load among multiple servers based on each server's utilization. Finally, operators can use sophisticated analytical models such as queueing theory to determine the best sampling rate and sample spreading ratios in failure scenarios [1].

Key Insight: Lowering the sampling rate decreases the server's utilization, allowing higher application packing rates

[1]https://pytorch.org/vision/stable/models.html
[2]https://sparsezoo.neuralmagic.com/

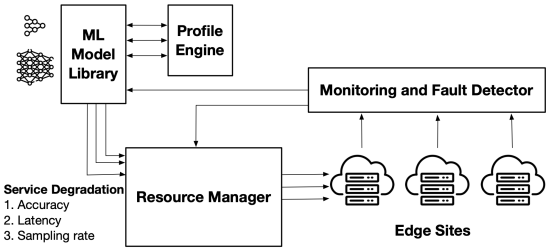


Fig. 5: System overview.

and increasing the reliability of ML inference systems without the need for additional resources.

IV. System Design

This section presents key artifacts required to design failure-resilient ML inference systems at the edge. We begin with an overview of the architecture and subsequently describe the key components.

A. System Overview

Figure 5 illustrates the architecture of the system. It comprises four main components: **ML model library** and **profile engine** are for storing and analyzing the ML inference models, while **monitoring and fault detector** is for detecting failures in the edge network and measuring the performance of applications and edge nodes. **Resource Manager** leverages the information mentioned above to determine how to place and configures models to ensure failure resiliency.

B. ML Model Library

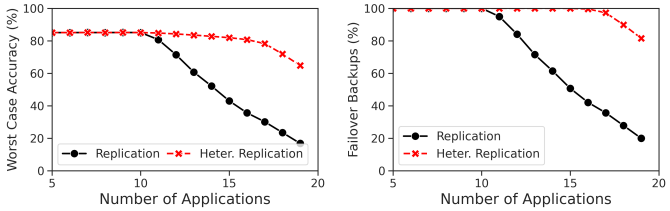
This component stores models for edge ML inference tasks. For each task, users can upload multiple versions of the same model. The model library offers flexibility in selecting a suitable model to match the available resources at edge sites, reducing the time required to load and start selected models after node failures are detected.

C. Profile Engine

The profiling engine provides information about the models stored in the library, assisting the model placement component in decision-making. It collects three types of profiling data for each model: (i) FLOPS and memory usage, indicating the number of floating-point operations and memory required to load the model, (ii) model accuracy, and (iii) model inference latency. Profiling data is stored as metadata in the ML model library and can be updated with runtime monitoring metrics to adapt to dynamic and heterogeneous edge environments.

D. Monitoring and Fault Detector

This component continuously monitors the status of edge sites and detects node failures in real-time. Metrics for both edge nodes and inference tasks are collected. Metrics such as uptime and resource utilization of edge nodes are gathered, while resource usage and execution time metrics are collected for inference tasks. The *uptime* metric is used to detect node failures, and the metrics related to inference tasks aid in updating model profiling data. These metrics, along with the resource utilization of edge nodes, contribute to decision-making in the model placement component. Once a node failure is detected, it triggers the failover procedure.



(a) Worst-case accuracy (b) Models with a backup (%)

Fig. 6: Evaluating traditional and heterogeneous replication for providing failover backups in terms of worst case accuracy and % of applications with a backup.

E. Resource Manager

This module serves as the core of our system, enhancing the resilience of ML inference at the edge by placing and configuring failover models with graceful service degradation techniques. The primary problem in this component is determining *how to select the ML models and distribute them across multiple edge sites given the limited resources and the models provided in the library?* The techniques described in Section III, which consider resource constraints and balance accuracy, latency, and sampling rate, are critical to this decision-making process.

V. Results

This section demonstrates the effectiveness of graceful service degradation in enhancing the system's resiliency. We profile the memory requirements and latency for many pre-trained models from the torchvision library on a Jetson Nano 4G node and use simulation-based analysis to demonstrate the benefits of accuracy and sampling graceful degradation in providing failure resiliency.

Figure 6 evaluates the benefits of accuracy degradation using heterogeneous replication by comparing it to traditional replication. In this case, we simulate a cluster of 10 Jetson Nano nodes, deploy different numbers of applications, and try to deploy a failover replica for each. We place both applications and replicas based on a greedy worst-fit approach. We also sample different combinations of models as primary and backups and repeat the experiment 100 times to ensure the result's stability. Figure 6a shows the worst-case accuracy of both methods, where we assume that accuracy drops to zero when the application does not have a failover replica. As shown, heterogeneous replication can retain a high accuracy with negligible drops for up to 17 applications. However, traditional replication can only support 10 applications, leading to a rapid decrease in accuracy. The reasons for this are depicted in figure 6b, which shows the percentage of applications with a backup. In the worst case, 80% of applications can have a backup model. We note that for more than 19 replicas, hosting the initial replicas became unfeasible.

Next, we evaluate the capacity of sampling degradation to provide resilience. Figure 7 depicts a scenario where a

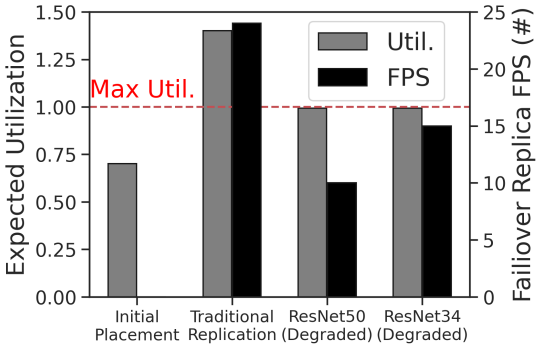


Fig. 7: Evaluating sampling degradation for providing failover backups.

node hosts a ResNet50 model processing samples at 24 FPS, denoted as the initial placement. To ensure failure-resilient execution, we try to deploy a failover replica with the least possible degradation. We follow the methods in [1] to compute the utilization and service rate when deploying multiple models. For simplicity, we assumed that the failover replica for the other application was also a ResNet50 model. As shown, in the case of traditional replication, the expected utilization exceeded 1, indicating that the system could not process all incoming requests, potentially resulting in queuing or dropping of requests. However, by degrading the sampling rate to 10FPS, the system can host the replica without any issues. In the last column, we show that the user can utilize both accuracy and sampling degradation, retaining most of the sampling rate when employing a smaller (less accurate) model.

VI. Challenges and Future Directions

In this section, we explore the crucial challenges faced in enabling failure-resilient ML inference at the edge through graceful degradation and provide research directions for future work.

A. Challenges

Determining the optimal placement of primary and backup models at the edge sites with quality of service degraded, presents a formidable challenge. One one hand, the placement problem is complicated by the dynamics, heterogeneity, and quantity of nodes and ML models. On the other hand, the trade-offs between accuracy, latency, and sampling rate might vary with scenarios. An adaptive approach becomes imperative to identify the most suitable trade-offs for degradation.

Furthermore, the dispersion of edge sites across geographic locations introduces complexities that render a centralized system inadequate for meeting the low-latency requirements of edge ML inference tasks. Latency increases due to data transmission, the retrieval of pretrained models from the ML model library, and global optimization across multiple edge sites. Consequently, devising strategies for deploying a fault-tolerance system across widely distributed edge sites becomes a demanding endeavor.

B. Limiting Degradation

This paper addresses the challenge of achieving resilient ML inference at the edge by replicating ML models while introducing service degradation. However, the introduced service degradation can have negative consequences in certain scenarios (e.g., the degraded accuracy of object detection in autonomous driving can be life-threatening). This limitation presents excellent opportunities for future research. One potential direction is to employ a larger number of failover replicas and utilize model cascades and ensembles to reduce the runtime resource requirements without sacrificing accuracy [18].

C. Byzantine Sensors

In addition to resilience against crash and transient failures that can lead to insufficient resources in the edge network, the system should also be capable of withstanding byzantine failures due to faulty sensors and malicious inputs. One way to address byzantine failures is by utilizing complementary information from multiple vantage points and multiple heterogeneous sensors and applying ML-based consensus algorithms, generally referred to as information or sensor fusion. Applying such techniques in resource-constrained environments remains an open research question.

References

[1] Q. Liang, W. A. Hanafy, A. Ali-Eldin, and P. Shenoy, "Model-Driven Cluster Resource Management for AI Workloads in Edge Clouds," *ACM Trans. Auton. Adapt. Syst.*, vol. 18, mar 2023.

[2] W. A. Hanafy, L. Wang, H. Chang, S. Mukherjee, T. V. Lakshman, and P. Shenoy, "Understanding the Benefits of Hardware-Accelerated Communication in Model-Serving Applications," in *Proceedings of IEEE/ACM 31st International Symposium on Quality of Service*, IWQoS'23, 2023.

[3] T. Abdelzaher, N. Ayanian, T. Basar, S. Diggavi, J. Diesner, D. Ganesan, R. Govindan, S. Jha, T. Lepoint, B. Marlin, K. Nahrstedt, D. Nicol, R. Rajkumar, S. Russell, S. Seshia, F. Sha, P. Shenoy, M. Srivastava, G. Sukhatme, A. Swami, P. Tabuada, D. Towsley, N. Vaidya, and V. Veeravalli, "Toward an Internet of Battlefield Things: A Resilience Perspective," *Computer*, vol. 51, no. 11, pp. 24–36, 2018.

[4] R. Schwartz, J. Dodge, N. Smith, and O. Etzioni, "Green AI," *Communications of the ACM*, vol. 63, pp. 54 – 63, 2019.

[5] W. A. Hanafy, T. Molom-Ochir, and R. Shenoy, "Design considerations for energy-efficient inference on edge devices," in *Proceedings of the Twelfth ACM International Conference on Future Energy Systems*, e-Energy '21, (New York, NY, USA), p. 302–308, Association for Computing Machinery, 2021.

[6] N. Liu, X. Ma, Z. Xu, Y. Wang, J. Tang, and J. Ye, "Autocompress: An automatic dnn structured pruning framework for ultra-high compression rates," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, pp. 4876–4883, Apr. 2020.

[7] S. Hashemi, N. Anthony, H. Tann, R. I. Bahar, and S. Reda, "Understanding the Impact of Precision Quantization on the Accuracy and Energy of Neural Networks," in *Proceedings of the Conference on Design, Automation & Test in Europe*, DATE '17, (Leuven, BEL), p. 1478–1483, European Design and Automation Association, 2017.

[8] Q. Liang, W. A. Hanafy, N. Bashir, A. Ali-Eldin, D. Irwin, and P. Shenoy, "Dělen: Enabling flexible and adaptive model-serving for multi-tenant edge ai," in *Proceedings of the 8th ACM/IEEE Conference on Internet of Things Design and Implementation*, IoTDI '23, (New York, NY, USA), p. 209–221, Association for Computing Machinery, 2023.

[9] C. Wan, M. Santriaji, E. Rogers, H. Hoffmann, M. Maire, and S. Lu, "ALERT: Accurate learning for energy and timeliness," in *2020 USENIX Annual Technical Conference (USENIX ATC 20)*, pp. 353–369, USENIX Association, July 2020.

[10] F. Romero, Q. Li, N. J. Yadwadkar, and C. Kozyrakis, "INFaaS: Automated model-less inference serving," in *2021 USENIX Annual Technical Conference (USENIX ATC 21)*, pp. 397–411, USENIX Association, July 2021.

[11] M. Kuniavsky, "Chapter 18 - common design challenges," in *Smart Things* (M. Kuniavsky, ed.), pp. 273–286, Boston: Morgan Kaufmann, 2010.

[12] M. Satyanarayanan and N. Davies, "Augmenting Cognition Through Edge Computing," *Computer*, vol. 52, no. 7, pp. 37–46, 2019.

[13] J. Soifer, J. Li, M. Li, J. Zhu, Y. Li, Y. He, E. Zheng, A. Oltean, M. Mosyak, C. Barnes, T. Liu, and J. Wang, "Deep Learning Inference Service at Microsoft," in *2019 USENIX Conference on Operational Machine Learning (OpML 19)*, (Santa Clara, CA), pp. 15–17, may 2019.

[14] P. A. Alsberg and J. D. Day, "A Principle for Resilient Sharing of Distributed Resources," in *Proceedings of the 2nd International Conference on Software Engineering*, ICSE '76, (Washington, DC, USA), p. 562–570, IEEE Computer Society Press, 1976.

[15] J. J. Meza, T. Gowda, A. Eid, T. Ijaware, D. Chernyshev, Y. Yu, M. N. Uddin, R. Das, C. Nachiappan, S. Tran, S. Shi, T. Luo, D. K. Hong, S. Panneerselvam, H. Ragas, S. Manavski, W. Wang, and F. Richard, "Defcon: Preventing Overload with Graceful Feature Degradation," in *17th USENIX Symposium on Operating Systems Design and Implementation (OSDI 23)*, (Boston, MA), pp. 607–622, USENIX Association, July 2023.

[16] C. Samplawski, J. Huang, D. Ganesan, and B. M. Marlin, "Towards objection detection under iot resource constraints: Combining partitioning, slicing and compression," in *Proceedings of the 2nd International Workshop on Challenges in Artificial Intelligence and Machine Learning for Internet of Things*, AIChallengeIoT '20, (New York, NY, USA), p. 14–20, Association for Computing Machinery, 2020.

[17] S. Jamil, M. J. Piran, M. Rahman, and O.-J. Kwon, "Learning-driven lossy image compression: A comprehensive survey," *Engineering Applications of Artificial Intelligence*, vol. 123, p. 106361, 2023.

[18] X. Wang, D. Kondratyuk, E. Christiansen, K. M. Kitani, Y. Movshovitz-Attias, and E. Eban, "Wisdom of Committees: An Overlooked Approach To Faster and More Accurate Models," in *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*, 2022.