Deep at the Edge

An investigation and application of deep learning on constrained IoT devices.

Kieran Brooks

University of Regina

200236702

CS 490CO

Dr. Maher Elshakankiri

Abstract

This study investigates the feasibility of deploying quantized deep learning models for object detection tasks on specialized Tensor Processing Unit (TPU) hardware in edge devices. A proof-of-concept worker safety compliance monitoring application was implemented using the EfficientDet-Lite family of models, and the impact of post-training quantization on model performance was evaluated. Transfer learning was employed to accelerate model iteration during training and development. The evaluation results revealed a decrease in performance across all evaluation metrics following quantization, with smaller models demonstrating superior throughput in the constrained environment of the development board. Despite this reduction in performance, quantization is essential for fully utilizing the capabilities of edge deployed TPU hardware. The study also highlights the importance of evaluating performance trade-offs, specific use case requirements, and long-term sustainability of toolchains and libraries when selecting a platform for deployment. Future research should focus on optimization strategies, novel model architectures, and robust toolchains to improve the performance and maintainability of computer vision applications in edge devices, particularly for safety-critical applications where more efficient models and higher resource availability are necessary.

Introduction

Edge computing is an emerging paradigm where data processing for IoT sensor data occurs at or near the data source instead of transmitting all data to a centralized cloud for analysis. This approach is gaining popularity due to the increasing volume of data generated by IoT devices and the need for real-time analysis and decision-making. Benefits of edge computing include reduced latency, enhanced privacy, and decreased network bandwidth usage.

Computer vision, a subset of artificial intelligence, enables machines to interpret and understand visual information from the world. Advancements in this field have made it possible to create models capable of recognizing objects, people, and even emotions from images and videos. These models have numerous practical applications, such as surveillance, object tracking, and facial recognition.

Industrial IoT applications involve the use of connected devices to monitor and manage industrial processes. One crucial aspect of industrial IoT is ensuring worker safety compliance. In industries such as manufacturing, construction, and mining, adhering to safety regulations is imperative to prevent accidents and injuries. By leveraging computer vision technology, systems can be developed to monitor workers and

identify instances of non-compliance with safety regulations, thereby preventing accidents and injuries, reducing liability, and ensuring regulatory compliance.

The study investigates the feasibility of using edge computing devices for implementing computer vision applications in industrial IoT settings, with a focus on the challenges of carrying out complex inference on edge IoT devices. A prototype industrial IoT system for identifying worker safety compliance using computer vision is implemented and deployed on a Google Coral development board, designed for running machine learning models at the edge. This implementation aims to demonstrate a possible use case for edge computing and computer vision and understand the challenges involved in rapid prototyping a computer vision device. The research will help to identify the potential benefits and limitations of using edge devices for computer vision applications in IIoT settings and will point to promising techniques and methods to accelerate development.

Of primary concern to developers of systems that would like to take advantage of deep learning in an industrial setting are the resource constraints placed on the devices that are deployed. Constraints such as processing power, memory, and energy consumption, significantly impact the performance of deep learning models running on edge devices and require that specific considerations be made for model optimization and conversion for the edge computing platform. Inference on edge devices presents several challenges due to these resource limitations. Academic research and commercial development in this area have uncovered several strategies for model development that an industrial application could benefit from during model training and deployment.

Related Work

In a comprehensive study on the anatomy of deep learning image classification and object detection in commercial edge devices, Kolosov et al. (2022) focused on a case study of face mask detection. They optimized, evaluated, and compared 7 image classification and 6 object detection state-of-the-art models on 5 commercial off-the-shelf edge devices, including Raspberry Pi 4, Intel Neural Compute Stick 2, NVIDIA

Jetson Nano, NVIDIA Jetson Xavier NX, and i.MX 8M Plus. They developed a full end-to-end video pipeline face mask detection architecture and optimized the 13 deep learning models for the edge devices using optimization frameworks such as TensorFlow Lite, OpenVINO, TensorRT, and eIQ. Different optimization options, like various levels of quantization, were also evaluated and compared. This study offers valuable insights into the performance trade-offs among inference time, energy consumption, efficiency (throughput/watt), and value (throughput/dollar) for different deep learning models and edge devices. It provides recommendations on which optimization frameworks, libraries, and options to use and how to select the right device depending on the target metric. This research complements the work of Winzig et al. (2022) on the Google Coral Dev Board Mini, as both studies explore the capabilities and limitations of edge devices for AI applications. The comprehensive benchmarks provided by Kolosov et al. (2022) for various edge devices and optimization frameworks can serve as a valuable reference for developers and researchers working with edge AI applications, including those using the Google Coral Dev Board.

Bhardwaj, Suda, and Marculescu explore the challenges and opportunities for widespread deployment of deep learning in the IoT era in their article "EdgeAI: A Vision for Deep Learning in the IoT Era". They emphasize that to achieve what they refer to as the EdgeAI paradigm, it is crucial to address three significant challenges: “the lack of big data at the edge, computation-aware deployment of learning models, and communication-aware distributed inference” (Bhardwaj et al., 2021). The authors highlight the need for new breakthroughs in artificial intelligence research that can effectively deploy learning at the edge, thereby exploiting the vast network of connected IoT devices. By tackling these challenges, they argue that rapid adoption of intelligence at the edge can be facilitated, providing a foundation for future research directions in the field. Though all the goals of their research are beyond the limited scope of this feasibility study, the underlying concepts related to constrained devices are applicable. To realize distributed inference at the edge, breakthroughs in research are needed that optimize the deployment and development methodologies involved as well as the devices themselves.

Development for an industrial IoT computer vision application poses challenges that are specific to the IoT domain. For example, data collection and labeling are critical steps in training machine learning models for

computer vision applications, but collecting data in industrial environments raises concerns related to privacy, data ownership, and security. Additionally in an industrial setting, latency of data transfer can significantly affect the performance of machine learning models, necessitating exploration of the challenges of network connectivity and latency in edge devices and strategies for optimizing network performance for machine learning inference. Security is a crucial concern in industrial IoT settings, leading to consideration of strategies for securing the system against potential attacks. IoT networks are typically constrained and can have significant effects on edge device performance and communication, prompting the exploration of strategies for overcoming network constraints to ensure optimal performance. Finally, dealing with small and fragile devices requires a robust design and error handling to maintain system reliability and fault tolerance, emphasizing the importance of these factors in ensuring the durability and dependability of edge IoT devices.

In recent years, deep learning models have been applied for weed detection in agricultural fields to help farmers control weeds more effectively, increase crop yields, and reduce pesticide usage. Umar Farooq et al. (2022) propose a lightweight deep learning model for weed detection on IoT devices, utilizing the YOLOv4-tiny architecture, which demonstrates impressive performance on edge devices and potential for application in other IoT applications as well. Earlier research, including studies by Lopez-Granados (2010), Romeo et al. (2013), Pulido-Rojas et al. (2016), Osorio et al. (2020), and Chen et al. (2021), have explored various aspects of weed detection using computer vision and deep learning techniques. Farooq et al.'s work contributes to this growing body of research by introducing a lightweight and cost-effective deep learning model for weed detection on IoT devices, expanding the potential for real-time, precision weed management in agricultural settings. Their work applies specifically to the computer vision model selection aspects of the feasibility study as it points to the performance benefits of single shot multi class detection algorithms as a basis for the choice of implementation model.

Ancilotto et al. investigate the role of smart vision sensors (SVSs) in deep learning-based object detection pipelines and aim to determine the optimal input space for IoT end-node applications. Their findings suggest that grayscale augmented with the motion bitmap yields the best performance, providing promising results for real-world applications with low power consumption. This research is relevant as it highlights the

importance of considering constraints on inputs and the use of SVSs when developing energy-efficient computer vision solutions for edge devices. Our study attempts to examine a modest first approach to feasibility for an industrial application of edge inference for visual data. Using techniques such as model quantization, transfer learning, and on-device model compilation, inference performance will be assessed. In summary, the work seeks to propose a feasible development proposition for computer vision on constrained edge devices and identify challenges and potential pitfalls.

Methodology

Dataset and Model Selection

The study began with data collection and preprocessing, where suitable data sources for worker safety compliance are identified. The dataset that has been chosen is the Syin "Hardhat and Safety Vest Image for Object Detection" dataset available on Kaggle (J. Syin, 2020). This dataset has been preprocessed and annotated in a standardized format for training and validation setup. Data gathering, annotating, and preprocessing is a time-consuming endeavour and has implications for the future commercial viability of an actual product. While the chosen dataset for this feasibility study is sufficient for rapid prototyping in a hypothetical application, it is crucial to acknowledge the challenges associated with data availability and annotation in real-world implementations of computer vision models. In a feasibility context, relying on a preprocessed and annotated dataset like the one used in this study has its limitations, as obtaining a sufficient volume of high-quality data in a production environment for widespread implementation can be challenging. Real-world applications will likely require the collection and annotation of large volumes of data to ensure accurate and reliable performance. One approach involves sourcing the annotation process to third-party platforms such as Amazon Mechanical Turk. This may reduce costs, though it introduces additional challenges related to quality control, privacy, confidentiality, and trade secrets. Using a public dataset for this study is convenient; however, a production implementation would require significant human and financial resources to generate quality annotation data. Organizations interested in deploying computer vision

solutions for worker safety compliance monitoring must be prepared to invest in data collection, annotation, and preprocessing efforts. This investment is essential for achieving the desired level of accuracy and reliability required of a computer vision application deployed in a safety critical environment. It is acknowledged that the challenges and limitations associated with data availability, annotation, and resource allocation in real-world implementations of computer vision models for industrial IoT settings are non-trivial. Recognizing these challenges will help guide future research and development efforts in this field, ensuring that the proposed solutions are not only feasible but also scalable and sustainable.

Model selection for this feasibility study was carried out with a transfer learning model training strategy in mind. Transfer learning is an approach in which a pre-trained model is fine-tuned for a new task or domain by leveraging the knowledge it gained from a previous task. This technique enables rapid prototyping and serves as a suitable analog for production implementation, as it reduces the time and resources needed to develop a custom model from scratch while still achieving significant results. For this feasibility study, two models were selected: the efficientdet-lite0 and efficientdet-lite4. Both models have been pretrained on the COCO dataset, a widely used benchmark dataset for object detection tasks. Efficientdet-lite0 is the least complex variant of the efficientdet-lite model family. EfficientDet, (short for 'efficient detector') demonstrated superior performance when presented in the originating paper by Tan et al. (2020). This model architecture represents the most efficient modern object detection model available with sufficient accuracy and performance characteristics to form a baseline understanding of what can be achieved on the constrained operating environment of the Google Coral dev board. Efficientdet-lite0 has a model size of 5.7MB, an input resolution of 320x320, and the lowest benchmarked inference times of the efficientdet family at 29.26ms (coral.ai, 2021). On the other hand, efficientdet-lite4 is the most complex variant of the efficientdet-lite family available in the TensorFlow model library, with a model size of 19.88MB an input resolution of 640x640 and a benchmark inference latency of 721.78ms (coral.ai, 2021)

By benchmarking both the simplest and the most complex models in the efficientdet-lite family, this study aims to observe the differences in performance across the range of model complexity points available and provide insights into the feasibility and performance capabilities of on-device inference for a computer

vision task. The rationale for choosing these two extremes is to understand the trade-offs between efficiency, accuracy, and performance when deploying computer vision models on edge devices, while taking into account hardware limitations and performance requirements in a side-by-side comparison.

Transfer Learning

Transfer learning, a concept that has gained popularity since the mid-2000s, is the process of applying knowledge learned from one or more source tasks to a different target task, often leading to faster and better solutions (S. J. Pan et al., 2010). This technique has become particularly important in the context of object detection, where labeled data can be expensive or time-consuming to obtain. By employing transfer learning, practitioners can more rapidly complete training and move into deployment development phases.

One key advantage of transfer learning is the open sourcing of pre-trained models that have been trained on highly generalized datasets. For example, the efficientdet-lite family of models accessed through the TensorFlow library are pretrained on the COCO dataset (T. Lin et al., 2014). This dataset is designed for high generalization in object detection and contains 90 different objects.

In transfer learning, the upper layers of the pretrained model are "unfrozen," allowing the weights and biases of the neural network architecture in these layers to be subject to training. This approach leads to faster convergence when training for specific tasks while retaining the general abstract representation from the source task (S. J. Pan et al., 2010). The underlying theoretical assumption behind transfer learning is that similar tasks share abstract representations that are present deeper in the layers of the model (S. J. Pan et al., 2010). For instance, a model trained to detect bananas, cars, and boats would have some abstract generalizations in common with a model trained to detect people, helmets, and heads.

It is important to note that transfer learning is not a magic bullet, some target tasks may not benefit from this approach, however generally speaking, in hypothetical production training scenarios, transfer learning is the preferred approach as its results have been shown to be superior to naive models trained on the same data, in terms of training times (S. J. Pan et al., 2010). This makes transfer learning an indispensable tool in the

development and deployment of effective and efficient computer vision models across various applications and industries.

In this study, model training was conducted using the TensorFlow Lite Python library, a high-level library specifically designed for streamlining dataset formation and training tasks for TensorFlow models. This library facilitated the efficient training of the two models under investigation: efficientdet-lite0 and efficientdet-lite4. Both the data and the models were imported into the Google Colab notebook web IDE, which provides an interactive environment for executing and Python code on cloud-based compute resources. The models were then configured to enable training only for the top layers, in line with the transfer learning strategy. By doing so, the models could leverage the pre-trained knowledge from the COCO dataset while fine-tuning the model for the specific target task of worker safety compliance monitoring.

Training was carried out for 50 epochs on each model, using GPU hardware. To ensure a comprehensive evaluation of the models' performance, the training data was divided into three subsets: training, test, and validation. An 80%, 10%, 10% split was used, with the majority of the data utilized for training the models, while the test and validation sets were reserved for assessing the models' performance and generalization capabilities following each epoch and after training, respectively.

Model Evaluation

During the evaluation of trained models, unseen test samples extracted from the same dataset but not used during the training process are employed. The evaluation process involves exposing the model to each test example and obtaining a set of predictions, which are then compared to the known ground truth for the respective example. Subsequently, an array of performance metrics are calculated and analyzed, both before and after model quantization.

Precision refers to the proportion of true positive predictions (correctly identified objects) out of all the positive predictions made by the model, measuring the accuracy of the model's positive predictions. A higher precision indicates that the model's positive predictions are more reliable. Conversely, recall is the proportion

of true positive predictions out of all the actual positive instances (objects) present in the dataset. Recall essentially quantifies the model's ability to identify all relevant objects within the given data, with a higher recall implying that the model is more effective at detecting all objects of interest. Models are further evaluated for precision and recall across three object sizes and Intersection over Union (IoU) thresholds. IoU is a widely-used metric for evaluating the performance of object detection and segmentation algorithms. It refers to the degree of overlap between predicted bounding boxes and ground truth, and is calculated by dividing the area of intersection of the predicted bounding box by the area of the union of the bounding boxes for prediction and ground truth, respectively. Average Precision (AP), measures the model's object detection capabilities by averaging precision values across all recall levels. A higher AP value signifies superior performance overall. Additionally, AP values at different IoU thresholds, specifically at 50% (AP50) and 75% (AP75), are considered to offer further insight into the model's performance under varying degrees of strictness. Class-specific AP values, such as 'AP_helmet', 'AP_person', and 'AP_head', are calculated for each object class to evaluate the model's performance in detecting specific objects. Higher values denote better detection capabilities for the corresponding objects. Furthermore, the evaluation encompasses the average precision for detecting objects of different sizes, including large (AP_lg), medium-sized (AP_med), and small objects (AP_sm). Average Recall (AR) is another important metric employed to measure the model's ability to correctly identify objects of various sizes. AR values are calculated for large (AR_lg), medium-sized (AR_med), and small objects (AR_sm).

In this study, post-training quantization will be employed to prepare the object detection models for deployment on the target TPU platform. Post-training quantization involves converting the model parameters into a lower bit-depth integer representation after the model has been fully trained. This conversion process affects accuracy but also ensures compatibility with TPU operations which are limited to integer-only arithmetic. Once the models have been quantized and deployed, they will be evaluated on the device in real time, specifically focusing on inference latency. By measuring the time required for the models to generate predictions on the target platform, this study aims to provide a comprehensive assessment of the performance improvements offered by model quantization and TPU deployment.

Model Quantization

Model quantization is a technique employed to compress deep learning models by reducing the numerical precision of the model's parameters, such as weights and biases, from floating-point representation to a lower bit-depth integer representation. This compression method effectively enhances the deployment of models on hardware with limited resources without significantly compromising their performance. Integer-only arithmetic exhibits several advantages over floating-point arithmetic, particularly in terms of hardware efficiency (B. Jacob et al., 2018) These advantages include a reduced memory footprint, faster computations, energy efficiency, and hardware support, as numerous platforms, such as the TPU chip on the Google Coral development board, are specifically engineered to optimize integer-only arithmetic for quantized models. The TPU chip is an application-specific integrated circuit (ASIC). It accelerates learning workloads for neural network models by facilitation of efficient integer-only matrix multiplication by means of a systolic array implemented in hardware to compute matrix partial sums (Sato and Young, 2017). Quantizing deep learning models for deployment on TPUs can significantly enhance their performance in aspects such as latency, energy efficiency, and computational cost, proving especially advantageous for edge devices and mobile platforms where computational resources and energy consumption present critical limitations.

Implementation

Implementation was carried out by means of a proof-of-concept demo application capable of performing inference on live video (K. Brooks, 2023). The demo application was built using the Flask web framework. It initializes the application, specifies a model file and label file, and sets parameters for object detection. The main webpage is rendered, and a generator function captures camera frames and passes them to the model for inference. A continuous video stream is returned and passed to the flask webserver, with the generator function called using camera and model instances as arguments. A separate camera class is responsible for capturing video frames from the webcam by extracting them and resizes them using the OpenCV library. An inference class initializes the TPU model and handles object detection by performing

inference on the resized frames and appending the detected objects to the image and returns the frame to flask for serving.

By employing this structure, the demo application captures live video, performs real-time inference using the quantized model on the TPU platform, and displays the results with bounding boxes and labels. This proof of concept validates the feasibility of deploying a post-training quantized model to the device and the suitability of the TPU platform for object detection tasks.

Results

	B	A	
	efficientdet-lite0		**Δ**
AP	0.230	0.215	1.50%
AP_helmet	0.378	0.364	1.37%
AP_person	0.006	0.004	0.19%
AP_head	0.305	0.276	2.93%
AP50	0.427	0.400	2.70%
AP75	0.231	0.214	1.65%
AP_sm	0.158	0.145	1.39%
AP_med	0.346	0.324	2.17%
AP_lg	0.211	0.204	0.73%
AR_sm	0.288	0.197	9.07%
AR_med	0.507	0.409	9.79%
AR_lg	0.424	0.290	13.38%
Latency			**69ms**
	efficientdet-lite4		**%Δ**
AP	0.347	0.331	1.68%
AP_helmet	0.523	0.505	1.78%
AP_person	0.017	0.013	0.41%
AP_head	0.502	0.473	2.84%
AP50	0.566	0.542	2.40%
AP75	0.392	0.375	1.76%
AP_sm	0.297	0.280	1.64%
AP_med	0.423	0.406	1.68%
AP_lg	0.339	0.307	3.12%
AR_sm	0.491	0.426	6.58%
AR_med	0.608	0.530	7.78%
AR_lg	0.580	0.434	14.69%
Latency			**902ms**

The results presented compare the performance of a model before (Column B) and after (Column A) the quantization process, using various evaluation metrics. The Δ column shows the difference between the pre- and post-quantized models. All metrics show an expected reduction due to the reduction in precision from float64 to int8. AP metrics showed a modest decrease compared to AR metrics generally. In all AR metrics, the post-quantized model demonstrates a significant decrease in performance compared to the pre-quantized model in all size categories: 9.07% for small objects, 9.79% for medium objects, and 13.38% for large objects. Significant increases in AP and AR scoring across all categories is observed between efficientdet-lite0 and efficientdet-lite4 with a mean increase of 0.13 generally observed between models. Most strikingly in the results is the difference in latency of model inference, with efficientdet-lite4 frame inference latency costing 13-fold the time cost as the smaller model. Overall, these results indicate that the quantization process has led to a decrease in the model's performance across all evaluation metrics, though the true impact on performance is the complexity of the model and it is expressed in terms of the inference latency and total model throughput.

Discussion

In this study, we investigated the feasibility of edge deployed computer vision models by implementing a proof-of-concept worker safety compliance monitoring application. The performance of quantized deep learning models for object detection tasks on a specialized TPU hardware platform was evaluated. The most critical findings were the differences in model inference throughput, with simpler models exhibiting superior performance in the constrained environment of the development board. Inference throughput is a significant limiting factor in the applicability of computer vision at the edge. Thus, specific use case testing is necessary for each application to ensure that the hardware under consideration can meet the requirements.

Quantization did affect performance; however, it is essential for executing tensor operations on the TPU hardware. While quantization resulted in a slight decrease in performance, CPU-bound operations would have been significantly slower (Google, 2023). As such, quantization is a necessary trade-off to take full advantage of the specialized TPU hardware.

Considering the feasibility of a worker safety compliance proof of concept, it could be achievable if adapted to the limited throughput and precision of smaller models or if applications could be specifically optimized to operate on lower precision data. However, for safety-critical applications where lives may be at stake, higher-performing and more efficient models, or edge computing platforms with higher resource availability should be considered if available. It is essential to evaluate the performance trade-offs and the specific use case requirements when selecting a platform for deployment.

Transfer learning proved to be an efficient method for accelerating model iteration during training and development of both prototype and production applications. This approach allows for faster adaptation of pre-trained models to new tasks, reducing the need for extensive training on large datasets. A key limitation and barrier to entry for firms aiming to implement such solutions is the availability and quality of training data. Bespoke datasets are often required for each application, and data annotation remains a time-consuming and resource-intensive process.

Future research should focus on developing models that take full advantage of the TPU hardware capabilities. This may involve exploring novel model architectures or training techniques specifically tailored for edge device deployment.

Another crucial aspect to consider is the toolchain and the limitations of high-level libraries. The complexity of working with highly abstracted libraries and potential vendor lock-in with proprietary toolchains can pose challenges if support for popular libraries or dependencies ceases, resulting in unmaintainable production code. It is essential to carefully evaluate the long-term sustainability of toolchains and libraries when selecting a platform for deployment.

In conclusion, this study highlights the potential of deploying quantized deep learning models on specialized TPU hardware for object detection tasks at the edge. Despite the trade-offs in performance, quantization is essential for fully utilizing the capabilities of TPU hardware. Future research should continue to explore optimization strategies, model architectures, and robust toolchains to further improve the performance and maintainability of computer vision applications in edge devices, particularly for safety-critical applications where more efficient models and higher resource availability are necessary.

References

D. Kolosov, V. Kelefouras, P. Kourtessis, and I. Mporas, "Anatomy of Deep Learning Image Classification and Object Detection on Commercial Edge Devices: A Case Study on Face Mask Detection," in IEEE Access, vol. 10, pp. 109167-109186, 2022, doi: 10.1109/ACCESS.2022.3214214.

J. Winzig, J. C. A. Almanza, M. G. Mendoza, and T. Schumann, "Edge AI - Use Case on Google Coral Dev Board Mini," in 2022 IET International Conference on Engineering Technologies and Applications (IET-ICETA), Changhua, Taiwan, 2022, pp. 1-2, doi: 10.1109/IET-ICETA56553.2022.9971614.

K. Bhardwaj, N. Suda, and R. Marculescu, "EdgeAI: A Vision for Deep Learning in the IoT Era," in IEEE Design & Test, vol. 38, no. 4, pp. 37-43, Aug. 2021, doi: 10.1109/MDAT.2019.2952350.

U. Farooq, A. Rehman, T. Khanam, A. Amtullah, M. A. Bou-Rabee, and M. Tariq, "Lightweight Deep Learning Model for Weed Detection for IoT Devices," in 2022 2nd International Conference on Emerging Frontiers in Electrical and Electronic Technologies (ICEFEET), Patna, India, 2022, pp. 1-5, doi: 10.1109/ICEFEET51821.2022.9847812.

F. Lopez-Granados, "Weed detection for site-specific weed management: mapping and real-time approaches," Weed Research, vol. 51, no. 1, pp. 1–11, 2011.

J. Romeo, J. M. Guerrero, M. Montalvo, L. Emmi, M. Guijarro, P. Gonzalez-de Santos, and G. Pajares, "Camera sensor arrangement for crop/weed detection accuracy in agronomic images," Sensors, vol. 13, no. 4, pp. 4348–4366, 2013.

K. Osorio, A. Puerto, C. Pedraza, D. Jamaica, and L. Rodríguez, "A deep learning approach for weed detection in lettuce crops using multispectral images," AgriEngineering, vol. 2, no. 3, pp. 471–488, 2020.

Z. Wu, Y. Chen, B. Zhao, X. Kang, and Y. Ding, "Review of weed detection methods based on computer vision," Sensors, vol. 21, no. 11, p. 3647, 2021.

A. Ancilotto, F. Paissan, and E. Farella, "On the Role of Smart Vision Sensors in Energy-Efficient Computer Vision at the Edge," in 2022 IEEE International Conference on Pervasive Computing and Communications Workshops and other Affiliated Events (PerCom Workshops), Pisa, Italy, 2022, pp. 497-502, doi: 10.1109/PerComWorkshops53856.2022.9767380.

J. Syin, "Hardhat and Safety Vest Image for Object Detection," Kaggle Dataset, 2020. [Online]. Available: https://www.kaggle.com/datasets/johnsyin97/hardhat-and-safety-vest-image-for-object-detection

M. Tan, R. Pang, and Q. V. Le, "EfficientDet: Scalable and Efficient Object Detection," in Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2020. [Online]. Available: https://doi.org/10.48550/arXiv.1911.09070

Coral.ai, "Object Detection Models," Coral.ai, 2021. [Online]. Available: https://coral.ai/models/object-detection/

S. J. Pan and Q. Yang, "A Survey on Transfer Learning," in IEEE Transactions on Knowledge and Data Engineering, vol. 22, no. 10, pp. 1345-1359, Oct. 2010, doi: 10.1109/TKDE.2009.191.

T. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft COCO: Common Objects in Context," in Proceedings of the European Conference on Computer Vision (ECCV), 2014, pp. 740-755.

B. Jacob, S. Kligys, B. Chen, M. Zhu, M. Tang, A. Howard, H. Adam, and D. Kalenichenko, "Quantization and Training of Neural Networks for Efficient Integer-Arithmetic-Only Inference," in Proceedings of the IEEE

K. Sato and C. Young, "An in-depth look at Google's first Tensor Processing Unit (TPU)," Google Cloud Blog, May 12, 2017. [Online]. Available: https://cloud.google.com/blog/products/ai-machine-learning/an-in-depth-look-at-googles-first-tensor-processing-unit-tpu

Conference on Computer Vision and Pattern Recognition (CVPR), pp. 2704-2713, 2018. Available: http://openaccess.thecvf.com/content_cvpr_2018/html/Jacob_Quantization_and_Training_CVPR_2018_paper.html

K. Brooks, "Safety Vision," GitHub Repository, 2023. Available: https://github.com/kierankyllo/safety-vision

OpenCV Development Team, "OpenCV (Open Source Computer Vision Library)," GitHub Repository. [Online]. Available: https://github.com/opencv/opencv-python

Google, "Edge TPU model benchmarks," Coral. [Online]. Available: https://coral.ai/docs/edgetpu/benchmarks/. [Accessed: 23-Apr-2023].

TensorFlow, "EfficientDet-Lite0 Object Detection Model (EfficientNet-Lite0 Backbone with BiFPN Feature Extractor, Shared Box Predictor and Focal Loss), Trained on COCO 2017 Dataset, Optimized for TFLite, Designed for Performance on Mobile CPU, GPU, and EdgeTPU," TensorFlow Hub, 2023. [Online]. Available: https://www.tensorflow.org/hub/models/efficientdet/lite0/detection. [Accessed: April 22, 2023].

TensorFlow, "EfficientDet-Lite4 Object Detection Model (EfficientNet-Lite4 Backbone with BiFPN Feature Extractor, Shared Box Predictor and Focal Loss), Trained on COCO 2017 Dataset, Optimized for TFLite, Designed for Performance on Mobile CPU, GPU, and EdgeTPU," TensorFlow Hub, 2023. [Online]. Available: https://www.tensorflow.org/hub/models/efficientdet/lite4/detection. [Accessed: April 22, 2023].