A2. Annotated Bibliography

[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

Comment:

This paper is one of the earlier works addressing the challenge of adapting deep learning models for mobile devices. It introduces a software framework that focuses exclusively on the inference step, optimizing it for the end user.

A key innovation of this framework is its dynamic adaptation to changing mobile environments. It employs a pair of algorithms that compress and partition the model in real time with each inference request, based on the device's currently available resources. This allows the framework to adjust its behavior as conditions fluctuate, making it highly efficient and responsive.

To conserve energy while maintaining performance, the framework prioritizes low-power processors (such as LPUs) over high-performance counterparts (like GPUs) whenever possible. One of the core design principles is leveraging a deep learning prediction model to intelligently plan each execution cycle. This prediction model is continually updated with actual performance costs after every inference task, ensuring that the planning process becomes increasingly accurate over time.

The framework introduces two novel algorithms to control resource utilization:

1. **Runtime Layer Compression (RLC):** This algorithm compresses individual layers during inference using Singular Value Decomposition (SVD) without requiring model retraining. RLC can predict the accuracy loss resulting from compression without directly testing the model.
2. **Deep Architecture Decomposition (DAD):** This algorithm breaks down models into smaller unit-blocks, which are then assigned to available local and remote processors. The goal is to enhance energy efficiency and reduce latency by distributing the computational load effectively.

These two algorithms work together to create an optimal execution plan for every inference request. DAD is in charge of planning and explores multiple configuration options based on resource constraints, while RLC provides feedback on the potential accuracy loss from compression.

[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

Comment:

This paper provides a comprehensive overview of techniques and methodologies aimed at enhancing the energy efficiency of ML models deployed on edge devices. The authors emphasize that a holistic approach is needed to optimize energy efficiency in edge devices. To assist in writing energy efficient code, they also introduced a tool called JEPO to aid Java development.

Two categories are discussed in hardware architecture advancements. First, they examine a number of accelerator architectures such as GPUs, FPGAs, and ASICs. Notable examples include NVIDIA's Jetson platforms and Google's Edge TPU, which offer enhanced performance while reducing energy consumption. Second, they explore a number of co-design approaches that integrate hardware and software optimizations. Techniques such as model pruning and quantization are discussed in the context of their implementation on hardware platforms to achieve significant energy savings.

The software and packages section delves into software platforms and frameworks optimized for ML on edge devices. A number of software platforms are highlighted, including TinyOS, Phi-Stack, OpenEI, and OpenVDAP. They review several ML libraries and frameworks designed for use on the edge. Some libraries handle both training and inference, while others are designed only for the latter.

In their discussion of algorithms, they broadly categorize design approaches to reducing computational requirements and reducing the accuracy of operations and operands. Highlighted examples of the former category include SqueezeNet, MobileNets, ESPNetv2, and Bonsai. For each model they discuss the optimization techniques used. An analysis of several quantization techniques is presented for the latter category.

[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

Comment:

Introduces an optimized machine learning model based on SqueezeNet and a lightweight hardware accelerator for use in real-time object detection. With increased desire to deploy deep learning to edge devices, the authors emphasize the need to approach implementation holistically. First, by creating models that are compact and optimized for the specific hardware. Second, by designing energy efficient hardware accelerators for the machine learning task.

Their methodology further compresses SqueezeNet for this application in a two-part fashion. They start by simplifying the model architecture. They don't discuss all the details of the methodology for doing this. They start by reducing the model to four fire modules from the original eight, but do not mention how they came up with this particular configuration. Additionally, they

mention that they embed maxpool functions into convolutional layers by increasing stride length to 2 in the middle of the convolution operation. For several convolution operations they adopt a stride of 2 for the entire operation. These maxpool optimizations achieve lower latency and less computation, but again, they don't discuss how they came upon these particular choices or tradeoff considerations. A final simplification is to reduce channel number to four. This helps tailor the model to the hardware.

After simplification, they compress the model by applying quantization to activations and parameters. They present a symmetric quantization formula which they use for weight parameters. There is a second asymmetric quantization formula they use for activations, but don't mention the formula besides that additional calculations using the zero point are required.

The proposed hardware architecture is characterized by a 3D tensor-like processing element (PE) that enables parallel processing between layers and channels. It also employs a on-chip memory management strategy that minimizes the need for large on-chip memory.

Results show improvement in logic size, memory size, and power consumption compared to similar works. Tests were run on object detection tasks. Most importantly, the system was able to operate at ~44 frames per second, demonstrating its ability to perform in real-time.

[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

Comment:

This paper presents power consumption and efficiency results for well-known optimization methods applied to established ML models. The researchers identified a gap in empirical power consumption data for these methods. They present results for uncompressed and compressed models on two different microcontroller architectures with an additional metric of idle power consumption of the hardware to define the power overhead created by the models.

The experimental setup used included two commonly used development boards, the ESP32-DevKit by Espressif and STM32H743-nucleo from STMicroelectronics. Both boards were chosen because of their relatively large memory (for a microcontroller) to accommodate uncompressed ML models without requiring any external memory. They provide the lab setup in great detail in order to make the study reproducible as well as present the strict control over external factors such as measurement accuracy. ML models included LeNet-5, a simple sine calculation model, MobileNet 025, and an AI based indoor localization model. The software framework used for model compression was TensorFlow Micro (or TinyML), which is specifically designed to be used on resource-constrained devices.

Results of the study show that there are factors besides model size that affect power consumption. Notably, ML models with reduced memory fetching requirements demonstrate better power efficiency even when their overall size is larger. This indicates that special attention

should be paid to memory handling efficiency in programs. Compression techniques, such as quantization, have the effect of decreased reliance on core subsystems in the hardware, like floating-point units, which also leads to enhanced power efficiency. The energy performance between devices was also significantly different, emphasizing the need to use a holistic approach when deploying ML models on edge devices.

Conclusions of the study indicate that a nuanced approach is required, balancing model performance with power efficiency. Key data indicates that SRAM usage is an important indicator of power efficiency and they advocate for further research on a broader set of models and software frameworks to extend these conclusions and explore other factors affecting power efficiency.

[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

Comment:

This study provides a comprehensive analysis of optimizing DNN inference on edge devices from the perspective of the developer. A three-part tradeoff is considered between accuracy, energy consumption, and latency. They introduce a metric of accuracy-per-joule to compare accuracy and energy efficiency, as well as a recommendation algorithm to assist in the best compromise of choosing a specific DNN model for specific hardware settings and constraints.

In their analysis, they evaluate over 40 popular DNN models in terms of energy footprint, latency, and prediction accuracy while also trying to address two hypotheses: Larger DNN models provide more accuracy at the expense of energy expenditure and energy expenditure can be reduced by lowering GPU speed at the cost of greater latency.

The metrics used to indicate model size are number of model parameters and number of floating point operations (FLOPs). They begin experiments by comparing accuracy vs model size. Generally, accuracy increases with model size but at a decreasing rate. Interestingly this rule is not absolute across different families of models. For example VGG models are larger than ResNet and EfficientNet, yet yield lower accuracies. Similarly, larger models are not always slower or consume higher energy. EfficientNet models are smaller, provide high accuracy, and consume relatively high energy. The data suggests that trends of size to accuracy, size to latency, and size to energy consumption should be constrained to models in the same family. Within families, the findings show that larger models have higher accuracy at the cost of higher latency and energy cost. Another notable finding is that FLOPs is a better metric to characterize model size than number of parameters. The correlation is stronger between FLOPs vs latency and FLOPs vs energy.

Building upon the earlier introduced accuracy-per-joule metric, the recommendation algorithm considers constraints on accuracy, energy consumption, and latency to identify suitable models. The authors advocate for the use of the accuracy-per-joule metric and recommendation algorithm as useful tools to aid in the decision-making process.

[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

Comment:

The authors perform a comprehensive three phase study to accurately measure, predict and score energy consumption of kernels, state-of-the-art (SOTA) DNN models, and popular edge AI applications. They highlight the lack of robust energy data for each of the above classes as a gap for needed exploration. To assist the end user with a non-technical background, they present two novel scoring metrics: power consumption score (PCS) and inference energy consumption score (IECS).

During the measurement portion of the study the authors measure the energy consumption of 16 kernel types, nine SOTA DNN models with 50 variants each, and six popular edge AI applications. Since it is impractical to measure every possible DNN model, they determine the trend of energy consumption according to the following configuration variables from the kernel measurements: height and width (HW), input channel number (C_{in}), output channel number (C_{out}), kernel size (KS), and stride (S). Based on the trends from the kernel variables, they are able to extend energy consumption predictions to untested models. To make the prediction process user-friendly, they extend the nn-meter from “NN-meter: Towards accurate latency prediction of deep-learning model inference on diverse edge devices” by Zhang et al. to incorporate the kernel variables listed above. The resulting predictor achieves an accuracy of 86.2%, surpassing previous prediction models.

The proposed scoring metrics are designed to present an intuitive energy efficiency understanding of a particular device for the non-technical end user. Power consumption score (PCS) is defined as 1 minus the ratio of average power consumption to max device power of a device tested on the six edge AI applications. So, a higher PCS indicates higher energy efficiency. Inference energy consumption score (IECS) shows the average number of inferences per unit energy for each device. Higher IECS indicates better inference performance per unit energy.

[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

Comment:

The authors identified that with the proliferation of resource-intensive computing on edge devices there is a growing reliance on edge servers to share the workload. Instead of addressing this need with more servers, this paper presents a DEWOrch, a deep reinforcement learning

algorithm designed to optimize workload orchestration by prioritizing inter-device task sharing when possible.

An overview of reinforcement learning and deep reinforcement learning is given to provide a foundational understanding of the class of models that learn optimal policies through interaction with the environment. Reinforcement learning can be modeled with a Markov decision process that track rewards gained from state-action pairs in a Q table. DRL extends this concept, replacing the Q table with a neural network. The challenges inherent in DRL for edge devices include the following:

1. State representation depends on the attributes of each task generated. Since multiple devices are generating tasks asynchronously, often with identical attributes, it is important to be able to identify new states from old states.
2. The delayed environment of networked devices leads to delayed feedback for each action taken.

The model must be able to handle these challenges.

The study focuses on collaborative capabilities of edge devices. The system modeled for the study is a network of multiple heterogeneous devices without edge servers. Key components of the system include task generators, varying computational capacities per device, and a central orchestrator responsible for task allocation and resource management.

The DRL model is applied to the workload orchestration problem which is formulated as a multi-objective optimization challenge. The objectives are to maximize task success rate and minimize resource waste. Experimental metrics are defined as resource utilization, task completion, and energy consumption. Other factors for resource selection include availability, reliability, and response time for each device. The model uses these factors to prevent assignments with a high likelihood of failure, thereby reducing waste incurred by reallocation.

The experimental setup is a set of simulation scenarios for a smart manufacturing environment. The environment consists of microcontrollers and sensors in a simulated industrial complex that includes employees with edge devices (such as smartphones, tablets, etc.) that are distributed throughout the complex. They compare DEWOrch with two other DRL models: AdWOrch and DeepEdge. DEWOrch shows improved energy efficiency and less resource waste compared to the other two models. Resource waste can be thought of as assigning simple tasks to devices with high computational capacity when a smaller device would suffice. AdWOrch excelled at task success rate at the expense of greater resource waste. The authors argue that prioritizing resource-constrained devices when possible leads to enhanced energy efficiency. By extension, they conclude that this would lessen the need for more edge servers in these environments.

[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

Comment:

This paper discusses the challenges of computational offloading in industrial internet of things (IIoT) systems. It emphasizes the need for energy-efficient offloading strategies to enhance system performance and prolong device lifespan. They highlight the shortcomings of many proposed strategies and propose a multi-agent deep reinforcement learning approach to address the gaps in current research.

The related work section highlights many previous approaches to the offloading problem and the authors discuss the shortcomings of each approach. Later approaches similarly introduce multi-agent deep learning implementations, but the main drawbacks identified are prioritizing energy efficiency over system latency and jointly prioritizing energy efficiency and system latency without regard to individual task deadlines. In many systems, missing task deadlines can have catastrophic consequences such as in nuclear power plants. The authors' proposed system seeks to address this gap.

The proposed model in this paper is formulated as a Markov decision process. States are represented as the current status of the device, available resources, and task queues. Actions are defined as decisions to either process locally or offload the task to other devices or edge servers. Rewards are based on two criteria – first completion of the task before the deadline and second ensure energy efficiency. By prioritizing the deadline, the reward structure aims to minimize energy consumption in the long run while ensuring tasks are always completed on time.

The DRL agent model is initially trained in the cloud and a duplicate of the trained model is assigned to edge device and sent to the corresponding edge servers. Edge servers can be assigned to multiple devices. Once deployed, agents can communicate with each other and their assigned device to collaborate in task offloading decisions. Collaboration allows tasks to be offloaded to any edge server, ensuring effective load-balancing and adaptation to dynamic environments.

Comparing the proposed model to others in terms of energy efficiency, an exhaustive search algorithm always finds the optimal offloading scheme. However, compared to the model in this paper, the exhaustive search is three orders of magnitude slower. This model finds better configurations than the other models but is an order of magnitude slower (milliseconds vs sub-millisecond). In terms of task completion before deadline, the model in this paper outperforms all of the other models. Additionally, the model exhibits better weak scalability, showing a similar task completion rate between small and large scale simulations.

[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

Comment:

Machine learning on edge devices faces significant constraints. Traditionally, complex models are offloaded to the cloud for inference, but his presents a concern for data-privacy. The

authors identified that in most cases users only utilize a subset of available tasks on a device. This paper presents a novel method to prune large generic models in such a way that they are tailored to each user's needs, which they call MyML. Via this technique, the models are compressed and can run on devices, improving energy efficiency and data-privacy, as well as addressing network connectivity issues.

Another approach that addresses data-privacy in ML models is called Federated Learning. This method deploys models on edge devices, where training and inference is performed. To ensure robust training, the devices share parameter updates with a global model. This takes advantage of the variety of data created and collected by the entire group of users. The averaged weights calculated by the global model are then sent back to every device. For a large model to be deployed on edge devices, it must first be compressed. In the case of Federated Learning there is a tradeoff in model size versus accuracy. The approach in this paper seeks to maintain the level of feature extraction performed in early layers of the original model and prunes the later layers involved in classification. The result is a compressed model that retains a level of accuracy more similar to the original.

Because user preferences can by dynamic, MyML also tracks preferences based on prediction probability and entropy over the probability distribution. Based on an estimate of divergence, it determines when to discard the tailored model and restart the tailoring process.

Building the user model is based on transfer learning, where a previously trained generic model is fine-tuned to a specific task. In this method, the original classification layer is replaced with one for the new task and the model is tuned for the new dataset. The building process in this paper is discussed in three phases. First, the learning phase keeps track of the most frequently identified classes in an initial batch of user data. A tunable parameter is set as a threshold for determining which classes to keep. Second, the pruning phase prunes the model based on the user classes identified. A minimum threshold is set, defined as maintaining at least 99% accuracy compared to the original model. Last, the pruned model is adopted as the working model in the device until MyML determines that a new model must be built.

The paper evaluates several pruning methods and considerations. A bottom-up method is adopted because it allows for simpler computations in the building process. They detail the process used to evaluate accuracy while iteratively pruning each layer. In addition, they detail the techniques used during the process. Finally, they detail how they repurpose Edge TPU for speeding up the backward pass during the pruning phase.

[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

Comment:

There is a growing importance placed on deploying ML models to edge devices. This paper highlights the design challenges present for these types of applications, such as power consumption, network traffic, limited hardware resources, and privacy concerns. To address these challenges, the authors introduce a hardware accelerator designed for ML deployment in edge devices called Intelligence Boost Engine (IBE).

For background, they highlight the performance and power consumption differences between three classes of devices. Cortex-M series devices are the lowest powered and lowest performing devices considered. Clock frequency for these devices is typically in the 10-100 MHz range and use-cases are generally limited to wearable sensors and smart components (i.e. programmable LEDs, smart thermostats, etc.). Compared to the next class, there is about a 100x difference in power consumption. Cortex-A series devices typically run in the neighborhood of several GHz. This class is typically used in mobile devices capable of running applications (i.e. phones, smart watches, etc.). The final class considered is that of desktop/laptop cpus, which have a similar clock speed to Cortex-A devices but consume 100x more power.

To bridge the performance gap between Cortex-M devices and the others, other research pathways have developed highly specialized hardware accelerators, such as application-specific integrated circuits (ASICs). The downside of these devices is their lack of flexibility. More flexible hardware accelerators have been developed (like Eyeriss) at the cost of higher energy consumption. The developers of IBE sought to optimize both performance and energy efficiency while maintaining flexibility. To accomplish this they identified common kernel operations shared by target applications for optimizing their hardware and also provide limited programmability to allow for some level of flexibility.

The target applications for IBE are sensor fusion and motion recognition. Multiply-accumulate (MAC) operations account for the bulk of kernel operations performed in the learning process for these applications. More specifically, these operations consist of matrix-matrix multiplication, scalar product, scalar addition, and power operations. IBE consists of 12 MAC units and 1 kb of memory. For comparison, the highly flexible Eyeriss v2 chip has 384 MACs and 246 kb of memory at a higher power consumption cost. To further boost performance on IBE with its relatively small footprint, a special zero-skipping method was devised to optimize multiplication of sparse vectors.

Six programmable operating modes are included in IBE. The first four modes are considered general purpose and can reliably support various algorithms. The last two modes are optimized for the target applications in the paper.

Testing of IBE was conducted by integrating it onto a SHL-200 test chip with a Cortex-M4F core. For a baseline, target applications were run using only the Cortex-M4F core. The same applications were then run using IBE. Kernel execution speedup varied by application. The sensor fusion application saw a kernel execution speedup of 3.5 and 4.3X (there were two versions), with an overall application speedup of 1.9 and 2.5X, respectively. The motion recognition applications saw a more dramatic speedup. LIBSVM linear saw a 69.9X kernel execution speedup and 3.5X

overall speedup. LIBSVM polynomial resulted in 31.8X kernel execution speedup and 4X overall speedup. For the sake of conciseness, the applications saw 45 – 75 % energy reduction. Additionally, there was a 300X network communication reduction in the motion recognition algorithm.

The conclusion summarizes the effectiveness of IBE for ML applications on edge devices. The authors also suggest future research in adopting similar hardware for DNN applications.

[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

Comment:

This paper presents a comprehensive analysis of performance limitations on current edge ML accelerators. The study was motivated by the observation that Google Edge TPU's performance varies considerably across different families of NN models.

For background, the paper highlights several types of NNs, including CNNs, LSTMs, and Recurrent CNNs (RCNNs). The study evaluates performance for 24 Google edge NN models. The Edge TPU operates significantly below its peak computational throughput due to mismatches between its monolithic design and the diverse computational patterns of different layer configurations. The device achieves only a fraction of its theoretical energy efficiency (only 37% in the best case). The memory architecture emerges as a significant constraint, with frequent off-chip memory accesses. These shortcomings underscore the challenges of a one-size-fits-all hardware design in accommodating heterogeneous NN architectures.

In response to the findings, the authors propose "Mensa", a heterogeneous acceleration framework with multiple specialized accelerators. Based on the study, they found that layers naturally group into subsets comprising shared characteristics. With a proposed runtime scheduler, the Mensa framework is able to dynamically assign layers to an optimal accelerator. This makes the new framework more adaptable than generic hardware solutions.

There are three specialized accelerators included in Mensa. Pascal targets compute-intensive layers, maintaining high processing element utilization while reducing on-chip buffer sizes and network traffic. Pavlov is optimized for LSTM-like data-centric layers, employing dataflows that enhance parameter reuse and reduce off-chip memory access. Jacquard is designed for other data-centric layers, focusing on minimizing on-chip parameter buff sizes and exposing data reuse opportunities.

Compared to Edge TPU and Eyeriss v2, Mensa achieves a 3X improvement in energy efficiency, 3.1X increase in computational throughput, and 1.9X reduction in inference latency.

[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

Comment:

This paper introduces Edge2Train, a framework designed to enable resource-constrained edge devices retrain support ML models offline, specifically support vector machines (SVMs). ML models often see novel data patterns in the real world unlike the training data set. In such cases, inferences lead to false or less accurate results. Traditionally, retraining is outsourced to the cloud where high compute resources are available, but this comes with connectivity and privacy challenges. Edge2Train is presented as a solution to these challenges in model retraining.

The Edge2Train framework uses a background process to monitor inferences from the ML model. If an actuation is satisfactory, the routine is not disturbed. If not, a training set is generated by stitching together the model's input and output array with the ground truth obtained by observing real-world information. Once sufficient training sets are generated, the model is automatically retrained.

Edge2Train algorithms are implemented in C++ and tailored for efficiency and compatibility with various microcontroller units (MCUs). Learning is handled incrementally, allowing models to adapt to new data without requiring extensive computational power or memory. The entire library is contained in a single .h file consisting of only five functions, showcasing the lightweight nature of the framework. Additional operators are imported from the TensorFlow Micro Library. The paper provides pseudocode and explanation for the five functions from the library.

Five popular MCU boards are tested with the Edge2Train framework, assessing training time, inference speed, classification accuracy, and energy consumption. Results indicate that SVMs retrained with Edge2Train achieve classification accuracy close to those trained on high-resource machines. Notably, inference times are up to 3.5X faster than those on CPUs along with significant energy savings in three of the five MCUs tested. The training phase is faster and more efficient on the CPUs tested, however.

The benefits of Edge2Train are summarized as extending the ability of MCU-based edge devices to continuously improve locally and offline with real-world data. This extends the flexibility of ML deployment for scenarios when connectivity is limited and when data-privacy is a significant concern. Plans for future work includes additional algorithms and functions for multi-class SVM training and inference.

[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

Comment:

IoT devices are increasingly incorporated into new applications along with an increased desire to employ ML models to create smart devices. Due to the computational complexity of modern supervised machine learning techniques, there is often a tradeoff that has to be addressed between accuracy and computational cost for resource-constrained systems. This paper seeks to address this challenge by simplifying the early stages of data processing rather than focusing on model tuning and parameter adjustment.

The authors present a drinking activity monitoring system as a case study to analyze the performance of various supervised machine learning techniques. The methodology incudes an initial optimization process that selects the most important features from raw sensor data to enhance classification efficiency.

The methodology in the paper proposes an activity recognition system, utilizing publicly available accelerometer data with 14 labeled activities. For the study, the researchers binarize the classifications into drinking and non-drinking activities. They acknowledge the potential bias introduced due to the unbalanced classes, but still consider it a suitable framework for the study. Data preprocessing is undertaken by first filtering the signal to reduce spike and noise. Thereafter, the signal is segmented into five equal-length parts, with the a priori knowledge that drinking activity corresponds to a unique set of actions: grasping the glass and lifting to mouth level, tilting the glass, and returning the glass to the table. Feature selection is performed by evaluating and ranking combinations of features to determine the best combination for classification.

Evaluation is performed on two Raspberry Pi devices and a notebook CPU used as the baseline comparison. Seven different machine learning models are compared in the analysis. A test was performed for feature combination sizes in the range of 1 to 162. All models exhibit >90% accuracy given three or more best features, with minimal accuracy difference between both extremes of the range. Furthermore, small feature combinations significantly enhance computational latency, up to 5X when compared to using all features.

The authors emphasize the importance of understanding the initial data and identifying the most relevant signal characteristics when optimizing ML methods for resource-constrained systems. Experimental results reveal substantial performance gains through the data simplification methodology presented.

[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

Comment:

This paper presents an end-to-end code generator that facilitates the deployment of pre-trained ML models on edge devices using MicroTVM. This tool automates the conversion of models into C source libraries suitable for bare-metal devices.

There is a surge in ML deployment to resource-constrained edge devices. Popular software frameworks like PyTorch and TensorFlow are not directly compatible with bare-metal edge devices due to Python's just-in-time (JIT) compilation. There simply isn't room for the runtime libraries on the device. MicroTVM is an automated tool that converts Python code directly into C source code. The authors also introduce a Universal Modular Accelerator (UMA) interface, that allows developers to offload specific operator patterns to on-chip accelerators. Traditionally, this mapping process relies on the varied APIs provided by different backend chip vendors. UMA seeks to streamline implementation for the developer.

MicroTVM supports multiple input formats, including TensorFlow, TFLite, and ONNX. The output is structured C code, which is organized into directories containing headers, source files and compilation scripts, streamlining the entire process. At the time of this paper, MicroTVM is under active development so some operators in pre-trained models might not be supported. There is a detailed method to register new or custom operators by extending the frontend's conversion map.

The framework is evaluated on a hand gesture recognition case study. A pre-trained model is deployed on an ARM Cortex M4F using the generated C code, with some minor manual adjustments. Real-time inference is performed using an experimental glove with an integrated inertial measurement unit (IMU). The ARM chip is able to successfully recognize hand gestures from the experimental setup, showcasing the practicality of the framework.

MicroTVM is able to significantly simplify the deployment of ML models on edge devices, reducing the manual effort typically involved in such deployments. The experiment emphasizes the utility of the current work and underscores the need for future development.

[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

Comment:

This paper investigates the energy consumption profiles of object detection and motion tracking ML algorithms in edge devices. The authors seek to understand the factors that contribute to energy expenditure, including hardware specifications and tracking algorithms. Additionally, they discuss how various algorithm parameters can impact energy consumption so that researchers are

provided with a comprehensive explanation of the degrees of freedom made available by these edge devices.

There are a multitude of studies that investigate energy consumption of object detection and object tracking on edge devices. However, the authors identified that these two algorithms are typically performed independently and sequentially, each with their own resource demands. None of the previous studies cited explore the impact of the two algorithms working in tandem.

The experimental setup uses two of the most popular edge hardware accelerators, Google Coral AI and Nvidia Jetson Nano. Two common object detection models are used, MobileNet v2 and ResNet. Three common multi-object tracking algorithms are used, Intersection over Union (IOU), Simple Online and Realtime Tracking (SORT), and NvDCF. An external lab grade power supply was used to power the devices, and energy expenditure was measured via voltage drop across a sensing resistor placed in series with the device. Results first measured the energy overhead of each object detection algorithm, using default parameters, and then exploring energy consumption of various parameter configurations. To find the energy consumption of the tracking algorithms, the original detection configurations were remeasured with the addition of each tracking algorithm. The difference between the original detection energy measurement and the energy measurement from the detection + tracking setup determined how much additional overhead was created by the tracking algorithms.

The IOU tracker introduced the least amount of additional energy expenditure on both devices, with only 9% additional energy. The NvDCF tracker was configured for three different levels of accuracy, with 18%, 35% and 288% additional energy in the least to most accurate configurations, respectively. The most accurate configuration was not able to achieve real-time latency, with only 10 frames per second. The SORT algorithm added 64% additional energy when tested. Coral AI outperforms Jetson Nano for each of the setups, achieving a nearly 9% margin in many cases.

Parameter sensitivity reported for the IOU algorithm seems to be almost negligible. SORT also remains almost constant while varying parameters, but NvDCF appears to produce an exponential dependency of the energy consumed by the feature size parameter.

[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

Comment:

Several methods have been developed for training ML models intended for deployment on edge devices. Notable methods that train models directly on devices are transfer learning and federated learning, however these methods are limited to smaller models due to resource constraints. An alternative is device-cloud training, where the model is partitioned between the device and the more powerful cloud environment. This scheme suffers from a variety of challenges

including high communication latency, data privacy concerns and unstable network conditions. Edge servers have been introduced to bring less constrained resources closer to edge devices, improving latency. Yet another training method, takes advantage of edge servers by offloading model training entirely to the edge server. Again, this method suffers from privacy concerns and unstable network conditions. This paper introduces EdgeMove, which is similar in nature to device-cloud training, partitioning the model between the device and edge servers, but with a few key differences.

EdgeMove partitions model training between the edge device and edge servers with a few key enhancements. Initial layers are trained on the device and subsequent layers are handle by the edge server, with only activations and gradients exchanged between stages. This step helps to preserve data-privacy since user data is not exchanged similar to federated learning. One key optimization of EdgeMove is the implementation of pipelining, similar to PipeDream, which overlaps forward and backward propagation steps of smaller minibatches of data. Another enhancement is that EdgeMove monitors end-to-end training performance of multiple edge servers during runtime and dynamically adapts the training pipeline to utilize the fastest available resources. Unlike device-cloud training, which assumes a near limitless amount of computational resources in the network, EdgeMove is able to compensate for changing environmental resources.

Experiments compare EdgeMove with baseline on-device training, device-cloud training, and PipeDream-E which randomly picks a nearby edge server during the partitioning phase. EdgeMove outperforms all of the other methods, with a ~2.3X speedup over the baseline method. The data underscores the advantages presented by EdgeMove in a resource-constrained mobile environment.

[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

Comment:

Increasingly, object detection and image recognition ML models are being designed to accommodate very high-resolution images. These tasks are constrained by the available resources on edge devices, putting a cap on the scale of model sizes and individual input samples. This paper presents HiRISE, an in-sensor compression and selective region-of-interest (ROI) framework. The combined methodology seeks to adapt high-resolution inputs for compatibility with edge ML models.

HiRISE operates in two stages. First, the system performs in-sensor compression by converting RGB images to grayscale and applying analog pooling, reducing the image size before transferring it to the processing unit. In the second step, ROIs are identified in the compressed

image and the regions are mapped to the original high-resolution input. The selected regions are then extracted for processing in the network.

The study demonstrates that in-sensor scaling achieves similar accuracy compared to traditional in-processor scaling methods. The data reveals substantial reductions in memory utilization, data transfer, and energy consumption. For instance, HiRISE achieves up to 17.7X reduction in data transfer and energy usage. This methodology addresses critical challenges related to device resource constraints, paving the way for more sophisticated ML applications on edge devices.

[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

Comment:

Smart agriculture has increased with the ubiquitous nature of IoT devices. Applications include water allocation, targeted pest control, and environmental monitoring. Challenges in this domain are similar those in other IoT domains, including sparse connectivity, requirements of energy efficiency, latency concerns for automated machinery, and data privacy. However, due to the rural nature of agriculture, these challenges are amplified. This paper seeks to implement machine learning applications in agricultural IoT devices through a transfer learning framework with the assistance of unmanned aerial vehicles (UAV).

TinyML is a software framework developed on TensorFlow that uses optimization techniques to compress models so that they can be deployed on microcontrollers. The methodology proposed in this paper seeks to deploy TinyML-based models on field devices using UAVs as the delivery vector. The authors identified a gap in related research for this particular implementation method.

The next generation of smart farming, which the paper terms as farming 5.0, seeks to increase the productivity of farms while simultaneously decreasing the resources required. One of the ML solutions presented in this paper is a computer vision application to detect pests in fruit orchards. Another application uses image classification to detect diseases in grape leaves. This background establishes the utility of using ML technology in the agricultural domain, but the question remains on how best to deploy models and gather data from field devices when connectivity is sparse. The authors propose using UAVs to tackle this challenge.

Transfer learning is the process of applying a pre-trained model on different but related applications and data domains. Typically, this means pruning classification layers of the original model and fine tuning them for a specific task. UAVs in the scenario presented would act as sort of flying internet, collecting data from various devices, providing real-time training, and distributing the learned model.

The framework presented in this paper aims to highlight the strengths of the combined UAV-TinyML-TL approach. The authors develop a ML technique to accurately forecast soil humidity using both DNN and LSTM models and deploy the model to edge sensors in the field. The UAV starts by collecting initial data from the array of sensors, including soil humidity, air humidity, and air temperature. The data is shared between sensors via the UAV, each of which has the pretrained model stored in memory. Fine tuning then occurs on the model with the new dataset on each node before finally leading to the inference phase.

Metrics are presented inference time, accuracy, mean square error (MSE) and coefficient of determination (R^2). MSE shows the average prediction accuracy, while R^2 shows how effectively the model learns underlying patterns in the data compared to a baseline average prediction. Both metrics show convergence in the data. Consequently, the results demonstrate that the approach is effective. Future work in implementing more complex models and examining federated learning techniques is suggested by the authors.

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

Comment:

This paper is one of the earliest to explore means to compress CNN models in order to address storage constraints on mobile devices. The research considers a number of vector quantization methods and compares them to traditional matrix factorization methods in terms of compression rate and accuracy loss. The four quantization methods considered are binarization, product quantization (PQ), k means quantization (KM), and residual quantization (RQ). They found that PQ worked the best, achieving a 20X compression rate with an acceptable level of accuracy loss (within 1% drop). When compared to SVD matrix factorization, the best quantization methods were able to achieve higher compression rates, particularly in dense fully connected layers.

The paper recommends future research into hardware optimization methods for embedded devices and fine tuning to the quantization methods presented.

[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

Comment:

This is an often-referenced paper on model compression in CNNs for application in mobile computing platforms. The researchers observed that there is a high level of redundancy in model parameters which suggested that there may be opportunities for approximation of these parameters without sacrificing too much accuracy.

A number of approximation techniques are discussed, starting with selection of a good approximation criteria: Mahalanobis distance and data covariance distance. The former seeks to focus training on the input coordinates that contribute to the most error. The latter compares

covariance matrices of original and approximated filters when run over a representative dataset. This comparison provides insight into the impact of the approximation filters on the network.

In order to exploit redundancy, they utilize singular value decomposition (SVD) to deconstruct weight tensors into smaller matrices. This leads to a reduction in the number of parameters and computational operations. Additionally, they propose clustering of filters, by representing each cluster with a shared set of weights.

The implementation methodology used was to compress a layer and fine tune the following layers to restore accuracy before continuing to the next layer. This method is repeated for each layer that is being compressed. Experimental results show at least 2X weight reduction while maintaining less than 1% error increase for each layer tested. The testing was limited to fully connected convolutional layers.

[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.

Comment:

An early presentation of a low-power hardware accelerator for deep learning applications in mobile devices. The paper introduces nn-X (neural network next), a low-power coprocessor designed to enable real-time execution of DNNs on mobile devices.

Key features of the hardware include:

1. Scalability – The architecture is designed to scale efficiently to different DNN models, supporting convolution kernels up to 10x10
2. Low Power Consumption – Compared to other mobile processors of the time, nn-X was able to achieve ~ 3X power efficiency (25 G-ops/s-W compared to 8 G-ops/s-W)
3. Real-time processing capability – The computational speedup of nn-X compared to other mobile processors is significant for a variety of tasks (~100 to ~200x depending on the task). This is in large part due to the following factors: a) large parallelism, b) built-in pipelining and lack of conditional statements c) memory cache architecture optimized for deep learning workloads
4. Programmable hardware – This enhances the adaptability of nn-X to various DNN models

[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

Comment: Key strategies for compression: Replace 3x3 filters with 1x1 filters because they have 9x fewer parameters. Decrease number of input channels to the number of 3x3 filters. Delay downsampling to improve classification accuracy on a limited budget of parameters. The Fire module is the basic building block of SqueezeNet. It consists of a squeeze layer (only 1x1 filters) and an expand layer (a mixture of 1x1 and 3x3 filters), The three filter counts are tunable dimensions.

[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

Comment: A three-prong approach to compressing a model. Pruning weight connections below a certain threshold and retrain on the sparse weights. Trained quantization and weight sharing forces multiple connections to share the same weight and fine tune those weights to be represented in fewer bits. Use Huffman coding on quantized weights to further reduce memory overhead.

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

Comment: A class of efficient models for mobile and embedded **computer vision applications**. Core layers of MobileNet architecture: Depthwise separable convolutions that split a conventional convolution into two simpler operations. Width multiplier is a parameter to reduce the number of channels per layer. Resolution multiplier is a parameter to reduce input image resolution.

[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

Comment: A hardware accelerator designed for energy efficiency on the entire system running a DNN model. Main features include: A spatial architecture of 168 processing elements with a four-level memory hierarchy to minimize data access at high-cost levels. Row stationary – reconfigurable mapping to CNN shape. A reconfigurable communication architecture. Compression and data gating to reduce memory footprint

[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

Comment: An introduction to federated learning. A model is trained on local data and weight updates are communicated to a central server that averages weight updates from multiple devices. Local data is never shared between devices. Introduces the FederatedAveraging Algorithm – local stochastic gradient descent and model averaging on a central server, eliminating communication overhead between devices.

[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

Comment: Introduces a system to intelligently partition a DNN between the cloud and an edge device. Experiments were run on the AlexNet model. A performance analysis is done to determine data and computation characteristics of each layer and is combined with the communication latency between the cloud and the device. These metrics determine the ideal partitions for the model.

[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

Comment: An introduction of a technique to reduce communication overhead of IoT devices in a ML context. The paradigm consists of fast data compression technique at the IoT device before transmission to an intermediate edge device, where the data is decompressed and analyzed before transmission to the cloud. The results show a 103x reduction in data transmission for the IoT device, significantly lowering power consumption.

[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

Comment: Introduces a machine learning algorithm (EEDOS) for optimal computational offloading between a mobile edge device and the cloud. The algorithm considers a variety of conditions, including remaining battery life (possibly the first algorithm to do so), in picking the best set of partitions. The training dataset is exhaustive, which allows EEDOS to be 100% accurate. Thus, EEDOS outperforms previous approaches.

[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

Comment: This paper explores the problem of optimal computation offloading between mobile devices and a base station server with the support of UAVs that can aid in computation tasks or act as a communication relay. They leverage the concept of a digital twin edge network (DITEN) hosted on the base station to model the optimization process before deploying a solution to the devices on the network.

[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

Comment: An introduction of a deep reinforcement learning framework for optimizing multiple objectives in UAV operations for mobile edge computing networks. A priority mechanism is built into the learning framework to handle complex decision making. The objectives are energy efficiency, trajectory planning, communication scheduling, charge scheduling and task offloading.

[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

Comment: A paper on the workflow design of incorporating machine learning models on small UAVs using consumer-off-the-shelf parts and free-open-source software. ML model focuses on target detection and interfaces with Draper UAV flight software. Compatible with CPU/GPU and CPU only systems.

[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

Comment: There is some interesting related work cited in this paper. This paper showcases the ML model and related dataset used in suspicious object detection. It also contributes a resource allocation algorithm for creating communication links between surveillance drone, anchor drone and base station. The training data for object detection may be questionable as the example image is at ground level.

[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

Comment: This paper tests various object detection CNNs run on mobile devices to detect objects in streaming video from commercial-off-the-shelf UAVS. They chose to focus on the scenario where the ML model is not run directly on the UAV. Metrics tested are accuracy, processing speed, and resource consumption.

[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

Comment: In Korea, Rail-side units (RSUs) act as edge servers to maintain 5G cellular connectivity for passengers. Frequent handovers between RSUs represent a significant communication overhead. The authors present a reinforcement learning model to optimize the deployment of multiple UAVs to maintain longer, stable connections with criteria for UAV energy efficiency, trajectory, and altitude.

[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

Comment: Another approach of optimizing UAV trajectory and scheduling of status updates for UAVs for UAV-assisted wireless networks. This one focuses on minimizing age-of-information for critical information updates along with the normal constraints of power consumption and optimal flight trajectory. Also uses a deep reinforcement learning model to optimize the state space similar to other papers. Results show improvement over distance-based and random walk models.

[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

Comment: To enhance defensibility of power substations from UAV attack, this paper presents a deep reinforcement learning model to simulate a 3-dimensional pursuit evasion game, learning optimal interception strategies. They show improved interception success rates with the model.

[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

Comment: A paper investigating the complex task of deploying a UAV swarm to intelligently address multiple objectives. They present several deep learning frameworks such as Federated Learning, Multi-Agent Reinforcement Learning, Distributed Inference, and Split Learning. Multiple UAV swarm applications are investigated as well as areas for future research.

[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

Comment: An analysis of the challenges inherent in deploying autonomous UAVs for strike missions. The paper discusses the DRL model for training the UAV, smart contract and blockchain

technology for ensuring data integrity, security, and auditability, and a simulation model to develop a synthetic data set for such missions.

[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

Comment: A software attestation tool to protect against malicious network attacks during runtime. Since computation overhead is a significant constraint in EdgeML, they design the tool to run integrity checks on small chunks of the software at random intervals to minimize runtime interruptions and combat attacks designed to run between attestation events. PracAttest achieves 50x-80x speedup over previous state-of-the-art baselines.

[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

Comment: A performance analysis of running the inference step of an object classification ML model on three different edge device setups. Measurement metrics include throughput (fps), execution time, accuracy, energy efficiency, and total power consumption. There is discussion on cost considerations between hardware and the difficulty of implementing the ML model on each setup.

[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

Comment: A survey paper that assesses current cloud computing limitations, domains for edge computing growth, lightweight ML algorithms, and potential future directions in edge intelligence.

[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

Comment: A survey paper on drone swarm applications, an identification of the predominant use of CNNs in applications, and the common use of Python as the programming language of choice in drone applications.

[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

Comment: This paper focuses on implementing ML models on IoT edge devices to identify side-channel attack vulnerabilities. Specifically, they target the Elliptic-Curve Cryptographic (ECC) algorithm used on these devices with a 96% accuracy in discovery of secret keys. Identified vulnerabilities can then be used to implement relevant mitigation efforts.

[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

Comment: A selected chapter from the book Model Optimization Methods for Efficient and Edge AI: Federated Learning Architectures, Frameworks and Applications that discusses the various considerations for training ML models on edge devices.

[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

Comment: A book on Fog Computing, including chapters on various applications, challenges, information security, energy harvesting, energy efficiency, etc.

[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

Comment: This paper investigates the utilization of on-the-fly data compression for remote GPU-edge device integration in machine learning. They study various compression libraries and an adaptive compression algorithm that dynamically decides compression level based on compression ratio, data size, and network speed.

[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

Comment: This paper presents a framework for enhancing the efficiency of Distributed Intrusion Detection Systems (DIDS) with adaptive ML models and edge computing utilization. Key contributions include utilizing edge devices for computing to reduce latency and bandwidth usage as well as an adaptive machine learning model to reduce false alarms.

[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

Comment: A review of applications of ML in the Internet of Vehicles. Applications range from computation offloading decisions, network resource management, quality of experience for the user, and security enhancement. Future work looks at integrating 6G, using ML models to forecast issues with self-driving cars, and further experience enhancements.

[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

Comment: A selected chapter from Fundamentals of IoT and Wearable Technology Design, First Edition. Provides definition of Fog computing, cloud topologies and services, and several cloud platforms and considerations for choosing a platform for a specific application.

[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

Comment: A selected chapter from Fog, Edge, and Pervasive Computing in Intelligent IoT Driven Applications, First Edition. Gives definitions for fog, edge and pervasive computing. Provides an overview of machine learning frameworks designed for these applications. Defines ML techniques for edge computing such as Naïve Bayes, Support Vector Machines, K-nearest Neighbor and K-means.

[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

Comment: A selected chapter from Model Optimization Methods for Efficient and Edge AI: Federated Learning Architectures, Frameworks and Applications, First Edition. An explanation of ML model optimization techniques categorized by predeployment, deployment-time, and postdeployment. Each category is given ample attention.

[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.

Comment: A selected chapter from TinyML for Edge Intelligence in IoT and LPWAN Networks. This chapter present techniques for indoor localization of mobile devices, specifically two types of signal fingerprinting techniques. Indoor obstacles can impede signal strength which causes inaccuracy in conventional trilateration approaches, so locations can be fingerprinted based on the signal profile. They introduce a TinyML model to move computation of the location from edge servers to the device.

[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

Comment: A study comparing the accuracy of two ML models deployed on a consumer grade drone for the application of object detection and tracking. They employed YOLOv4 and Mask R-CNN on a Parrot Mambo drone. Transfer learning was done on a YOLOv4 model pretrained on the COCO dataset with a small subset of hand labeled images specific to the task. Automated labeling was done for the rest of the study's custom data set using this model.

[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

Comment: Integration of Deep Belief Networks (DBN) and Q-Learning to construct a mission decision-making model. A common battlefield scenario is a situation with a lot of unknowns. They seek to implement a ML model to allow for decision making adaptations based on real-time ground truth rather than mission pre-planning objectives. The specific tasks studied are reconnaissance and air-to-air confrontation.

[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

Comment: This paper proposes the Adaptive-Structure Extreme Learning Machine (ASELM) algorithm for optimizing inter-UAV communication in a dynamic environment. The algorithm accurately predicts future characteristics of the communication channel to minimize signal loss and retransmission, thus improving energy efficiency.

[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

Comment: A study on employing multiple UAVs to perform data collection on field sensors. A novel metric is introduced that measures Age of Information vs unit of energy consumption to optimize the flight path for each UAV. And a deep reinforcement learning algorithm is used to optimize collaboration amongst all UAVs.

[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

Comment: This paper showcases a graph transformer network approach to enhance virtual network (VNE) on a UAV swarm network. They are able to show significant improvements in embedding efficiency.

[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

Comment: This paper introduces a multi-module deep learning network for multi-target tracking in UAV fire control systems. The proposed architecture improves handling of challenges such as rapid movement and occlusion. They also achieve a better latency value versus YOLOv4-DeepSORT and Faster R-CNN.

[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

Comment: The paper proposes a reinforcement learning framework for maintaining connectivity among multiple UAVs during multi-target search and rescue missions. It highlights the importance of balancing search efficiency with connectivity. And demonstrates that this approach is scalable and adaptable to various mission scenarios.

[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

Comment: This paper presents attention mechanisms to enhance a multi-agent reinforcement learning (MARL) algorithm to optimize routing strategies for UAV swarms. Attention mechanisms allow UAVs to focus on pertinent information from neighboring agents leading to more efficient and effective communication. This also enhances scalability due to the reduction in communication overhead.

[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

Comment: This paper integrates charging considerations into the DRL to incorporate charging stops in route planning for UAVs. The DRL model allows the UAV to make dynamic decisions in real-time based on unforeseen events that require more energy usage, leading to path or charging schedule adjustments.

[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

Comment: This study introduces a two-phase methodology for SAR missions. In phase one the UAV navigates to a particular location in a building leveraging a DL model for obstacle detection and avoidance. Stage two the UAV employs a brute force searching algorithm to locate potential victims in the structure.

[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*

Comment: Introduction of the Multi-Environment Asynchronous Double Deep Q-Network (MEADDQN) algorithm, which integrates prioritized experience replay (PER) to improve data efficiency and training speed. Emphasizes role specialization in multi-UAV systems to enhance collaboration in a multi-objective mission.

[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.

Comment: Survey on mobile edge AI in vehicular networks

[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.

Comment: Survey on edge intelligence in wireless networks

[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.

Comment: Survey on On-Device AI models

[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.