A Survey on Artificial Intelligence and Machine Learning for Edge Devices

Johnathan Edwards

Abstract

Artificial intelligence (AI) and machine learning (ML) have rapidly become integral to a wide array of consumer, industrial, and commercial technologies. While early AI deployment heavily relied on cloud computing, growing concerns over latency, energy efficiency, and data privacy have driven a shift toward local, on-device machine learning at the network edge. This transition introduces new challenges, as edge devices typically possess limited computational power, memory capacity, and energy resources. This survey explores key advancements that enable efficient ML deployment on edge platforms, including model compression techniques, runtime optimization strategies, lightweight hardware accelerators, and collaborative resource management algorithms. In addition, this paper highlights a comparatively underexplored area: empirical energy profiling of models, kernels, and edge hardware. By aggregating findings from recent studies, this survey aims to aid developers and researchers in designing efficient, high-performance ML systems suitable for the evolving edge computing landscape.

Table of Contents

1 Introduction **1**

1.1 Related Surveys 2

1.2 Contribution of this Survey 3

1.3 Organization of this Survey 3

2 Background **4**

2.1 Edge Devices 4

2.2 Challenges of Cloud Computing 4

2.3 On-Device Models 5

3 The Survey **5**

3.1 Edge Machine Learning Applications 5

3.2 Model Development 6

3.2.1 Data Gathering and Preprocessing 7

3.2.2 Software Frameworks 7

3.2.3 Model Optimization Techniques 8

3.2.3.1 Pre-Runtime Optimization 9
3.2.3.2 Runtime Optimization 10
3.3 Hardware Accelerators for the Edge 11
3.4 Resource Management 12
3.4.1 Training Edge Models 12
3.4.2 Optimizing Resources During Inference 14
3.5 Resource Usage Profiling 15
4 Conclusion **16**
5 References **17**
Appendices .....
Reference Hierarchy **A1**
Annotated Bibliography **A2**

1 Introduction

Artificial intelligence (AI) and machine learning (ML) have become integral parts of modern life. People interact with intelligent applications on a daily basis through smartphones, wearable health monitors, and home automation assistants that can adjust the thermostat or manage grocery lists. Vehicles now feature smart technologies like collision avoidance and lane centering sensors. In industry, predictive maintenance tools help companies service machines before failures occur, while retailers automate inventory tracking with computer vision and use algorithms to forecast consumer demand. As computer technology improves and brings computational power closer to users, AI is being embedded in more aspects of everyday life.

Generally speaking, AI refers to computer models designed to approximate human reasoning based on the available data. Machine learning is a specialized branch of AI where models improve their decision-making ability by learning directly from data. Training often involves a staggering number of computations, requiring significant computational resources. Most training today is conducted on large server clusters – commonly referred to as “the cloud” – comprised of tens of thousands of interconnected hardware units, financed by corporations that can afford the multimillion-dollar infrastructure cost. Without access to such resources, developing cutting-edge AI models would be prohibitively slow.

Early deployment of AI applications to consumer devices relied on cloud-based models because mobile devices lacked the memory and processing speed to run them

locally. When a device needed to perform an AI task, it sent data to the cloud, where the model processed it and returned the results. However, cloud computing introduced several drawbacks including data privacy concerns, high communication latency, and reliability issues in areas with poor network coverage.

To overcome these limitations, researchers are increasingly focused on deploying AI models directly onto edge devices. Such devices include everything from mobile computers and smart phones to the so-called internet of things (IoT) – that is, the lightweight chips embedded in things like automobiles, appliances, and industrial machinery. Unlike cloud servers, edge have limited computing power, smaller memory capacities, and often operate under significant energy constraints, especially when battery-powered. Current research explores solutions such as model compression, task sharing between local devices and edge servers, specialized lightweight machine learning hardware, and privacy aware training approaches.

1.1 Related Surveys

Due to the popularity of AI and machine learning research, there have been numerous quality surveys on the topic of AI on the edge. Broadly, these works provide a comprehensive background of edge systems and the resource constraints of edge devices. References [42] and [67], for example, explore various techniques used to adapt AI and ML for edge deployment and highlight research gaps for future exploration. Reference [65] highlights AI in vehicular networks, while [66] provides similar insights mobile wireless network environments. Additionally, [68] and [69] analyze computational offloading algorithms for edge devices aimed at balancing workloads and optimizing metrics like energy consumption.

Together, these surveys highlight the complexity of designing intelligent applications for edge environments. However, a notable area that receives less coverage by these reviews is the empirical analysis of energy consumption profiles for machine learning architectures, individual kernel operations and lightweight hardware accelerators. Such data is useful for developers when evaluating design tradeoffs for edge intelligence problems and represents an important gap in the existing literature that warrants further investigation.

1.2 Contribution of this Survey

This paper follows a structure similar to that of other surveys, beginning with a background on edge computing and the growing trend in deploying intelligent models closer to the data source. It explores key challenges for on-device ML model development and highlights recent advancements in model compression and hardware optimization in

this domain. Additional sections are included that cover widely used software frameworks used for edge ML development, as well as some of the algorithms designed to manage resource sharing among edge devices in a network.

The unique contribution of this paper is a section dedicated to the energy consumption metrics, drawing from various empirical studies that evaluate model architectures and hardware accelerators under controlled laboratory conditions. This data provides valuable insights for guiding future research and design decisions in the field of edge computing.

1.3 Organization of this Survey

The purpose of this survey is to highlight contemporary research in machine learning and artificial intelligence for edge computing. In the following sections, we will discuss the development of edge intelligence, optimization techniques to enhance performance, and practical energy performance metrics derived from empirical studies.

The structure of the paper and brief summaries of each section follows:

1.3.1 Section 1: Introduction. This introduction serves to briefly describe the expansion of AI and ML innovation in the edge computing ecosystem as well as an outline of the survey.

1.3.2 Section 2: Background. The background section provides a summary of the types of devices included in the edge domain and challenges experienced by developers as edge intelligence applications gained popularity.

1.3.3 Section 3: The Survey. The research presented in this section forms the core of the survey. While the intention was to cover the foundational topics in edge intelligence, this work is not exhaustive, as there are many facets to the subject. Subsection 3.1 introduces several examples of edge AI applications that are already deployed and widely adopted across various industries. Subsection 3.2 outlines the phases of model development, including data gathering and preprocessing, software framework selection, and optimization techniques aimed at improving model performance. Section 3.3 covers the expanding array of specialized lightweight hardware architectures designed to support edge intelligence. Section 3.4 describes the methodologies and algorithms used to manage training and inference tasks across networks, exploring the trade-offs between local processing and task offloading. Finally, Section 3.5 highlights recent empirical studies that aggregate energy consumption data for machine learning models and edge hardware.

2 Background

2.1 Edge Devices

Edge devices refer to the broad range of network hardware positioned closest to the user or data source. Often, these devices serve as endpoints that provide users access to network resources, such as personal computers and mobile devices. This category also extends to a rapidly expanding variety of Internet of Things (IoT) systems, including autonomous vehicles, consumer drones, industrial machinery, and connected medical devices. The goal of edge computing is to deliver real-time, low-latency processing capabilities at the point where data is generated, ultimately enhancing the quality of service for the user.

2.2 Challenges of Cloud Computing

The cloud refers to the centralized servers that form the backbone of large networks such as the internet or large enterprise systems. These servers possess significantly greater processing power and use more energy than any single edge device, and they are commonly used to offload complex computational tasks when local resources are insufficient. In artificial intelligence, the traditional paradigm is to train and host AI systems in the cloud, where large volumes of data and computations can be handled efficiently. When an edge device requires inference, it sends a request to the cloud – along with relevant sensor data – via a network connection. After processing the request, the cloud server returns the inference result back to the device over the same network.

However, offloading AI tasks to the cloud introduces several drawbacks that make it less suitable for certain applications. For example, consider a company operating a smart manufacturing facility. Operational constraints might demand real-time control over critical processes and protection of sensitive data. Engineers testing cloud-based applications may find that communication delays introduce significant latency and transmitting raw sensor data over a shared network compromises data security. In such cases, cloud computing falls short of the facility's requirements due to latency and privacy concerns. A more appropriate solution might be to invest in private servers located on-site.

Yet, depending on the company's computational demands and financial resources, purchasing and maintaining high-performance servers could be cost prohibitive. An alternate approach is to leverage lightweight AI models specifically optimized to run in resource-constrained environments. Additionally, many hardware manufacturers offer ultra-low powered chips designed to efficiently handle machine learning tasks. Much of this paper focuses on surveying the relevant research dedicated to enabling locally run, on-device models.

2.3 On-device Models

On-device machine learning models, compared to their cloud-based counterparts, operate under tighter resource constraints, requiring smaller memory footprints and reduced processing power. Consequently, these models often suffer from decreased accuracy, but benefit from lower latency making them attractive for time sensitive tasks. Another advantage of on-device models is enhanced data security, as all computations are performed locally, eliminating the need to transmit sensitive information. However, adapting on-device models to the vast array of heterogeneous hardware architectures found in an edge environment can be a significant challenge. By contrast, developing a single cloud-based model is generally simpler and less labor-intensive.

3 The Survey

3.1 Edge Machine Learning Applications

Advances in computer hardware have opened new possibilities for deploying machine learning applications on edge devices. Today's mobile phones, for instance, possess hundreds of thousands of times more computing power and millions of times more memory than the Apollo 11 guidance computer [70]. These improvements have made applications that were formerly impossible a reality in modern edge environments. As hardware continues to evolve, the frontier of what is possible constantly expands. Nevertheless, a wide variety of edge intelligence applications are already available.

Current applications span a diverse range of industries – from consumer electronics and industrial manufacturing controls to medical devices, autonomous vehicles and beyond. Subhash et al. [71] explore the development and utility of artificial intelligence-based voice assistants, showcasing how AI has been integrated into everyday consumer products. Smartphones now commonly feature image recognition and object detection capabilities embedded directly into camera applications, tasks well suited for convolutional neural networks like ResNet [72] and MobileNet [24]. Beyond traditional approaches, researchers like Brendan Reidy et al. [17] have investigated methods to further enhance on-device performance in computer vision tasks. Their work introduces an in-sensor pre-processing methodology that reduces computational overhead during inference for high-resolution images.

In the healthcare domain, Linjuan Ma et al. [72] present a machine learning model designed for brain tumor diagnosis. The system seeks to balance the trade-off between computational efficiency of a lightweight neural network and diagnostic accuracy.

Smart agriculture represents another promising field for edge AI. A study by Hayajneh et al. [18] proposes a novel system where unmanned aerial vehicles (UAVs) collect data from widely distributed soil sensors in rural areas with limited network

coverage. These same UAVs assist in delivering lightweight machine learning models to field devices, enabling localized prediction of optimal watering schedules and gathering inference results efficiently.

Modern security systems employ a variety of cameras, sensors and intrusion denial devices to enhance protection for people and property. People commonly interact with such systems at transportation hubs, schools and businesses. Wang et al. [48] explore a common drawback of these systems – frequent false alarms – which degrades the effectiveness of detection and response of security personnel. Their paper introduces a machine learning algorithm that learns from sensor data to filter false alarms from real emergencies.

3.2 Model Development

Developing new machine learning models is a complex process, involving several interdependent phases, each critical to building a robust system. These phases typically include data gathering and preprocessing, model development and optimization, and finally system integration [42, 67]. For some instances, relevant datasets are already publicly available through online repositories, or an existing model can be fine-tuned to suit a specific task. Regardless of the starting point, it is important to understand the key steps involved in model development.

3.2.1 Data Gathering and Preprocessing

Every machine learning model relies on relevant data for effective reasoning and decision making. Data can be collected from a variety of sources, including raw sensor outputs, crowdsourcing efforts, synthetic generation, and internet repositories [75]. However, raw data is often not immediately useful, as it may contain missing information, duplicate records, or irrelevant values [42]. Before being used for model training, the data must be cleaned to minimize inference biases and protect against malicious or corrupted entries [75].

3.2.2 Software Frameworks

Developing machine learning programs for edge devices requires a delicate balance between minimizing resource usage and maximizing performance metrics like accuracy and low latency. Device compatibility can be a major concern, as some high-level programming languages depend on runtime libraries that cannot fit in the limited memory of lightweight hardware often used for edge applications [14]. Fortunately, several software frameworks are available that have been specifically designed to address the constraints of resource-limited environments.

TensorFlow LiteRT, formerly known as TensorFlow Lite, is a Google supported framework optimized for on-device AI models. LiteRT supports multiple programming languages – including Java/Kotlin, Swift, Objective-C, C++, and Python – and targets a range of platforms such as Android, iOS, embedded Linux, and microcontrollers. Models developed in other popular frameworks like TensorFlow, PyTorch, and JAX can also be converted into .tflite format for deployment. LiteRT models feature low-latency, strong data-privacy, small size and efficient power consumption [78].

Caffe2 is a lightweight, modular deep learning framework developed by Facebook to support scalable deployment. While the original Caffe framework was designed for large-scale systems, Caffe2 extends its reach to mobile devices, with enhanced support for distributed training on multiple CPU and GPU configurations, and cross-platform compatibility. Currently, APIs are included for C++ and Python, and methods are available for converting Caffe models to Caffe2 format [79].

MXNet is an open-source framework known for its flexibility in both research and production settings. It is well-suited for distributed training on multi-GPU systems using a parameter server architecture and Horovod integration. Although less widely adopted than TensorFlow and PyTorch, MXNet offers support for eight programming languages – including Python, Scala, Julia, Clojure, Java, C++, R, and Perl – along with a host of high-performance libraries and backend tools [80].

MicroTVM provides an end-to-end code generator that automates conversion of models from high-level languages directly into structured C source libraries suitable for bare-metal devices, like microcontrollers, that lack an operating system. Developers can write their models using popular frameworks like TensorFlow or TF LiteRT, and MicroTVM automatically generates header files, source code, and compilation scripts, greatly simplifying deployment. While still under active development, MicroTVM also allows developers to register new or custom operators when needed. Its primary goal is to streamline the conversion process, allowing engineers to focus on higher level design challenges [14].

3.2.3 Model Optimization Techniques

A variety of techniques exist to enhance machine learning models in terms of performance, memory footprint, communication overhead, and energy efficiency. This area of research is one of the fundamental features critical to the advancement of machine learning and expanding its range of practical applications. As AI becomes increasingly popular, especially in on-device deployments, it is important for developers to understand how to adapt models for resource-constrained environments while preserving key

performance metrics. This section explores some of the techniques that have been proposed to address these challenges.

3.2.3.1 Pre-Runtime Optimization

Model pruning is a technique that systematically identifies and removes redundant parameters, or consolidates them among similar features, to reduce model size and computational complexity. By eliminating unused or less important parameters, and in some cases entire layers, pruned models are able to bypass unnecessary calculations. This is vital for on-device models that would otherwise be unable to fit in available storage or limited by lower processing throughput. Gao et al. [82] introduce a structured pruning method that leverages a discrete gate mechanism for filter channels, restricting gradient updates to active channels while freezing unused parameters. Oftentimes it is possible to simplify existing models by reducing the number of layers and combining operations. Kim et al. [3] applied this approach to SqueezeNet, first reducing the architecture from eight fire modules to four. Additional compression was achieved by embedding max pool functions into convolutional layers by increasing stride length in the middle of convolution operations and applying quantization to activations and parameters.

Careful data analysis and preprocessing is a viable method for reducing model computations by selecting only the most important features from training examples. In their experiments, Gomez-Carmona et al. [13] cleaned and partitioned accelerometer signals from wearable health monitors, removing noise and segmenting the signal into discrete parts. They then evaluated pre-trained movement classification models using different combinations of features of varying lengths to identify the minimum feature set required to maintain at least 90% accuracy. Their findings concluded that just the three best features were necessary to achieve the convergence target.

Reidy et al. [17] proposed an automated approach for an object recognition task involving very-high resolution images. To accommodate such large data inputs, their method introduced an in-sensor compression algorithm that first converts images to grayscale and then identifies regions of interest within the compressed sample. The object detection model processes extracted regions in a piecewise fashion, significantly reducing computational overhead by focusing only on relevant sections of the original image.

At a more granular level, convolutional layers in neural networks involve fundamental operations such matrix multiplication, scalar addition and non-linear transformations. To improve efficiency, parameter matrices can be decomposed into smaller components through matrix factorization, resulting in simplified calculations and improved latency [20]. The resulting representation effectively compresses the original

parameter matrix at a rate of $mn/k(m+n+1)$, where *m* and *n* are the matrix dimensions and *k* is the number of singular vectors – vectors whose directions remain unchanged by the transformation. In addition to matrix factorization, Gong et al. [19] explore several vector quantization techniques, including binarization, k means grouping, product quantization, and residual quantization. Like matrix factorization, these methods aim to exploit parameter redundancy to produce more compact and efficient model representations.

3.2.3.2 Runtime Optimization

Edge devices frequently operate in dynamic mobile environments where wireless connectivity and resource availability can fluctuate dramatically. Thus, machine learning applications benefit from having a degree of adaptability to the surrounding computing conditions.

Lane et al. [1] were among the early innovators with their 2016 DeepX framework, which dynamically selects layer compression and model partitioning configurations according to available computational resources for each inference request. DeepX employs a two-stage workflow: the Deep Architecture Decomposition (DAD) algorithm explores multiple partitioning strategies and ranks them according to accuracy estimates generated by the Runtime Layer Compression (RLC) algorithm. An additional goal of the framework is energy-efficient inference, which it accomplishes by favoring offloading schemes that prioritize low-power processors over high-performance ones whenever possible.

More recently, MyML introduced a privacy-conscious optimization approach grounded in a transfer learning style framework. Designed to meet device resource constraints, MyML intelligently simplifies large, generic models based on individual user preferences. Observing that user behavior typically centers around a narrow set of functions, the authors noted that conventional compression methods fail to adapt accordingly. In contrast, MyML first learns user habits by aggregating commonly used task categories. To minimize accuracy losses from compression, it avoids pruning shallow feature extraction layers and instead removes unused categories from the deeper classification layers, effectively personalizing the model. This entire process is performed locally on the user's device to preserve privacy, and adaptation is ongoing, with periodic re-pruning initiated whenever the frequency of new use cases surpasses a defined threshold [9].

Another runtime optimization explored in edge settings involves multi-exit frameworks for deep neural networks, which take advantage of the varying complexity of

input data. In many cases, object classification tasks can achieve sufficient accuracy at earlier layers during inference when presented with simpler inputs, making it unnecessary to process through the entire model. MAMO exemplifies this approach by identifying optimal exit points using a bidirectional dynamic programming algorithm. It further enhances system performance by employing a reinforcement learning model to select ideal model partitioning configurations based on the device's current resource availability [81].

3.3 Hardware Accelerators for the Edge

Researchers often emphasize the need for a holistic approach when designing machine learning models for edge deployment [2, 3]. This perspective highlights the importance of accumulating marginal performance gains not only through optimized software implementations but also through carefully tailored hardware architectures. To meet these hardware demands, a growing ecosystem of manufacturers designs lightweight chips specifically for machine learning, supported by ongoing academic research. Developers must often weigh the strengths and trade-offs of these specialized devices, as some are optimized for specific tasks while others offer broader flexibility.

Among prominent commercial offerings, Google's flagship product for edge machine learning is the Edge TPU – a lightweight coprocessor available in various form factors, capable of performing four trillion operations per second while consuming just two watts of power [84]. The Edge TPU is optimized for deep feed-forward neural networks, such as convolutional neural networks (CNNs), and provides native support for TensorFlow Lite, Google's lightweight ML framework [83]. However, performance can degrade substantially when running network architectures that do not align with its design [11].

Another widely used commercial platform is Nvidia's Jetson, developed by a company renowned for AI hardware innovation. Jetson distinguishes itself through extensive software ecosystem support and an active developer community. It is compatible with TensorFlow Lite, PyTorch, and MXNet, and offers a suite of tools and libraries through its comprehensive SDKs. Like the Edge TPU, Jetson modules come in a range of form factors with varying levels of computational power and memory, all engineered for edge applications [85].

Turning to academic contributions, Mensa, introduced by Boroumand et al. [11], is a novel lightweight hardware accelerator. Mensa's key innovation is its heterogeneous architecture: a collection of specialized accelerators, each tailored for a subset of common neural network operations. Recognizing that neural network layers naturally cluster by similar computational patterns, the Mensa framework dynamically maps each

group to the most appropriate accelerator via a runtime scheduler. Unlike the monolithic design of the Edge TPU, Mensa offers greater adaptability across diverse model architectures.

The Intelligence Boost Engine (IBE) targets ultra-low-power applications such as sensor fusion and motion recognition, benchmarking its performance against the Cortex-M class of processors. Cortex-M devices, often found in wearable sensors and smart components, operate at clock speeds between 10–100 MHz and are prized for their extreme energy efficiency, consuming roughly 100 times less power than Cortex-A processors. IBE is optimized for core operations like matrix multiplication, scalar operations, and power functions, and provides additional flexibility with six programmable operating modes [10].

Eyeriss is an early hardware accelerator developed at MIT that is frequently referenced in research literature. It features a spatial architecture with 168 processing elements (PEs), designed to efficiently parallelize common neural network computations while minimizing memory access overhead. Eyeriss improves memory efficiency through a novel row-stationary dataflow, promoting data reuse across inputs, weights, and partial sums. Its four-level memory hierarchy reduces costly off-chip memory accesses, and an integrated network-on-chip (NoC) communication framework further enhances data transfer efficiency between memory levels and processing elements [25].

3.4 Resource Management

A central challenge in implementing machine learning on edge devices is adapting computationally intensive tasks to environments with limited resources, a theme that has been emphasized throughout this paper. Thus far, we have examined model compression techniques aimed at improving efficiency and simplifying computations, data processing strategies that further enhance performance metrics, and hardware optimizations tailored to efficiently execute algebraic operations common to model layers. This section shifts focus to algorithms that intelligently manage task distribution across connected devices. These algorithms often navigate competing objectives, such as balancing task deadlines against energy efficiency requirements [8], or preserving data privacy without compromising model accuracy [26]. For clarity, the following discussion organizes the algorithms into two categories based on the phase of machine learning in which they operate: training and inference.

3.4.1 Training Edge Models

Federated learning is designed to preserve user privacy while leveraging the diversity of data generated across a broad population of mobile users. Similar to traditional cloud-

based training paradigms, a central server maintains a global model that is shared among the various devices within the network. A straightforward, but problematic, approach would involve uploading all user data to the cloud, aggregating it into a massive training set, and updating the global model with each training epoch. However, this method presents two major challenges: the sheer volume of data transfer imposes heavy network demands, and the transmission of potentially sensitive personal information raises serious privacy and security concerns. Federated learning addresses these issues by selecting a group of devices to train local copies of the model using their own data and then aggregating only the resulting gradient updates to refine the global model. McMahan et al. introduced the FederatedAveraging algorithm, which combines local stochastic gradient descent with centralized gradient averaging, demonstrating the feasibility of this decentralized approach. By transmitting only model updates rather than raw data, federated learning significantly reduces network traffic and mitigates privacy risks, ensuring that personal information remains securely on each user's device [26].

Transfer learning is another framework that preserves data privacy by enabling local model training. Unlike federated learning, which focuses on collaboratively training a shared model, transfer learning emphasizes fine-tuning a pre-trained model for a specific application. For many applications there are existing models that perform well at the general task, such as ResNet or MobileNet for object detection [15]. However, individual users may require support for additional detection classes not covered by the original model. These can be incorporated by fine-tuning the model's classification layers using new, locally collected data. Hayajneh et al. [18] applied this approach in a smart agriculture setting, introducing a novel communication system supported by unmanned aerial vehicles (UAVs). In their design, UAVs transport data to and from field-based microcontrollers that control irrigation equipment. Each microcontroller is equipped with sensors to monitor soil humidity and a built-in DNN to estimate optimal watering schedules. Given the limited connectivity – provided solely by drones – transfer learning enables the DNN to be locally fine-tuned with sensor data, ensuring model adaptation without the need for continuous network access.

Dong et al. [16] present a distributed training framework called EdgeMove that implements model parallelism by partitioning model layers between the device and edge servers, which they refer to as workers or worker devices. Their approach improves upon traditional device-to-cloud training by leveraging the lower communication latency offered by geographically local edge servers. Two key innovations in EdgeMove are a pipelining algorithm that overlaps minibatches to further reduce communication delays, and a runtime adaptation algorithm that dynamically adjusts to fluctuating network conditions. The pipelining strategy draws inspiration from PipeDream [86], a multi-dimensional

parallelism technique that maximizes throughput. Meanwhile, the runtime adaptation mechanism continuously monitors the performance of the assigned worker and neighboring workers. If a worker's performance degrades, the client can reconfigure the model partitioning or offload specific stages to a different worker, maintaining efficient training performance despite network variability.

Edge2Train is a training framework specifically targeted at lightweight machine learning models on ultra-low power devices, such as microcontrollers, and shares some conceptual similarities with transfer learning. The key challenges addressed by the framework are sparse connectivity and privacy issues with mobile networks. Edge2Train employs a background process that continuously monitors the inference accuracy of the device's model. A configurable threshold is set to determine when the model's performance is outside of acceptable limits, in which case new training data is accumulated from the associated inputs provided by the device sensors. Once sufficient new training data is accumulated, the model is automatically retrained, effectively fine-tuning it to adapt to the local environment [12].

3.4.2 Optimizing Resources During Inference

In Section 3.2.3.2, we discussed resource management strategies that dynamically compressed or pruned portions of a model based on the available connected devices in the network [1, 9]. However, in some situations, modifying the model structure can lead to unacceptable losses in accuracy. This section shifts focus to algorithms that manage task sharing without altering the underlying model.

Adding edge servers to a network is often a practical way to provide lightweight devices with additional processing power. Yet, as network size grows, this strategy becomes less sustainable due to increasing costs and energy consumption. To address this, the developers of DEWOrch introduced a deep reinforcement learning algorithm for workload orchestration aimed at enabling inter-device collaboration in mobile networks. Recognizing that many tasks require only a fraction of a server's maximum resources, DEWOrch ranks all connected devices by computational performance and, when possible, offloads tasks to the least powerful capable device. This strategy preserves powerful resources for more demanding tasks while minimizing overall energy consumption by prioritizing energy-efficient devices [7].

A related system was developed by Chouikhi et al. for industrial Internet of Things (IIoT) environments. Unlike generic mobile settings, IIoT systems often operate under strict real-time constraints, where missing a task deadline could lead to system failure. To address these requirements, their framework employs multiple deep reinforcement

learning agents, each assigned to a device within the network. These agents optimize multiple objectives, prioritizing task deadlines while also considering long-term energy efficiency. Communication among agents is enabled to prevent conflicting decisions and maximize overall system performance [8].

Ali et al. [29] introduced the Energy-Efficient Deep Learning-Based Offloading Scheme (EEDOS) for cloud-based task sharing. Their method stands out by training the decision-making model with an exhaustive dataset that enumerates every possible partitioning configuration of the inference model. As a result, EEDOS achieves 100% decision accuracy during inference. Notably, it is also one of the first frameworks to account for the remaining battery life of the host device, dynamically adjusting offloading decisions based on the device's power state.

3.5 Resource Usage Profiling

State-of-the-art research in edge machine learning requires scientists to stay current with related advancements to remain competitive. While some efforts have been made to aggregate quantitative data, particularly concerning model and hardware energy consumption in independent laboratory settings, such studies have received relatively little attention in survey papers. This section highlights several publications dedicated to this line of research, with many authors motivated by the desire to create accessible references for future developers.

In a 2020 paper, Kumar et al. [2] highlight notable energy efficiency improvements in contemporary edge hardware accelerators and hardware based full-stack optimization methods. Devices like the Nvidia Jetson, Google Edge TPU, and ShiDianNao accelerators top the list for devices that outperform traditional server-grade CPUs and GPUs in energy metrics. However, the key contribution of the paper is an Eclipse IDE plugin called JEPO for Java programs that is able to suggest code improvements for more energy efficient applications. JEPO analyzes each line of code, analyzing data types, arithmetic operators, and algorithmic patterns such as array traversal, using empirical energy consumption data collected across various Java constructs. For example, experiments showed that the int primitive is more efficient than other data types, and ternary operators consume more energy than standard if-then-else statements. The tool was validated by refactoring WEKA, an open-source machine learning library, resulting in a 14.46% reduction in energy consumption compared to the baseline implementation.

Fanariotis et al. [4] evaluated the energy performance of four machine learning models – LeNet-5, MobileNet 025, an AI-based indoor localization model, and a sine calculation model – comparing original and quantized versions using TensorFlow Lite.

Although quantization generally improved energy efficiency by reducing kernel operations, the study found that the number of memory accesses was an even stronger predictor of energy savings. Interestingly, larger compute-bound models sometimes consumed less energy than smaller, memory-bound models. Moreover, energy performance varied significantly across devices, emphasizing the need for holistic hardware-software co-design. For example, an unoptimized LeNet model on an ESP32 chip consumed about three times less energy than a quantized version running on an ARM Cortex-M7 processor.

Hanafy et al. [5] evaluated 40 popular DNN models to explore the relationship between model size and three different metrics: accuracy, energy consumption and latency. They found no consistent correlation between model size and performance across different model families – VGG models, for example, were larger but less accurate than ResNet models. However, within individual model families, a positive correlation between size and performance was observed. Importantly, the number of floating-point operations, rather than model size alone, showed a stronger relationship with both latency and energy consumption. The results may also indirectly indicate that there are other variables that would yield better predictive quality, such as arithmetic intensity as shown in the previous study.

To avoid the impracticality of benchmarking every available model, Tu et al. [6] developed an experimental framework that aggregates energy consumption trends based on kernel characteristics. Their study evaluated 16 kernel types from nine cutting-edge DNN models across 50 variations each, along with six popular edge AI applications. They measured the energy consumption trends of variables like feature dimension, channel number, kernel size, and stride length. Additionally, they introduced two novel metrics – the Power Consumption Score (PCS) and Inference Energy Consumption Score (IECS) – to provide an intuitive energy efficiency rating for device-application pairings.

In computer vision applications, object detection and motion tracking are common tasks for smart cars and security systems. Object detection and motion tracking typically involves two independent algorithms in a sequential workflow. Using a Google Coral AI chip and a Nvidia Jetson Nano, Di Fabrizio et al. [15] tested MobileNet v2 and ResNet with three common multi-object tracking algorithms to measure the energy performance of each configuration. Beyond measuring energy consumption and latency for each configuration, their study also captured how tuning the accuracy settings of tracking algorithms impacted both performance metrics.

4 Conclusion

Edge computing represents a new era in how artificial intelligence and machine learning models are deployed, offering the potential for lower latency, improved data privacy, and greater autonomy across a wide range of applications. However, realizing these advancements requires overcoming significant constraints inherent to edge devices, including limited computing power, smaller memory storage, and restrictive energy budgets.

This survey reviewed many of the major techniques that address these challenges but falls short of being comprehensive. The variety of innovations continue to grow in this rapidly expanding field of study. We discussed model compression strategies designed to reduce resource demands without severely compromising accuracy, runtime optimization algorithms that adaptively manage computational overhead, specialized hardware accelerators for lightweight inference, and distributed resource management methods that enhance system efficiency through intelligent task sharing.

In addition to covering these well researched domains, this paper highlights a critical but less frequently discussed topic – empirical profiling of energy consumption in machine learning models and hardware accelerators. By aggregating findings from recent studies, this survey provides exposure to the practical insights derived from this avenue of research so that developers can benefit during future implementation. As edge intelligence continues to grow in importance, future innovation in methods that balance performance, energy efficiency, and system resilience will become even more crucial.

5 References

[1] N. D. Lane et al., "DeepX: A Software Accelerator for Low-Power Deep Learning Inference on Mobile Devices," 2016 15th ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN), Vienna, Austria, 2016, pp. 1-12, doi: 10.1109/IPSN.2016.7460664. Available: https://ieeexplore-ieee-org.libweb.lib.utsa.edu/stamp/stamp.jsp?tp=&arnumber=7460664

[2] M. Kumar, X. Zhang, L. Liu, Y. Wang and W. Shi, "Energy-Efficient Machine Learning on the Edges," 2020 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW), New Orleans, LA, USA, 2020, pp. 912-921, doi: 10.1109/IPDPSW50202.2020.00153. Available: https://ieeexplore-ieee-org.libweb.lib.utsa.edu/stamp/stamp.jsp?tp=&arnumber=9150337

[3] Kim, K.; Jang, S.-J.; Park, J.; Lee, E.; Lee, S.-S. Lightweight and Energy-Efficient Deep Learning Accelerator for Real-Time Object Detection on Edge Devices. *Sensors* **2023**, *23*, 1185. Available: https://doi.org/10.3390/s23031185

[4] Fanariotis, A.; Orphanoudakis, T.; Kotrotsios, K.; Fotopoulos, V.; Keramidas, G.; Karkazis, P. Power Efficient Machine Learning Models Deployment on Edge IoT Devices. *Sensors* **2023**, *23*, 1595. Available: https://doi.org/10.3390/s23031595

[5] Walid A. Hanafy, Tergel Molom-Ochir, and Rohan Shenoy. 2021. Design Considerations for Energy-efficient Inference on Edge Devices. In Proceedings of the Twelfth ACM International Conference on Future Energy Systems (e-Energy '21). Association for Computing Machinery, New York, NY, USA, 302–308. Available: https://doi.org/10.1145/3447555.3465326

[6] Xiaolong Tu, Anik Mallik, Dawei Chen, Kyungtae Han, Onur Altintas, Haoxin Wang, and Jiang Xie. 2024. Unveiling Energy Efficiency in Deep Learning: Measurement, Prediction, and Scoring across Edge Devices. In Proceedings of the Eighth ACM/IEEE Symposium on Edge Computing (SEC '23). Association for Computing Machinery, New York, NY, USA, 80–93. Available: https://doi.org/10.1145/3583740.3628442

[7] Z. Safavifar, E. Gyamfi, E. Mangina and F. Golpayegani, "Multi-Objective Deep Reinforcement Learning for Efficient Workload Orchestration in Extreme Edge Computing," in IEEE Access, vol. 12, pp. 74558-74571, 2024, doi: 10.1109/ACCESS.2024.3405411. Available: https://ieeexplore-ieee-org.libweb.lib.utsa.edu/document/10538273

[8] S. Chouikhi, M. Esseghir and L. Merghem-Boulahia, "Energy-Efficient Computation Offloading Based on Multiagent Deep Reinforcement Learning for Industrial Internet of Things Systems," in IEEE Internet of Things Journal, vol. 11, no. 7, pp. 12228-12239, 1 April1, 2024, doi: 10.1109/JIOT.2023.3333044. Available: https://ieeexplore-ieee-org.libweb.lib.utsa.edu/document/10318207

[9] Vidushi Goyal, Reetuparna Das, and Valeria Bertacco. 2022. Hardware-friendly User-specific Machine Learning for Edge Devices. ACM Trans. Embed. Comput. Syst. 21, 5, Article 62 (September 2022), 29 pages. Available: https://doi-org.libweb.lib.utsa.edu/10.1145/3524125

[10] Minkwan Kee and Gi-Ho Park. 2022. A Low-power Programmable Machine Learning Hardware Accelerator Design for Intelligent Edge Devices. ACM Trans. Des. Autom. Electron. Syst. 27, 5, Article 51 (September 2022), 13 pages. Available: https://doi-org.libweb.lib.utsa.edu/10.1145/3531479

[11] Amirali Boroumand, Saugata Ghose, Berkin Akin, Ravi Narayanaswami, Geraldo F. Oliveira, Xiaoyu Ma, Eric Shiu, and Onur Mutlu. 2024. Google Neural Network Models for Edge Devices: Analyzing and Mitigating Machine Learning Inference Bottlenecks. In Proceedings of the 30th International Conference on Parallel Architectures and

Compilation Techniques (PACT '21). IEEE Press, 159–172. Available: https://doi-org.libweb.lib.utsa.edu/10.1109/PACT52795.2021.00019

[12] Bharath Sudharsan, John G. Breslin, and Muhammad Intizar Ali. 2020. Edge2Train: a framework to train machine learning models (SVMs) on resource-constrained IoT edge devices. In Proceedings of the 10th International Conference on the Internet of Things (IoT '20). Association for Computing Machinery, New York, NY, USA, Article 6, 1–8. Available: https://doi-org.libweb.lib.utsa.edu/10.1145/3410992.3411014

[13] Oihane Gómez-Carmona, Diego Casado-Mansilla, Diego López-de-Ipiña, and Javier García-Zubia. 2019. Simplicity is Best: Addressing the Computational Cost of Machine Learning Classifiers in Constrained Edge Devices. In Proceedings of the 9th International Conference on the Internet of Things (IoT '19). Association for Computing Machinery, New York, NY, USA, Article 18, 1–8. Available: https://doi-org.libweb.lib.utsa.edu/10.1145/3365871.3365889

[14] Chen Liu, Matthias Jobst, Liyuan Guo, Xinyue Shi, Johannes Partzsch, and Christian Mayr. 2024. Deploying Machine Learning Models to Ahead-of-Time Runtime on Edge Using MicroTVM. In Proceedings of the 2023 Workshop on Compilers, Deployment, and Tooling for Edge AI (CODAI '23). Association for Computing Machinery, New York, NY, USA, 37–40. Available: https://doi-org.libweb.lib.utsa.edu/10.1145/3615338.3618125

[15] Giacomo Di Fabrizio, Lorenzo Calisti, Chiara Contoli, Nicholas Kania, and Emanuele Lattanzi. 2024. A Study on the energy-efficiency of the Object Tracking Algorithms in Edge Devices. In Proceedings of the IEEE/ACM 16th International Conference on Utility and Cloud Computing (UCC '23). Association for Computing Machinery, New York, NY, USA, Article 29, 1–6. Available: https://doi-org.libweb.lib.utsa.edu/10.1145/3603166.3632541

[16] Zeqian Dong, Qiang He, Feifei Chen, Hai Jin, Tao Gu, and Yun Yang. 2023. EdgeMove: Pipelining Device-Edge Model Training for Mobile Intelligence. In Proceedings of the ACM Web Conference 2023 (WWW '23). Association for Computing Machinery, New York, NY, USA, 3142–3153. Available: https://doi-org.libweb.lib.utsa.edu/10.1145/3543507.3583540

[17] Brendan Reidy, Sepehr Tabrizchi, Mohammadreza Mohammadi, Shaahin Angizi, Arman Roohi, and Ramtin Zand. 2024. HiRISE: High-Resolution Image Scaling for Edge ML via In-Sensor Compression and Selective ROI. In Proceedings of the 61st ACM/IEEE Design Automation Conference (DAC '24). Association for Computing Machinery, New York, NY, USA, Article 275, 1–6. Available: https://doi-org.libweb.lib.utsa.edu/10.1145/3649329.3656539

[18] Hayajneh, A.M., et al.: Tiny machine learning on the edge: a framework for transfer learning empowered unmanned aerial vehicle assisted smart farming. *IET Smart*

Cities. 6(1), 10–26 (2024). Available: https://doi-org.libweb.lib.utsa.edu/10.1049/smc2.12072

[19] Y. Gong, et al., "Compressing deep convolutional networks using vector quantization," arXiv preprint arXiv:1412.6115, 2014.

[20] Denton, Remi, Zaremba, Wojciech, Bruna, Joan, LeCun, Yann, and Fergus, Rob. Exploiting linear structure within convolutional networks for efficient evaluation. In NIPS. 2014. Available: https://arxiv.org/pdf/1404.0736

[21] Gokhale, V., Jin, Jonghoon, Dundar, A., Martini, B., and Culurciello, E. A 240 g-ops/s mobile coprocessor for deep neural networks. In CVPR Workshops. 2013.

[22] F. N. Iandola, S. Han, M. W. Moskewicz, K. Ashraf, W. J. Dally, and K. Keutzer, "SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and <0.5MB model size," *arXiv preprint arXiv:1602.07360*, 2016. Available: https://openreview.net/pdf?id=S1xh5sYgx

[23] S. Han, H. Mao, and W. Dally, "DEEP COMPRESSION: COMPRESSING DEEP NEURAL NETWORKS WITH PRUNING, TRAINED QUANTIZATION AND HUFFMAN CODING," ICLR, 2016. Available: https://arxiv.org/pdf/1510.00149

[24] A. Howard *et al.*, "MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications," Apr. 2017. Available: https://arxiv.org/pdf/1704.04861

[25] Y.-H. Chen, T. Krishna, J. S. Emer, and V. Sze, "Eyeriss: An Energy-Efficient Reconfigurable Accelerator for Deep Convolutional Neural Networks," *IEEE Journal of Solid-State Circuits*, vol. 52, no. 1, pp. 127–138, Jan. 2017, Available: https://ieeexplore-ieee-org.libweb.lib.utsa.edu/stamp/stamp.jsp?tp=&arnumber=7738524

[26] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-Efficient Learning of Deep Networks from Decentralized Data," *proceedings.mlr.press*, Apr. 10, 2017. Available: https://proceedings.mlr.press/v54/mcmahan17a/mcmahan17a.pdf

[27] Yiping Kang, Johann Hauswald, Cao Gao, Austin Rovinski, Trevor Mudge, Jason Mars, and Lingjia Tang. 2017. Neurosurgeon: Collaborative Intelligence Between the Cloud and Mobile Edge. In Proceedings of the Twenty-Second International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS '17). Association for Computing Machinery, New York, NY, USA, 615–629. Available: https://dl.acm.org/doi/pdf/10.1145/3093337.3037698

[28] J. Azar, A. Makhoul, M. Barhamgi, and R. Couturier, “An energy efficient IoT data compression approach for edge machine learning,” *Future Generation Computer Systems*, vol. 96, pp. 168–175, Jul. 2019, Available: https://doi.org/10.1016/j.future.2019.02.005

[29] Z. Ali, L. Jiao, T. Baker, G. Abbas, Z. H. Abbas and S. Khaf, "A Deep Learning Approach for Energy Efficient Computational Offloading in Mobile Edge Computing," in IEEE Access, vol. 7, pp. 149623-149633, 2019, doi: 10.1109/ACCESS.2019.2947053. Available: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=8866714

[30] W. Liu, B. Li, W. Xie, Y. Dai and Z. Fei, "Energy Efficient Computation Offloading in Aerial Edge Networks With Multi-Agent Cooperation," in IEEE Transactions on Wireless Communications, vol. 22, no. 9, pp. 5725-5739, Sept. 2023, doi: 10.1109/TWC.2023.3235997. Available: https://ieeexplore-ieee-org.libweb.lib.utsa.edu/document/10021296

[31] Q. Wei, Z. Zhou and X. Chen, "DRL-Based Energy-Efficient Trajectory Planning, Computation Offloading, and Charging Scheduling in UAV-MEC Network," 2022 IEEE/CIC International Conference on Communications in China (ICCC), Sanshui, Foshan, China, 2022, pp. 1056-1061, doi: 10.1109/ICCC55456.2022.9880711. Available: https://ieeexplore-ieee-org.libweb.lib.utsa.edu/document/9880711

[32] A. Cleary, K. Yoo, P. Samuel, S. George, F. Sun and S. A. Israel, "Machine Learning on Small UAVs," 2020 IEEE Applied Imagery Pattern Recognition Workshop (AIPR), Washington DC, DC, USA, 2020, pp. 1-5, doi: 10.1109/AIPR50011.2020.9425090. Available: https://ieeexplore-ieee-org.libweb.lib.utsa.edu/document/9425090

[33] P. K. Barik, S. Shah, K. Shah, A. Modi and H. Devisha, "UAV-Assisted Surveillance Using Machine Learning," 2022 Seventh International Conference on Parallel, Distributed and Grid Computing (PDGC), Solan, Himachal Pradesh, India, 2022, pp. 384-389, doi: 10.1109/PDGC56933.2022.10053282. Available: https://ieeexplore-ieee-org.libweb.lib.utsa.edu/document/10053282

[34] I. Martinez-Alpiste, P. Casaseca-de-la-Higuera, J. Alcaraz-Calero, C. Grecos and Q. Wang, "Benchmarking Machine-Learning-Based Object Detection on a UAV and Mobile Platform," 2019 IEEE Wireless Communications and Networking Conference (WCNC), Marrakesh, Morocco, 2019, pp. 1-6, doi: 10.1109/WCNC.2019.8885504. Available: https://ieeexplore-ieee-org.libweb.lib.utsa.edu/document/8885504

[35] Y. M. Park, Y. K. Tun and C. S. Hong, "Optimized Deployment of Multi-UAV based on Machine Learning in UAV-HST Networking," 2020 21st Asia-Pacific Network Operations and Management Symposium (APNOMS), Daegu, Korea (South), 2020, pp. 102-107, doi:

10.23919/APNOMS50412.2020.9236987. Available: https://ieeexplore-ieee-org.libweb.lib.utsa.edu/document/9236987

[36] M. A. Abd-Elmagid, A. Ferdowsi, H. S. Dhillon and W. Saad, "Deep Reinforcement Learning for Minimizing Age-of-Information in UAV-Assisted Networks," 2019 IEEE Global Communications Conference (GLOBECOM), Waikoloa, HI, USA, 2019, pp. 1-6, doi: 10.1109/GLOBECOM38437.2019.9013924. Available: https://ieeexplore-ieee-org.libweb.lib.utsa.edu/document/9013924

[37] Q. Dong, "Reinforcement Learning based Anti-UAV Three-dimensional Pursuit-evasion Game for Substation Security," 2024 5th International Conference on Mechatronics Technology and Intelligent Manufacturing (ICMTIM), Nanjing, China, 2024, pp. 224-227, doi: 10.1109/ICMTIM62047.2024.10629444. Available: https://ieeexplore-ieee-org.libweb.lib.utsa.edu/document/10629444

[38] Y. Ding, Z. Yang, Q. -V. Pham, Y. Hu, Z. Zhang and M. Shikh-Bahaei, "Distributed Machine Learning for UAV Swarms: Computing, Sensing, and Semantics," in IEEE Internet of Things Journal, vol. 11, no. 5, pp. 7447-7473, 1 March1, 2024, doi: 10.1109/JIOT.2023.3341307. Available: https://ieeexplore-ieee-org.libweb.lib.utsa.edu/document/10353003

[39] M. Aljohani, R. Mukkamala and S. Olariu, "Autonomous Strike UAVs in Support of Homeland Security Missions: Challenges and Preliminary Solutions," in IEEE Access, vol. 12, pp. 90979-90996, 2024, doi: 10.1109/ACCESS.2024.3420235. Available: https://ieeexplore-ieee-org.libweb.lib.utsa.edu/document/10576029

[40] Ismi Abidi, Vireshwar Kumar, and Rijurekha Sen. 2021. Practical Attestation for Edge Devices Running Compute Heavy Machine Learning Applications. In Proceedings of the 37th Annual Computer Security Applications Conference (ACSAC '21). Association for Computing Machinery, New York, NY, USA, 323–336. Available: https://doi-org.libweb.lib.utsa.edu/10.1145/3485832.3485909

[41] Petros Amanatidis, George Iosifidis, and Dimitris Karampatzakis. 2022. Comparative Evaluation of Machine Learning Inference Machines on Edge-class Devices. In Proceedings of the 25th Pan-Hellenic Conference on Informatics (PCI '21). Association for Computing Machinery, New York, NY, USA, 102–106. Available: https://doi-org.libweb.lib.utsa.edu/10.1145/3503823.3503843

[42] Kyle Hoffpauir, Jacob Simmons, Nikolas Schmidt, Rachitha Pittala, Isaac Briggs, Shanmukha Makani, and Yaser Jararweh. 2023. A Survey on Edge Intelligence and Lightweight Machine Learning Support for Future Applications and Services. J. Data and

Information Quality 15, 2, Article 20 (June 2023), 30 pages. Available: https://doi-org.libweb.lib.utsa.edu/10.1145/3581759

[43] Arash Heidari, Nima Jafari Navimipour, Mehmet Unal, and Guodao Zhang. 2023. Machine Learning Applications in Internet-of-Drones: Systematic Review, Recent Deployments, and Open Issues. ACM Comput. Surv. 55, 12, Article 247 (December 2023), 45 pages. Available: https://doi-org.libweb.lib.utsa.edu/10.1145/3571728

[44] Mukhtar N, Mehrabi A, Kong Y, Anjum A. Edge enhanced deep learning system for IoT edge device security analytics. *Concurrency Computat Pract Exper*. 2023; 35(13):e6764. doi:10.1002/cpe.6764

[45] Nimmagadda, Y. (2025). Training on Edge. In Model Optimization Methods for Efficient and Edge AI (eds P.R. Chelliah, A.M. Rahmani, R. Colby, G. Nagasubramanian and S. Ranganath). Available: https://doi-org.libweb.lib.utsa.edu/10.1002/9781394219230.ch11

[46] Zhang, M., Zhang, F., Lane, N.D., Shu, Y., Zeng, X., Fang, B., Yan, S. and Xu, H. (2020). Deep Learning in the Era of Edge Computing: Challenges and Opportunities. In Fog Computing (eds A. Zomaya, A. Abbas and S. Khan). Available: https://doi-org.libweb.lib.utsa.edu/10.1002/9781119551713.ch3

[47] Peñaranda C, Reaño C, Silla F. Exploring the use of data compression for accelerating machine learning in the edge with remote virtual graphics processing units. *Concurrency Computat Pract Exper*. 2023; 35(20):e7328. doi:10.1002/cpe.7328

[48] Wang Y, Meng W, Li W, Liu Z, Liu Y, Xue H. Adaptive machine learning-based alarm reduction via edge computing for distributed intrusion detection systems. *Concurrency Computat Pract Exper*. 2019; 31:e5101. Available: https://doi-org.libweb.lib.utsa.edu/10.1002/cpe.5101

[49] Ali, Elmustafa Sayed, Hasan, Mohammad Kamrul, Hassan, Rosilah, Saeed, Rashid A., Hassan, Mona Bakri, Islam, Shayla, Nafi, Nazmus
Shaker, Bevinakoppa, Savitri, Machine Learning Technologies for Secure Vehicular Communication in Internet of Vehicles: Recent Advances and Applications, *Security and Communication Networks*, 2021, 8868355, 23 pages, 2021. Available: https://doi-org.libweb.lib.utsa.edu/10.1155/2021/8868355

[50] Raad, H. (2020). Cloud and Edge. In Fundamentals of IoT and Wearable Technology Design, H. Raad (Ed.). Available: https://doi-org.libweb.lib.utsa.edu/10.1002/9781119617570.ch7

[51] Perumalla, M.M.R., Singh, S.K., Khamparia, A., Goyal, A. and Mishra, A. (2020). Machine Learning Frameworks and Algorithms for Fog and Edge Computing. In Fog, Edge,

and Pervasive Computing in Intelligent IoT Driven Applications (eds D. Gupta and A. Khamparia). Available: https://doi-org.libweb.lib.utsa.edu/10.1002/9781119670087.ch4

[52] Nimmagadda, Y. (2025). Model Optimization Techniques for Edge Devices. In Model Optimization Methods for Efficient and Edge AI (eds P.R. Chelliah, A.M. Rahmani, R. Colby, G. Nagasubramanian and S. Ranganath). Available: https://doi-org.libweb.lib.utsa.edu/10.1002/9781394219230.ch4

[53] Diego Méndez, Daniel Crovo, Diego Avellaneda. (2024). Chapter 15 - Machine learning techniques for indoor localization on edge devices: Integrating AI with embedded devices for indoor localization purposes, TinyML for Edge Intelligence in IoT and LPWAN Networks, Academic Press, Pages 355-376, ISBN 9780443222023, Available: https://doi.org/10.1016/B978-0-44-322202-3.00020-8.

[54] A. Guna, P. Ganeriwala and S. Bhattacharyya, "Exploring Machine Learning Engineering for Object Detection and Tracking by Unmanned Aerial Vehicle (UAV)," 2024 International Conference on Machine Learning and Applications (ICMLA), Miami, FL, USA, 2024, pp. 1001-1004, doi: 10.1109/ICMLA61862.2024.00149. Available: https://ieeexplore-ieee-org.libweb.lib.utsa.edu/stamp/stamp.jsp?tp=&arnumber=10903281

[55] J. Xu, Q. Guo, L. Xiao, Z. Li and G. Zhang, "Autonomous Decision-Making Method for Combat Mission of UAV based on Deep Reinforcement Learning," 2019 IEEE 4th Advanced Information Technology, Electronic and Automation Control Conference (IAEAC), Chengdu, China, 2019, pp. 538-544, doi: 10.1109/IAEAC47372.2019.8998066. Available: https://ieeexplore-ieee-org.libweb.lib.utsa.edu/stamp/stamp.jsp?tp=&arnumber=8998066

[56] Hongxing Zhang, Hui Gao, and Xin Su. 2020. Channel prediction based on adaptive structure extreme learning machine for UAV mmWave communications. In Proceedings of the 16th EAI International Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services (MobiQuitous '19). Association for Computing Machinery, New York, NY, USA, 492–497. Available: https://doi-org.libweb.lib.utsa.edu/10.1145/3360774.3368199

[57] Hengpeng Guo, Bo Zhang, and Yuanzhong Fu. 2025. Multi-Agent Deep Reinforcement Learning-Based UAV Trajectory Optimization for Data Collection. In Proceedings of the 3rd International Conference on Signal Processing, Computer Networks and Communications (SPCNC '24). Association for Computing Machinery, New York, NY, USA, 394–399. Available: https://doi-org.libweb.lib.utsa.edu/10.1145/3712335.3712404

[58] Chao Zhang, Haipeng Yao, and Tianle Mai. 2024. Graph Transformer Aided Resource Virtualization Embedding in UAV Swarm Networks. In Proceedings of the International Conference on Computing, Machine Learning and Data Science (CMLDS '24). Association

for Computing Machinery, New York, NY, USA, Article 36, 1–6. Available: https://doi-org.libweb.lib.utsa.edu/10.1145/3661725.3661763

[59] Le Qi and Wanyang Wang. 2024. Integrating Deep Learning Techniques for Enhanced Multi-Target Tracking in UAV Fire Control Systems. In Proceedings of the 2024 9th International Conference on Cyber Security and Information Engineering (ICCSIE '24). Association for Computing Machinery, New York, NY, USA, 865–870. Available: https://doi-org.libweb.lib.utsa.edu/10.1145/3689236.3696038

[60] Islam Güven and Evsen Yanmaz. 2023. Maintaining Connectivity for Multi-UAV Multi-Target Search Using Reinforcement Learning. In Proceedings of the Int'l ACM Symposium on Design and Analysis of Intelligent Vehicular Networks and Applications (DIVANet '23). Association for Computing Machinery, New York, NY, USA, 109–114. Available: https://doi-org.libweb.lib.utsa.edu/10.1145/3616392.3623414

[61] Ning Wu, Li Li, Xingyu Liu, Ziwen Wang, and Tongyao Jia. 2024. Enhancing UAV Swarm Routing with Multi-Agent Attention Reinforcement Learning. In Proceedings of the 2023 13th International Conference on Communication and Network Security (ICCNS '23). Association for Computing Machinery, New York, NY, USA, 258–264. Available: https://doi-org.libweb.lib.utsa.edu/10.1145/3638782.3638822

[62] Yanfan Zhang, Hongyuan Zheng, and Xiangping Zhai. 2023. Deep Reinforcement Learning Based UAV Mission Planning with Charging Module. In Proceedings of the 2023 4th International Conference on Computing, Networks and Internet of Things (CNIOT '23). Association for Computing Machinery, New York, NY, USA, 658–662. Available: https://doi-org.libweb.lib.utsa.edu/10.1145/3603781.3603897

[63] Tuan Do Trong, Tran Bao Duy, Vu Dinh Khai, and Hoang-Anh Pham. 2023. Applying Deep Learning for UAV Obstacle Avoidance: A Case Study in High-Rise Fire Victim Search. In Proceedings of the 12th International Symposium on Information and Communication Technology (SOICT '23). Association for Computing Machinery, New York, NY, USA, 831–837. Available: https://doi-org.libweb.lib.utsa.edu/10.1145/3628797.3628813

[64] Zhao, Y., Nie, Z., Dong, K., Huang, Q. and Li, X., 2024. Autonomous Decision Making for UAV Cooperative Pursuit-Evasion Game with Reinforcement Learning. Available: *https://arxiv.org/html/2411.02983v1*

[65] Jun Zhang and Khaled B. Letaief. 2019. Mobile edge intelligence and computing for the internet of vehicles. Proceed ings of the IEEE 108, 2 (2019), 246–261.

[66] Jihong Park, Sumudu Samarakoon, Mehdi Bennis, and Mérouane Debbah. 2019. Wireless network intelligence at the edge. Proceedings of the IEEE 107, 11 (2019), 2204–2239.

[67] X. Wang, Z. Tang, J. Guo, T. Meng, C. Wang, T. Wang and W. Jia, "Empowering Edge Intelligence: A comprehensive Survey on On-Device AI Models," *ACM Comput. Surv.,* vol. 57, no. 9, pp. 228:1 - 228:39, April 2025.

[68] Yuanming Shi, Kai Yang, Tao Jiang, Jun Zhang, and Khaled B. Letaief. 2020. Communication-efficient edge AI: Algo rithms and systems. IEEE Communications Surveys & Tutorials 22, 4 (2020), 2167–2191.

[69] Yueyue Dai, Ke Zhang, Sabita Maharjan, and Yan Zhang. 2020. Edge intelligence for energy-efficient computa tion offloading and resource allocation in 5G beyond. IEEE Transactions on Vehicular Technology 69, 10 (2020), 12175–12186.

[70] G. Kendall, "Real Clear Science," RealClear Media Group, 02 07 2019. [Online]. Available: https://www.realclearscience.com/articles/2019/07/02/your_mobile_phone_vs_apollo_11s_guidance_computer_111026.html. [Accessed 19 04 2025].

[71] S. S, U. A, P. N. Srivatsa, S. B and S. S, "Artificial Intelligence-based Voice Assistant," in *Smart Trends in Systems, Security, and Sustainability*, 2020.

[72] Linjuan Ma and Fuquan Zhang. 2021. End-to-end predictive intelligence diagnosis in brain tumor using lightweight neural network. Applied Soft Computing 111 (2021), 107666.

[73] He, K., Zhang, X., Ren, S., & Sun, J. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 770-778).

[74] Y. Roh, G. Heo and S. E. Whang, "A Survey on Data Collection for Machine Learning: A Big Data - AI Integration Perspective," *IEEE Transactions on Knowledge and Data Engineering,* vol. 33, no. 4, pp. 1328-1348, 2021.

[75] Tae, K. H., Roh, Y., Oh, Y. H., Kim, H., & Whang, S. E. 2019. Data cleaning for accurate, fair, and robust models: A big data-AI integration approach. In *Proceedings of the 3rd international workshop on data management for end-to-end machine learning* pp. 1-4.

[76] B. Zoph and Q. V. Le, "Neural Architecture Search With Reinforcement Learning," in *ICLR*, 2017.

[77] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child,

A. Ramesh, D. M. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever and D. Amodei, "Language Models are Few-Shot Learners," Open AI, 2020.

[78] "LiteRT overview," Google, March 4, 2025. [Online]. Available: https://ai.google.dev/edge/litert. [Accessed April 2025].

[79] "Caffe2," Facebook, [Online]. Available: https://caffe2.ai/docs/caffe-migration.html. [Accessed April 2025].

[80] "Apache MXNet," 2022. [Online]. Available: https://mxnet.apache.org/versions/1.9.1/. [Accessed April 2025].

[81] F. Dong, D. Shen, Z. Huang, Q. He, J. Zhang, L. Wen and T. Zhang, "Multi-Exit DNN Inference Acceleration Based on Mult-Dimensional Optimization for Edge Intelligence," *IEEE Transactions on Mobile Computing,* vol. 22, no. 9, pp. 5389 - 5406, 2023.

[82] S. Gao, F. Huang, J. Pei and H. Huang, "Discrete Model Compression with Resource Constraint for Deep Neural Networks," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Seattle, WA, 2020.

[83] "Coral," Google, 2020. [Online]. Available: https://coral.ai/products/. [Accessed April 2025].

[84] Seyedehfaezeh Hosseininoorbin, Siamak Layeghy, Brano Kusy, Raja Jurdak, Marius Portmann, “Exploring Edge TPU for deep feed-forward neural networks”, in Internet of Things, vol. 22, 2023. Available: https://doi.org/10.1016/j.iot.2023.100749

[85] "Robotics and Edge AI," Nvidia, 2025. [Online]. Available: https://www.nvidia.com/en-us/autonomous-machines/embedded-systems/. [Accessed April 2025].

[86] D. Narayanan, A. Harlap, A. Phanishayee, V. Seshadri, N. R. Devanur, G. R. Ganger, P. B. Gibbons and M. Zaharia, "PipeDream: Generalized Pipeline Parallelism for DNN Training," in *SOSP*, Huntsville, ON, Canada, 2019.