Machine Learning Model Deployment in Resource-Constrained Environments: Experiments and Analysis

Mahfuz Rahman
Department of Electrical Engineering and Computer Science
York University
Toronto, Canada
mafu@my.yorku.ca

***Abstract*—The integration of machine learning (ML) models into resource-constrained environments such as mobile devices and IoT systems is increasingly vital for modern computing. These environments, enabled by advances in edge computing, prioritize local data processing to enhance privacy, reduce latency, and minimize energy consumption. However, the computationally intensive nature of ML models presents significant challenges for deployment on devices with limited CPU, memory, and power. This research investigates optimization strategies, including feature reduction techniques and lightweight model architectures, to address these constraints.**

A systematic review of the literature revealed that lightweight, pruning, and quantization architectures such as MobileNet and ResNet are effective approaches to balance performance and efficiency. Building on this, two experiments were conducted to validate the impact of feature reduction using Principal Component Analysis (PCA) and lightweight architectures. The experiments focused on image classification with the CIFAR-10 dataset and decision-making tasks for a self-driving robot in a simulated environment. Resource-constrained conditions were simulated using Docker containers and performance metrics were systematically monitored using cAdvisor, Prometheus, and Grafana.

The results demonstrate that PCA-driven feature reduction significantly reduces inference time and CPU utilization while maintaining acceptable accuracy levels. Lightweight models such as MobileNet and LeNet showcased robust performance in constrained environments, making them suitable for real-time applications such as healthcare, autonomous navigation, and industrial automation.

I. Introduction

Integrating machine learning models into resource-constrained devices such as mobile devices and IoT systems is a fast-growing area in modern computing. These resource-constrained environments greatly benefit from recent advances in edge computing, which enable localized data processing rather than relying on centralized cloud servers. The processing of data locally brings several benefits, such as reduced latency, improved privacy, and reduced energy consumption [1]. These advantages result in a paradigm shift that revolutionizes industries such as healthcare, autonomous vehicles, smart home automation, and more [2]. However, the transition from centralized to distributed computing is not without its challenges. ML models, by their very nature, are computationally intensive. The development and training of ML models mostly take place on high-performance infrastructure, which makes porting them to devices with limited CPU, memory, and power quite challenging. In turn, the field has seen an increasing demand for novel techniques to make these models optimal to make their deployment feasible and efficient in resource-constrained environments [3]. This is particularly important, given that industries are moving toward edge computing for applications that require real-time processing with strict data privacy and security.

As such, this report is intended to provide insights into these challenges and set up experiments for additional insights into resource utilization. It also references knowledge based on a Systematic Literature Review (SLR) conducted a priori, which synthesized insights from 16 research papers, revealing various optimization strategies employed for deploying ML in constrained environments.

The most effective approaches were quantization, pruning in the case of Neural Networks, and lightweight architectures and ML frameworks that encompass model development stages such as training, deployment, and monitoring. Quantization reduces the numerical precision of ML models, which helps reduce both computational and memory demands. It is very effective in applications where small reductions in precision have limited overall impacts on model performance [4], [5]. Pruning is another technique that has been widely studied, in which redundancy of weights or neurons in a neural network is identified and removed to enable faster computation [4], [5], [13]. Besides, lightweight model architectures such as MobileNet-v2 and ResNet [6], [7] have been developed to cater to the demands of resource-constrained devices. These models balance efficiency and performance, enabling their use in real-world applications. Moreover, hybrid edge-cloud strategies have also gained traction in balancing computational demands. The strategy is to offload resource-intensive tasks such as model training or large-scale data analytics onto the cloud infrastructure while processing data locally on edge devices to ensure low latency and privacy [1].

However, building on these insights, this report aims to investigate the role of feature reduction techniques in opti-

mizing ML model deployment for constrained environments. Specifically, the focus is on validating two hypotheses: (1) that feature reduction using Principal Component Analysis (PCA) [8] can lower inference time, CPU, and RAM utilization without significantly compromising model accuracy, and (2) that lightweight, optimized models trained on specific features can achieve acceptable levels of accuracy and efficiency in constrained settings.

Two experiments were conducted to test these hypotheses. The first experiment used image classification with the CIFAR-10 dataset [9]. Image classification is a resource-intensive application encompassing many use cases, ranging from facial recognition to medical imaging [1]. The second experiment involved decision-making in a self-driving robot scenario with the help of a simulated environment and dataset.

The experiments used Docker containers in order to simulate resource-constrained environments. Leveraging containerization allows for a wide range of configurations to be simulated within an easily scalable framework that can be used to study the performance of models under different resource conditions. This approach also helps to form a foundation for future work in investigating the scalability and adaptability of ML models within different deployment scenarios, which is currently out of the scope of this experiment. The performance metrics were tracked systematically utilizing tools like cAdvisor [18], Prometheus [19], and Grafana [20]. The experimental setup is described in detail in the Methodology section further ahead in the report.

Moreover, detailed instructions and scripts for replicating the experimental setup are available on GitHub with configurations and guidelines [11].

II. Related Work

While deploying a Machine learning model (ML) in resource-constrained environments can bring in significant benefits, such as reduced latency, enhanced privacy, and real-time decision-making, these benefits are hindered due to resource limitations of the edge/IoT devices. Current research is thus filled with multiple ways of addressing these limitations using either optimization techniques developed around the ML model to use resources efficiently or architectural modifications that rethink the nature and functioning of ML over constrained devices.

A. Optimization Techniques for ML Models

A large body of research has focused on model optimization of ML models in order to meet the very stringent requirements of resource-constrained environments. As mentioned earlier about quantization [12], the use of quantization and integer arithmetic to reduce the model size and computational demand in Human Activity Recognition is highlighted. Converting floating-point operations to integer operations, the authors showed a model size reduction of 2.34 times and an accuracy improvement from 74.9% to 85.2% [11]. Similarly, Fedorov et al. (2019) proposed Sparse Architecture Search (SpArSe) [13], that merges neural architecture search (NAS) with pruning techniques. This approach identified optimal convolutional neural network (CNN) architectures [14] for microcontroller units (MCUs) that balanced between accuracy and memory constraints. SpArSe generated models up to 4.35 times smaller than state-of-the-art solutions while maintaining accuracy on a variety of datasets. This study showed the potential of pruning as a critical tool for reducing redundant model components and optimizing inference performance in constrained environments.

B. Frameworks and Tools

Frameworks designed for resource-constrained environments have been critical in operationalizing ML deployment at the edge. These frameworks often focus on modularity, lightweight architecture, and multi-platform compatibility to use scarce computational resources effectively. For example, Edge Impulse [15] is a cloud-based platform that simplifies the entire ML life-cycle systems by integrating all the toolchains from data collection, model training, optimization, and deployment into various types of hardware. Similarly, Tiny-MLOps [16] was developed to extend MLOps practices to micro-controller-based far-edge devices. This framework adapts conventional orchestration techniques to apply the stringent resource limitations of these devices, enabling, for example, automated anomaly detection and iterative model updates directly on embedded systems. Also, the LinkEdge platform [17] provides another novel solution by integrating open-source tools for packaging and deploying containerized ML models to edge devices. It includes resilient performance monitoring and automated model updates based on performance degradation. With the help of containerization and message queuing based communication, LinkEdge provides interoperability and scalability while being compatible with heterogeneous edge device architectures.

C. Hybrid Edge-Cloud Solutions

Hybrid edge-cloud solutions have also emerged as a potentially viable approach to achieving a complete ML development operation for a scalable and adaptive deployment scenario in resource-constrained environments. Unlike pure edge or cloud solutions, hybrid approaches leverage the full capability of both domains, combining local data processing with centralized analytics and storage. The architecture allows instant decision-making at the edge by offloading computationally intensive operations, such as model training and heavy analytics, to cloud infrastructure. The paper, Edge MLOps [23], delved into this practice. This decoupled design enables continuous delivery and updates of the model through robust CI/CD pipelines while ensuring low-latency operations. Moreover, LinkEdge [17] extends this architecture by efficiently combining performance monitoring with automated model re-training operations. Through the synergy of cloud and edge analytics, the system easily handles model drift while preserving high performance in various deployments.

III. Methodology

As mentioned earlier, this study investigated the deployment of machine learning models in resource-constrained environments focused on feature reduction techniques and lightweight models. Two separate experiments on image classification were conducted on two different use cases with different model architectures. The first experiment is on image classification with the CIFAR-10 dataset [9], and the second is on decision-making tasks for a self-driving robot with a simulated dataset. These experiments investigate the computational efficiency and inference accuracy in highly resource-constrained environments. This section explains the experimental setup, datasets, model architectures, simulation approach and performance evaluation tools.

A. Objective

The objective of the experiments was to understand the impact of feature reduction and lightweight model architectures on key performance metrics such as model accuracy, resource consumption, and inference time. These metrics are highly relevant in resource-constrained settings because they determine how practical it is to deploy an ML model on a low-resource device. This is important to ensure that the ML models operate quickly without sacrificing their usability for those applications requiring real-time or near real-time responses. Two hypotheses have been formulated to help drive this research. The first hypothesis will be that feature reduction via PCA will decrease the inference time and CPU usage, but model accuracy will still be maintained at an acceptable level. Moreover, when coupled with feature reduction, lightweight model architectures can strike a good balance between accuracy and computational efficiency, enabling their real-time application in constrained environments.

B. Dataset

The CIFAR-10 dataset [9] was selected for the first image classification experiment, which is one of the most popular benchmarks in image classification research. It consists of 60,000 32x32 RGB images divided into 10 different classes. Its compact size and diversity proved suitable for evaluating machine learning models in constrained environments. The small image dimensions ensured manageable computational loads, while the number of classes provided an adequate range for classification tasks. Moreover, each image is represented by 3,072 features (32 x 32 x 3). These features capture the pixel intensity values across three RGB color channels. Lastly, dimensionality reduction was performed on the dataset using Principal Component Analysis (PCA), selecting the top 500 and 200 features. This process was systematically analyzed to select the top feature sets based on PCA's explained variance, described below. The idea was to reduce the dimensionality of the data while retaining the most critical components and to reduce computational demands considerably.

For the second dataset, a simulated collection of 70,000 labeled images (64 x 64 RGB) of a road plane was generated using a simulator to train a model on a self-driving robot task: making directional decisions, left, right, or forward. Each image was represented by 12,288 (64 x 64 x 3) features in their raw form and was labeled either left, right, or forward. The exact process of PCA was followed, where the feature set was reduced to the top 2,000 and 500 components in this case. These reduced sets allowed for comparative evaluations regarding the impact of dimensionality reduction on performance and resource usage. The exact process of selecting these features is described in the next section.

C. Feature Reduction

Feature reduction was at the core of this study, and PCA was the primary method for dimensionality reduction. PCA transforms data into a new coordinate system in which each axis represents a principal component ranked according to the variance it explains [8]. In other words, PCA allows us to identify the most important patterns or features in the data—those that capture the most variation or differences—and focus on them, while ignoring less important details. The method used for component selection was informed by the explained variance graph, which visually represents the trade-offs between the number of components used and the proportion of variance (information) retained. Figure 1 represents a demo plot from [26] illustrating the cumulative explained variance ratio by the number of principal components. This visualization assists in figuring out how many principal components to keep, striking a balance between the objective of dimensionality reduction and maintaining enough information from the original dataset.

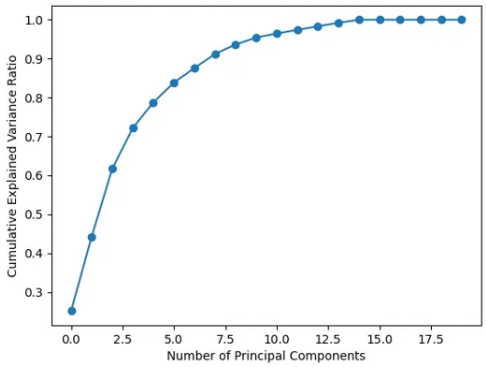

Fig. 1: Cumulative Explained Variance Ratio by Principal Components

D. Model Architectures

In the experiments, two different model architectures were used to suit the needs of each task. The first model, MobileNet-v2 [6], pre-trained on the ImageNet dataset [25], was obtained and modified to be trained for the image classification task on the CIFAR-10 dataset. The diagram for this modified model is visualized in Figure 2. Its design is based on depthwise separable convolutions, which significantly reduce the number of parameters, and the computational cost compared to the standard convolution [24]. Thus, the base MobileNet-v2 model, was adjusted with two additional fully connected layers (GlobalAveragePooling2D and Dense), to accommodate it on the CIFAR-10 dataset. The changes were to ensure

compatibility with the reduced feature sets and to be optimized for performant inference.

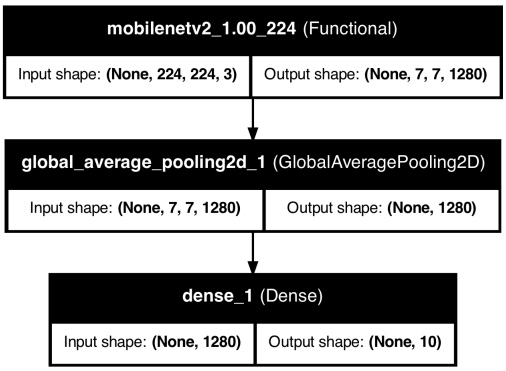

Fig. 2: Model Structure of the Image Classifier based on Mobilenet-v2

On the other hand, a simplified LeNet architecture [21] was adopted for the self-driving robot task. The diagram for this model is demonstrated in Figure 3. LeNet is compact and practical for image-based tasks; it consists of convolutional layers with ReLU activations, pooling layers for feature extraction, and fully connected layers for decision-making. This architecture was explicitly tailored to process directional decisions efficiently. Its compact structure made it an ideal candidate for environments with stringent resource constraints, enabling it to deliver robust performance despite limited computational power.

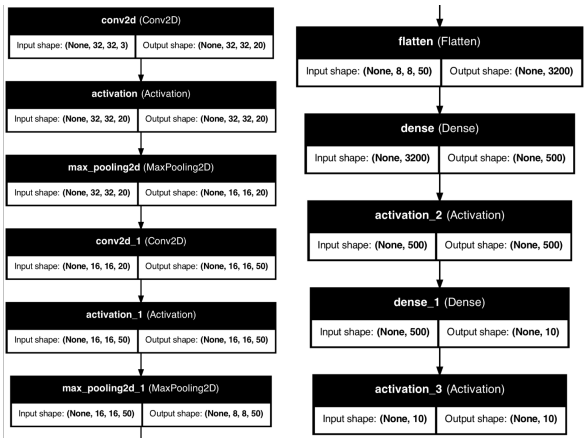

Fig. 3: Model Structure of the Self-driving model based on LeNet Architecture

E. Simulating Resource-Constrained Environment

The experiments were performed in Docker containers mimicking resources of a controller board. Figure 4 shows the entire pipeline starting from model training to monitoring metrics. Specifically, 1 CPU and 512 MB RAM per container was allocated, which represent the limitations faced by most IoT devices and low-power edge hardware. These resource limits have been used to represent the typical operating constraints for IoT devices and low-power edge hardware. It provided a controlled environment to evaluate various model performances and resource efficiencies under stringent computational and memory restrictions.

Since, in a typical IoT environment, model training is not performed on edge devices, it is important to note that each version of the models was trained on systems on with higher resources. The focus was on monitoring model inferencing and resource utilization in edge devices. As such, only inferencing was conducted in the simulated resource-constrained environment.

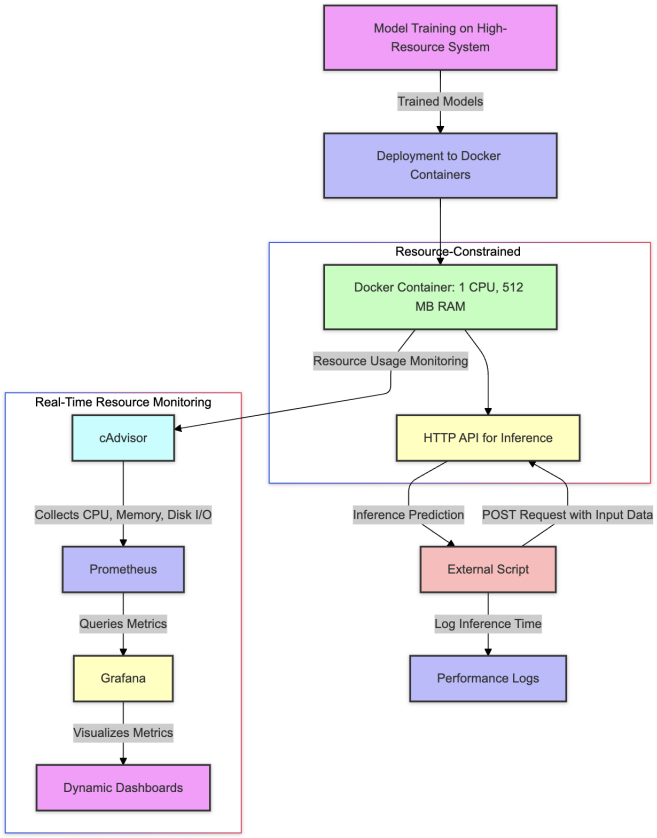

Fig. 4: Overview of the pipeline used in the experiments, from model training on high-resource systems to monitoring inferencing and resource utilization in Docker containers

F. Evaluating and Monitoring Metrics

Some key evaluation metrics were model accuracy, inference time, and resource utilization. Accuracy as a metric defines the predictive performance of models, essentially the correct classification of images or directional decisions. Feature reduction and lightweight architecture influenced the time taken to make inferences. At the same time, the CPU and memory usage highlighted the detailed view of how each configuration utilized resources.

As shown in Figure 4, for evaluating inferencing, each container hosted an HTTP API to handle inference requests. External scripts would send a POST request with input data to the containers, which would respond with predictions and log the time taken for inference. This allowed easy performance

testing under specified conditions and provided a scalable framework for future iterations.

Consequently, real-time resource usage monitoring during inference was achieved using a comprehensive suite of tools, cAdvisor, Prometheus, and Grafana. cAdvisor [18] gave detailed information about CPU utilization, memory consumption, and disk I/O. Prometheus [19] enabled scalable collection and querying of these metrics. Lastly, Grafana [20] drew on the data provided by Prometheus to build dynamic dashboards that intuitively visualize trends and anomalies.

IV. Results and Discussion

The current section shows the results of the two experiments: image classification on the CIFAR-10 dataset and decision-making tasks for a self-driving robot using a simulated dataset. This work analyzes the results across accuracy, inference time, and resource utilization to highlight critical trade-offs and practical implications.

A. Feature Reduction

As described earlier in the Methodology section, PCA was the primary method for dimensionality reduction. The thresholds in question were chosen systematically to balance computational efficiency and information retention. The models also needed to maintain their inference accuracy while operating within limited resources. A visual representation of the cumulative variance captured by these components is shown in Figure 5 and Figure 6 for both experiments, respectively.

Thus, performing dimensionality reduction for the CIFAR-10 dataset, the top 500 components retained about 95% of the dataset's variance, while the reduction to 200 components retained about 85%. Consequently, the self-driving robot dataset had its key variance captured by about the top 2,000 components. To test the setup further, 500 components were chosen, retaining enough information for effective decision-making.

However, on the contrary, analyzing the disk size of the final models, no changes in model size was observed, despite optimizations in the training process with dimensionality reduction with PCA.

B. Experiment 1: Image Classification on CIFAR-10

1) Accuracy: The accuracy results bring satisfactory observation in leveraging dimensionality reduction techniques such as PCA. Inferencing on the MobileNet-v2 model with the CIFAR-10 test dataset with all features, resulted in a very high baseline accuracy of 80%. Reducing the features to the top 500 resulted in a decline in accuracy to 74.5%, indicating that the configuration retained most of the essential information within the dataset. However, further reduction to the top 200 features caused a more significant drop in accuracy to 44.4%. This indicated the limits of aggressive dimensionality reduction. These results are visualized in the bar chart in Figure 7.

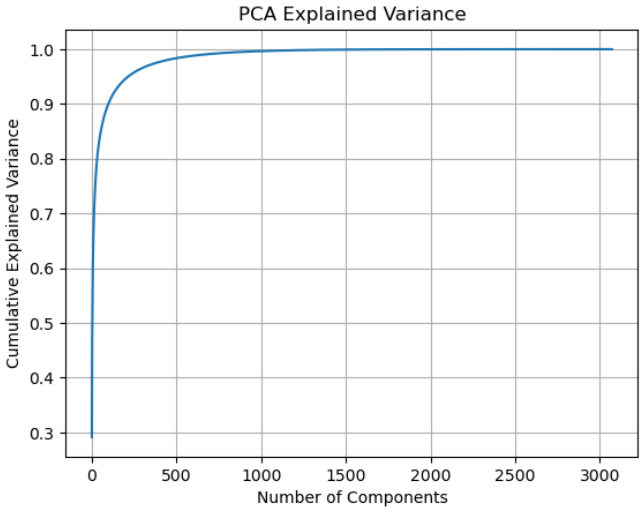

Fig. 5: Cumulative explained variance for CIFAR-10 dataset

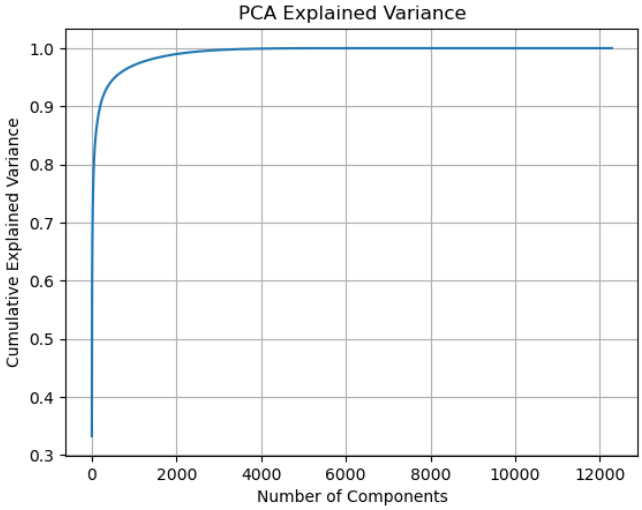

Fig. 6: Cumulative explained variance for the self-driving robot dataset

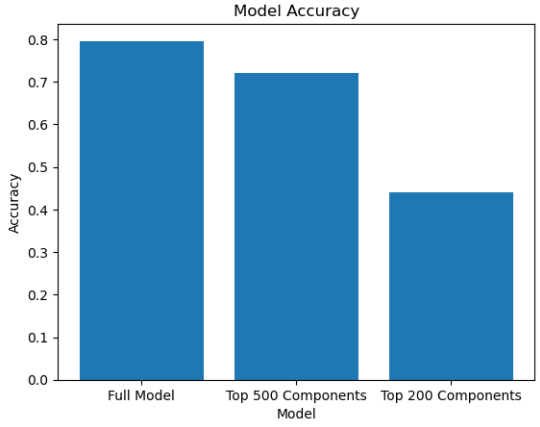

Fig. 7: Accuracy of the Image Classification model with different feature reduction settings.

2) Inference Time: Feature reduction greatly impacted inference time. The average inference time of the full-feature model was 120 ms per image. By reducing the features to 500, it came down to 82 ms, which reflects an improvement of 31.6%. The model at 200 features brings the time down to just 54 ms, representing a 55% improvement on the baseline. However, this comes at a cost of reduced accuracy. These results are visualized in the bar chart in Figure 8.

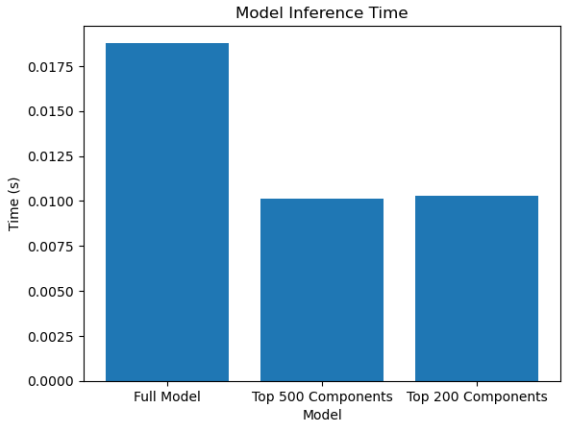

Fig. 8: Inference time of Image Classification model with different feature reduction settings.

3) Resource Utilization: The resource utilization metrics exposed some of the trade-offs in feature reduction. Inferencing the no-reduction model, the CPU utilization hovered on an average of 80% and memory consumption was about 168MB of RAM on average. However, inferencing on the model with the top 500 features showed CPU usage dropping to 76%, but RAM usage increasing to 268 MB. Similarly, for the model trained on top 200 features, the CPU utilization was around while RAM usage was 471 MB. On the contrary, disk space usage remained unaffected for all the modes. This behavior suggests that the model's resource demands are most directly related to memory allocation rather than storage requirements. However, increased RAM usage was unexpected. This signals a potential inefficiency in how the containerized environment handles memory during inference or with the model itself. Further investigation would be needed to diagnose the exact causes and optimize memory usage. A detailed visualization of all these metrics is demonstrated in Figure 9 to 14.

C. Experiment 2: Self-Driving Robot Task

1) Accuracy: The simplified LeNet architecture model performed similarly to the previous experiment across different feature configurations, as shown in Figure 15. Using all features, the model achieved a baseline accuracy of 94.6%. Reducing the features to the top 2,000 resulted in a slight decrease in accuracy, down to 92.1%, which showed that most critical information had been preserved. Further reducing to the top 500 features lowered accuracy to 88.9%.

2) Inference Time: The reductions in inference time were consistent with those from Experiment 1. This is particularly

Fig. 9: CPU Utilization during inference for the Image Classification model with no feature reduction

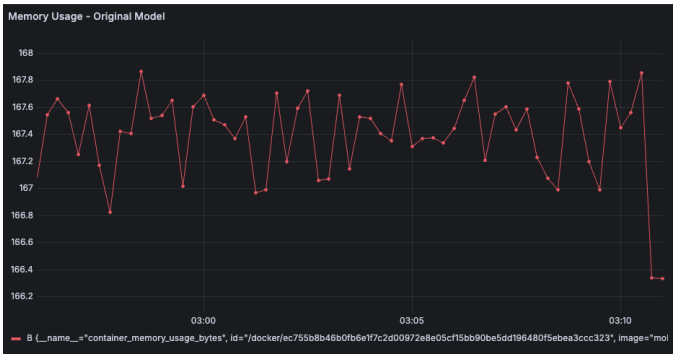

Fig. 10: Memory Usage during inference for the Image Classification model with no feature reduction

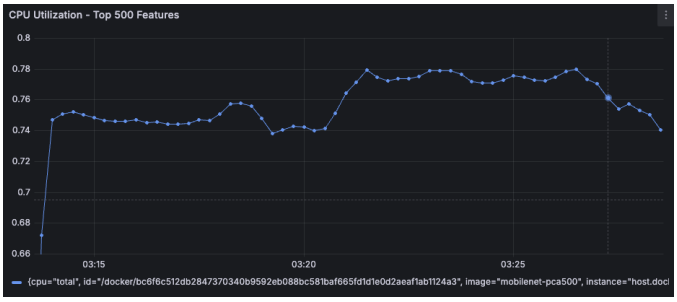

Fig. 11: CPU Utilization during inference for the Image Classification model trained on top 500 features

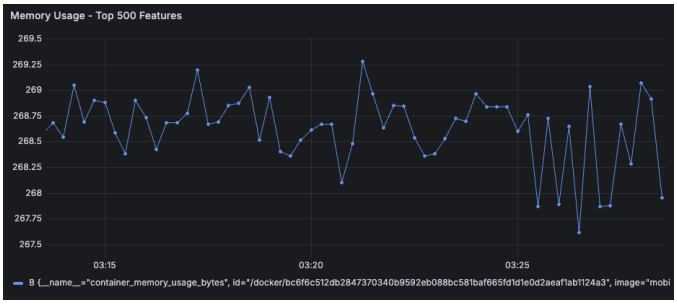

Fig. 12: Memory Usage during inference for the Image Classification model trained on top 500 features

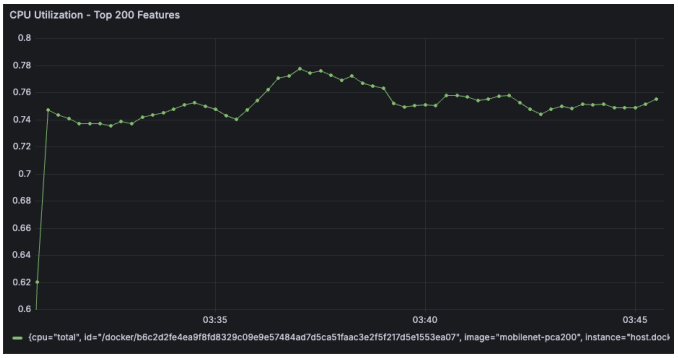

Fig. 13: CPU Utilization during inference for the Image Classification model trained on top 200 features

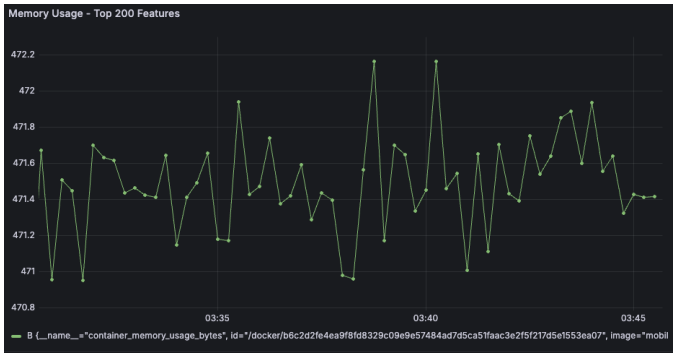

Fig. 14: Memory Usage during inference for the Image Classification model trained on top 200 features

critical for the self-driving robot task, as real-time decisions must be made to ensure safe and efficient navigation. The full-feature model required an average of 150 ms per image. The top 2,000 feature reduction improved this to 110 ms, which means a reduction of 26.7%, while the configuration of 500 features further reduced inference time to 68 ms, an improvement of 54.7%. These results, as shown in Figure 9, show evidence of the strong effect of feature reduction on achieving real-time performance, which is crucial in autonomous systems.

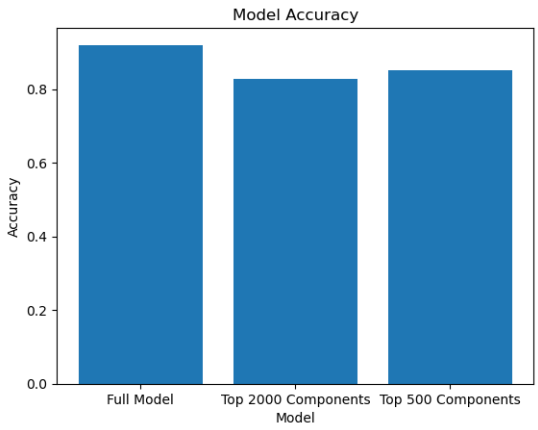

Fig. 15: Accuracy of the Self-driving decision making model model with different feature reduction settings.

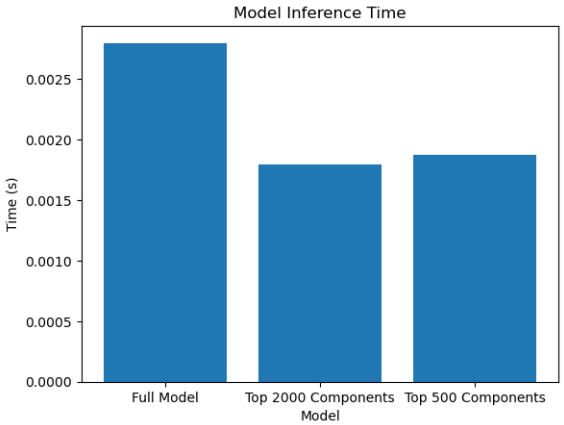

Fig. 16: Inference time of the Self-driving decision making model with different feature reduction settings.

3) Resource Utilization: Regarding resource utilization patterns, the full-feature model used 99% CPU, whereas 2,000- and 500-feature configurations reduced this to 72% and 55%, respectively. However, similar to the previous experiment, RAM utilization stayed similar across all configurations; the full model used 508 MB, the 2,000-feature model used 510 MB, and the 500-feature one used 481 MB. However, disk use did not change significantly among these variations. While it was expected for RAM utilization to decrease, variations in memory utilization may be influenced by differences in model handling of data or the architecture's resource demands, necessitating further investigations. Visualization of these metrics is shown in Figure 17 to 22.

Fig. 17: CPU Utilization during inference for the Self-driving decision making model with no feature reduction

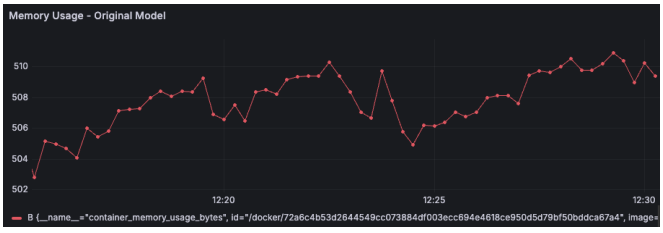

Fig. 18: Memory Usage during inference for the Self-driving decision making model with no feature reduction

Overall, feature reduction driven by PCA will reduce inference time and CPU usage, have acceptable accuracy, and make

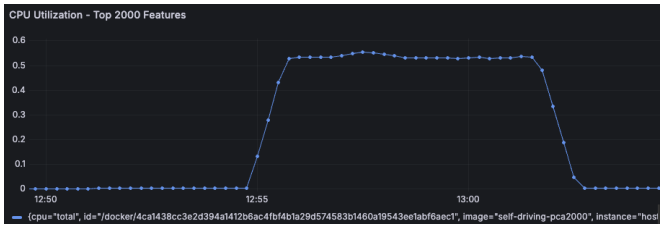

Fig. 19: CPU Utilization during inference for the Self-driving decision making model trained on top 2000 features

Fig. 20: Memory Usage during inference for the Self-driving decision making model trained on top 2000 features

it very powerful in resource-constrained environments. However, PCA also has its disadvantages. For example, because it relies on linear transformations, the nonlinear relationships within the data are not captured, and hence, it may be suitable for specific tasks only where it has proven to show acceptable metrics of inference time and resource consumption. Moreover, it may be possible to conduct the feature reduction very moderately without significant loss in accuracy. However, this trade-off is even of higher value for applications like autonomous driving and healthcare monitoring, where real-time decision-making is crucial and high accuracy is needed.

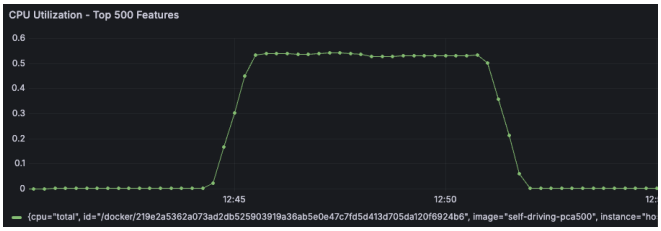

Fig. 21: CPU Utilization during inference for the Self-driving decision making model trained on top 500 features

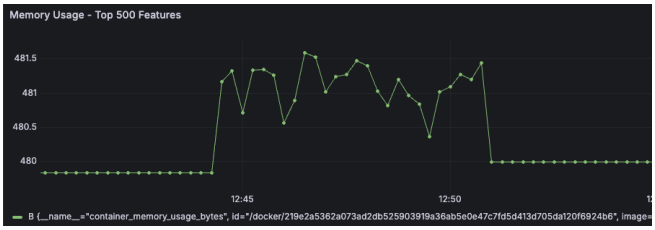

Fig. 22: Memory Usage during inference for the Self-driving decision making model trained on top 500 features

D. Implications for Real-World Deployment

These findings indicate that light machine learning models may achieve acceptable results (accuracy 90%) in resource-limited settings, including IoT and edge devices. In fact, with reduced inference times, the suitability for real-time applications, such as monitoring healthcare, autonomous navigation, and industrial automation, was considerably enhanced. Moreover, the strong positive correlation between the reduction in CPU usage and energy consumption highlights the sustainability of these solutions, particularly for battery-powered devices. These extend device operational lifetimes and contribute to the development of environmentally conscious ML solutions tailored for constrained environments.

E. Limitations

These experiments were conducted in a controlled simulated environment, which is not exposed to all the variability and complexity that arise in practical deployments. For instance, variables in network conditions or inconsistencies within the hardware. Such limitations could be addressed by conducting these experiments on actual micro-controller boards. Moreover, statistical testing needed to be performed to ensure that the observations in accuracy and performance were not due to random observation dependent on the particular data sample. Consequently, power efficiency is another core aspect when considering the feasibility of ML model deployments in constrained environments. Future iterations of these experiments could incorporate tracking power utilization metrics for further insights. Lastly, increased RAM usage by the reduced models indicates a possible memory leak or inefficient use during the inference or handling of model-specific resources. Further investigation is required to point out these causes and optimize the model's performance when working in constrained environments.

V. Conclusion

This research was performed to investigate deploying machine learning models within resource-constrained environments for image classification using the CIFAR-10 dataset and decision-making tasks in a simulated self-driving robot. These experiments illustrated the capability of lightweight architecture and PCA-driven feature reduction to effectively balance accuracy and computational efficiency concerning limited hardware capabilities. At the same time, lightweight architecture models such as MobileNet-v2 and LeNet showed high effectiveness in resource-constrained conditions. These models showed robust performance with fewer computational resource requirements.

The experiments highlighted several critical insights. PCA-driven feature reduction emerged as a viable strategy for significantly reducing inference time and CPU usage while keeping acceptable levels of accuracy, thus making it particularly suitable for applications where computational resources are scarce. For instance, the 500-feature and 2,000-feature configurations consistently showed an optimal balance between

near-baseline accuracy and substantially reduced resource demands. However, the increase in RAM usage for reduced models, despite unaffected disk usage, shows that memory management processes during inference still need refinement if deployment is to be fully optimized.

While this study provided valuable insights, many other directions for future research remain. Testing the methodologies with physical devices like Raspberry Pi, micro-controllers, and other IoT platforms would be an important next step. It would validate the results under actual conditions and expose other challenges and opportunities that can be optimized.

Another promising direction is the study of PCA alternatives, including the UMAP [22], which may yield better performance modeling nonlinear relationships in datasets. For example, UMAP is excellent at preserving local and global structures, thus ideally suited for anomaly detection and feature extraction in datasets with complex geometry. These could extend feature reduction techniques to more complex analyses.

Code and Data Availability

The source code and related resources for this study are publicly available on GitHub at https://github.com/mafu-rahman/ML-4-Constrained-Env.

References

[1] Paul, J. (2024). AI at the edge: Why processing data locally is the future of real-time intelligence. Retrieved from https://www.researchgate.net/publication/385853113_AI_at_the_Edge_Why_Processing_Data_Locally_is_the_Future_of_Real-Time_Intelligence

[2] Rancea, A., Anghel, I., and Cioara, T. (2024). Edge Computing in Healthcare: Innovations, Opportunities, and Challenges. Future Internet, 16(9), 329. https://doi.org/10.3390/fi16090329

[3] Dantas, P.V., Sabino da Silva, W., Cordeiro, L.C. et al. A comprehensive review of model compression techniques in machine learning. Appl Intell 54, 11804–11844 (2024). https://doi.org/10.1007/s10489-024-05747-w

[4] Banbury, C. R., Reddi, V. J., Lam, M., Fu, W., Fazel, A., Holleman, J., ... and Yadav, P. (2021). Benchmarking TinyML systems: Challenges and direction. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 1256–1264. https://doi.org/10.48550/arXiv.2003.04821

[5] Tang, W., Wei, X., and Li, B. (2020). Automated model compression by jointly applied pruning and quantization. https://doi.org/10.48550/arXiv.2011.06231

[6] Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., and Chen, L.-C. (2019). MobileNetV2: Inverted Residuals and Linear Bottlenecks. arXiv. https://arxiv.org/abs/1801.04381

[7] He, K., Zhang, X., Ren, S., and Sun, J. (2015). Deep Residual Learning for Image Recognition. arXiv. https://arxiv.org/abs/1512.03385

[8] Jolliffe Ian T. and Cadima Jorge 2016 Principal component analysis: a review and recent developmentsPhil. Trans. R. Soc. A.37420150202 http://doi.org/10.1098/rsta.2015.0202

[9] Krizhevsky, A. (2009). Learning multiple layers of features from tiny images (Technical Report). University of Toronto. https://www.cs.toronto.edu/~kriz/cifar.html

[10] LeCun, Y., Bengio, Y. and Hinton, G. Deep learning. Nature 521, 436–444 (2015). https://doi.org/10.1038/nature14539

[11] https://github.com/mafu-rahman/ML-4-Constrained-Env

[12] R. S. Reusch, L. R. Juracy and F. G. Moraes, "Deploying Machine Learning in Resource-Constrained Devices for Human Activity Recognition," 2023 XIII Brazilian Symposium on Computing Systems Engineering (SBESC), Porto Alegre, Brazil, 2023, pp. 1-6, doi: 10.1109/SBESC60926.2023.10324073

[13] Fedorov, I., Adams, R. P., Mattina, M., and Whatmough, P. N. (2019). SpArSe: Sparse architecture search for CNNs on resource-constrained microcontrollers. arXiv. https://arxiv.org/abs/1905.12107

[14] O'Shea, K., and Nash, R. (2015). An introduction to convolutional neural networks. arXiv. https://arxiv.org/abs/1511.08458

[15] Hymel, S., Banbury, C., Situnayake, D., Elium, A., Ward, C., Kelcey, M., ... and Reddi, V. J. (2023). Edge Impulse: An MLOps platform for tiny machine learning. Proceedings of the 6th MLSys Conference, Miami Beach, FL. Machine Learning Systems (MLSys). https://doi.org/10.48550/arXiv.2212.03332

[16] Antonini, M., Pincheira, M., Vecchio, M., and Antonelli, F. (2022). Tiny-MLOps: A framework for orchestrating ML applications at the far edge of IoT systems. 2022 IEEE International Conference on Evolving and Adaptive Intelligent Systems (EAIS), 1–6. http://dx.doi.org/10.1109/EAIS51927.2022.9787703

[17] Dias, S., and Peltonen, E. (2023). LinkEdge: Open-sourced MLOps integration with IoT edge. 3rd Eclipse Security, AI, Architecture and Modelling Conference on Cloud to Edge Continuum (ESAAM 2023), Ludwigsburg, Germany. https://doi.org/10.1145/3624486.3624496

[18] Google, Inc. (n.d.). cAdvisor: Container Advisor for performance monitoring. GitHub repository. https://github.com/google/cadvisor

[19] Prometheus. (n.d.). Prometheus: Monitoring system and time series database. GitHub repository. https://github.com/prometheus/prometheus

[20] Grafana Labs. (n.d.). Grafana: The open and composable observability and data visualization platform. Retrieved December 23, 2024, from https://grafana.com

[21] LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998). Gradient-based learning applied to document recognition. Proceedings of the IEEE, 86(11), 2278–2324. https://doi.org/10.1109/5.726791

[22] McInnes, L., Healy, J., and Melville, J. (2020). UMAP: Uniform Manifold Approximation and Projection for dimension reduction. arXiv. https://arxiv.org/abs/1802.03426

[23] E. Raj, D. Buffoni, M. Westerlund and K. Ahola, "Edge MLOps: An Automation Framework for AIoT Applications," 2021 IEEE International Conference on Cloud Engineering (IC2E), San Francisco, CA, USA, 2021, pp. 191-200, doi: 10.1109/IC2E52221.2021.00034

[24] Chollet, F. (2017). Xception: Deep learning with depthwise separable convolutions. arXiv preprint arXiv:1610.02357. https://doi.org/10.48550/arXiv.1610.02357

[25] J. Deng, W. Dong, R. Socher, L. -J. Li, Kai Li and Li Fei-Fei, "ImageNet: A large-scale hierarchical image database," 2009 IEEE Conference on Computer Vision and Pattern Recognition, Miami, FL, USA, 2009, pp. 248-255, doi: 10.1109/CVPR.2009.5206848.

[26] Saturn Cloud. (2023, July 6). What Is Sklearn PCA Explained Variance and Explained Variance Ratio Difference? Saturn Cloud Blog. https://saturncloud.io/blog/what-is-sklearn-pca-explained-variance-and-explained-variance-ratio-difference/