https://doi.org/10.31891/csit-2024-4-2
UDC 004.9

Tymur ISAIEV, Tetiana KYSIL
Khmelnytskyi National University

METHOD FOR IMPROVING THE PERFORMANCE OF CONVOLUTIONAL NEURAL NETWORKS USING AN ACCELERATOR

The effectiveness of convolutional neural networks (CNNs) has been demonstrated across various fields, including computer vision, natural language processing, medical imaging, and autonomous systems. However, achieving high performance in CNNs is not only a matter of model design but also of optimizing the training and inference processes. Using accelerators like the Google Coral TPU provides significant improvements in both computational efficiency and overall model performance. This paper focuses on the integration of the Coral TPU to enhance CNN performance by speeding up computations, reducing latency, and enabling real-time deployment.

Training deep learning models, particularly CNNs, is computationally intensive. Traditional CPUs or GPUs can take hours or even days to train large networks on complex data. The accelerator offloads these intensive tasks, allowing the host machine to focus on other operations and making training more efficient. This enables researchers to experiment with multiple architectures and hyperparameters within shorter cycles, thereby improving the model's accuracy and robustness.

CNNs are widely deployed in edge computing scenarios where real-time predictions are critical, such as in robotics, autonomous vehicles, and smart surveillance systems.Unlike traditional cloud-based solutions, where models are executed remotely and suffer from network delays, the Coral TPU ensures low-latency predictions directly on the device, making it ideal for time-sensitive applications.

Another key advantage of using accelerators like Coral TPU is the ability to efficiently handle optimized and lightweight models. These optimized models are well-suited for the Coral TPU's architecture, allowing developers to deploy high-performing networks even on resource-constrained devices. The TPU's ability to handle quantized models with minimal loss in accuracy further enhances the CNN's practical usability across various domains.

The Coral TPU is designed to minimize power consumption, making it an ideal solution for battery-powered or energy-constrained devices. This energy efficiency ensures that CNNs can run continuously on devices like drones, IoT sensors, or mobile platforms without exhausting their power supply. Additionally, the scalability of the TPU makes it easy to deploy multiple accelerators in parallel, further improving throughput for applications that require processing high volumes of data, such as real-time video analysis.

The Coral TPU also facilitates on-device learning, where models can be incrementally updated based on new data without requiring a full retraining session. This feature is particularly useful in dynamic environments, such as autonomous vehicles or security systems, where the model needs to adapt quickly to new conditions. With the TPU handling the computational workload, CNNs can be fine-tuned on the device, ensuring they remain accurate and responsive over time.

Keywords: GPU acceleration, TPU optimization, mixed precision training, parallel computing, model parallelism, data parallelism, batch normalization.

Тимур ІСАЄВ, Тетяна КИСІЛЬ
Хмельницький національний університет

МЕТОД ПОКРАЩЕННЯ РЕЗУЛЬТАТІВ РОБОТИ ЗГОРТКОВИХ НЕЙРОННИХ МЕРЕЖ З ДОПОМОГОЮ ПРИСКОРЮВАЧА

Ефективність згорткових нейронних мереж (CNN) була продемонстрована в різних галузях, включаючи комп'ютерний зір, обробку природної мови, медичну візуалізацію та автономні системи. Однак досягнення високої продуктивності ШНМ - це не лише питання дизайну моделі, але й оптимізації процесів навчання та висновків. Використання прискорювачів, таких як Google Coral TPU, забезпечує значне покращення як обчислювальної ефективності, так і загальної продуктивності моделі. Ця стаття присвячена інтеграції Coral TPU для підвищення продуктивності CNN за рахунок прискорення обчислень, зменшення затримок і розгортання в реальному часі.

Навчання моделей глибокого навчання, зокрема CNN, вимагає значних обчислень. Традиційні CPU або GPU можуть витрачати години або навіть дні на навчання великих мереж на складних даних. Прискорювач розвантажує ці інтенсивні завдання, дозволяючи хост-машині зосередитися на інших операціях і роблячи навчання більш ефективним. Це дозволяє дослідникам експериментувати з різними архітектурами та гіперпараметрами за коротші цикли, тим самим підвищуючи точність та надійність моделі.

CNN широко використовуються в сценаріях периферійних обчислень, де прогнозування в реальному часі є критично важливим, наприклад, в робототехніці, автономних транспортних засобах та інтелектуальних системах спостереження. На відміну від традиційних хмарних рішень, де моделі виконуються віддалено і страждають від мережевих затримок, Coral TPU забезпечує прогнозування з низькою затримкою безпосередньо на пристрої, що робить його ідеальним для додатків, чутливих до часу.

Ще однією ключовою перевагою використання таких прискорювачів, як Coral TPU, є можливість ефективно працювати з оптимізованими та полегшеними моделями. Ці оптимізовані моделі добре підходять для архітектури Coral TPU, дозволяючи розробникам розгортати високопродуктивні мережі навіть на пристроях з обмеженими ресурсами. Здатність TPU обробляти квантовані моделі з мінімальною втратою точності ще більше підвищує практичну придатність CNN у різних галузях.

Coral TPU розроблений для мінімізації енергоспоживання, що робить його ідеальним рішенням для пристроїв, що живляться від батареї, або пристроїв з обмеженим енергоспоживанням. Така енергоефективність гарантує, що CNN можуть безперервно працювати на таких пристроях, як дрони, датчики Інтернету речей або мобільні платформи, не виснажуючи їхні джерела живлення. Крім того, масштабованість TPU дозволяє легко розгортати кілька прискорювачів паралельно, що ще *більше підвищує пропускну здатність додатків, які потребують обробки великих обсягів даних, таких як аналіз відео в реальному часі.*

Coral TPU також полегшує навчання на пристрої, де моделі можуть поступово оновлюватися на основі нових даних без необхідності повного перенавчання. Ця функція особливо корисна в динамічних середовищах, таких як автономні транспортні засоби або системи безпеки, де модель повинна швидко адаптуватися до нових умов. Завдяки тому, що TPU справляється з обчислювальним навантаженням, ШНМ можна точно налаштувати на пристрої, гарантуючи, що вони залишатимуться точними і швидко реагуватимуть з плином часу.

Ключові слова: Прискорення GPU, оптимізація TPU, навчання змішаної точності, паралельні обчислення, паралелізм моделей, паралелізм даних, пакетна нормалізація.

Introduction

In recent years, the need to improve the performance of convolutional neural networks (CNNs) has grown rapidly due to their increasing deployment in various real-world applications. These applications span diverse fields, including healthcare, finance, autonomous systems, and smart cities, reflecting the versatility and effectiveness of CNNs in tackling complex problems. This demand is fueled by the availability of specialized accelerators, like the Google Coral TPU [1-6] (Tensor Processing Unit), which enable users to optimize training and inference processes without requiring extensive knowledge of hardware-level optimizations. The introduction of such accelerators has revolutionized the approach to deploying machine learning models, making it possible for developers and researchers to focus more on model design and less on the intricacies of hardware. Leveraging these accelerators has become essential in achieving high computational efficiency and ensuring fast, reliable performance across a diverse range of tasks, from image classification to real-time object detection.

The goal of this work is to develop a method for enhancing CNN performance by integrating the Coral TPU accelerator and designing a system that optimizes both the training and inference processes. This involves a comprehensive approach that not only seeks to improve the computational capabilities of CNNs but also enhances their operational efficiency in practical applications.

To achieve this goal, the following tasks must be completed:

1. Optimize the CNN architecture. This entails fine-tuning the design of the CNN to take full advantage of the Coral TPU's architecture and capabilities. Techniques such as pruning, layer fusion, and specialized activation functions can be employed to ensure the model aligns with the TPU's strengths.
2. Test and validate the performance of the CNN model using the TPU. Conduct rigorous benchmarking to evaluate the model's performance on the Coral TPU compared to traditional CPUs and GPUs. This step is critical to ascertain the actual benefits gained from using the TPU in terms of speed and accuracy.
3. Implement a scalable system. Design and deploy a system that supports real-time inference and efficient model updates. This includes creating a user-friendly interface that allows developers to interact with the model, monitor its performance, and update it as needed with new data.

Accelerators like the Coral TPU play a fundamental role in CNN optimization, as the speed and accuracy of a model depend heavily on the hardware used for both training and inference. The TPU is specifically designed for machine learning workloads, allowing CNNs to run on resource-constrained devices while maintaining high performance. This capability makes them ideal for edge applications, where traditional computing resources might not be available or feasible [7-13].

In fields like computer vision, CNN models are often deployed for tasks such as object detection, image classification, and medical diagnosis. These tasks require high-speed inference with minimal latency, particularly in real-time systems where decisions must be made swiftly based on incoming data. The Coral TPU significantly reduces training time by performing computationally intensive operations, such as convolutions, more efficiently than standard CPUs or GPUs. This capability allows developers to iterate quickly on model architectures and hyperparameters, ensuring the final model is accurate, robust, and well-suited for the intended application.

The Coral TPU is optimized for edge computing, where models must provide predictions in real-time without relying on cloud infrastructure. In robotics, autonomous vehicles, and smart surveillance systems, low-latency predictions are crucial for effective functioning. For example, in autonomous vehicles, the ability to make real-time decisions based on sensor inputs can determine safety and operational success. With TPU-accelerated inference, these systems can operate independently, eliminating delays associated with cloud-based solutions and enabling real-time decision-making directly on the device. This independence not only improves performance but also enhances privacy and security by keeping sensitive data on-device rather than transmitting it to external servers.

Key Components of the Optimization Process:

1. Model Architecture Optimization. Adjusting the CNN architecture to leverage the TPU’s capabilities includes optimizing layer arrangements and employing techniques such as depthwise separable convolutions to enhance performance. This also encompasses support for quantization to reduce model size while maintaining accuracy, allowing for efficient use of memory and computational resources.
2. Training Acceleration. Offloading matrix operations to the TPU significantly reduces training time, allowing developers to test multiple configurations more efficiently. This expedited training process is crucial, especially in research environments where rapid prototyping and experimentation are essential for innovation.
3. On-device Inference. Deploying models on devices equipped with Coral TPU for real-time predictions ensures fast and reliable performance without external dependencies. This capability is especially

important for applications like augmented reality, where user experience relies heavily on immediate feedback from the system.

The TPU supports quantized models, which reduce memory usage and computation costs, making them suitable for deployment on Internet of Things (IoT) devices, drones, and mobile platforms [10-15]. This ensures that high-performance CNNs can run continuously on devices with limited power and computational resources. The ability to maintain performance while minimizing resource consumption is a significant advantage in environments where battery life and processing power are constrained. Additionally, the TPU allows for parallel deployment of multiple accelerators, further improving the system's throughput for tasks like real-time video analysis, where multiple streams of data must be processed simultaneously.

The Coral TPU also enables incremental model updates, allowing CNNs to adapt to new data without requiring complete retraining. This feature is particularly useful in dynamic environments, such as autonomous vehicles or security systems, where models need to quickly adjust to new conditions or threats. For instance, in a security application, a model may need to learn to identify new types of suspicious behavior based on recent surveillance footage. With the TPU handling the computational workload, on-device fine-tuning ensures that models remain accurate and responsive over time, leading to better performance and reliability in practical applications.

Overall, the integration of the Coral TPU into the development and deployment of CNNs marks a significant advancement in the field of machine learning. By optimizing model performance through targeted enhancements, real-time processing capabilities, and the ability to operate effectively in resource-constrained environments, the Coral TPU facilitates a new era of intelligent systems capable of tackling the challenges of modern applications.

Analysis of Existing Solutions

The increasing integration of specialized accelerators in machine learning workflows has transformed the way convolutional neural networks (CNNs) are optimized for real-world tasks. This transformation is particularly evident in the realm of artificial intelligence, where the rise of sophisticated algorithms has significantly advanced capabilities in image recognition and classification. The deployment of CNNs has become more efficient and accessible on edge devices, thanks in large part to the advent of accelerators like the Google Coral TPU (Tensor Processing Unit). These accelerators are designed specifically for high-performance machine learning tasks, enabling models to be deployed in environments that demand both speed and efficiency.

This advancement has driven the exploration of TPU-accelerated models across various applications, including robotics, autonomous vehicles, and security systems. In these fields, the ability to process data in real-time is crucial for making timely decisions and ensuring operational safety. For instance, in robotics, quick processing of sensor data allows for immediate adjustments to movement, enhancing overall efficiency and effectiveness. However, deploying CNNs on hardware accelerators involves unique challenges that need to be addressed for optimal performance:

1. Computational Demands. Large models require significant computational power, and traditional CPUs or GPUs may struggle to handle complex networks efficiently within acceptable time frames. This inefficiency can lead to delays in processing, impacting the overall performance of applications reliant on quick responses. Accelerators such as the Coral TPU are specifically engineered to address this challenge by significantly speeding up operations like convolutions and matrix multiplications. This capability enables developers to deploy more complex models without the typical constraints associated with conventional hardware.
2. Performance Variability. The performance of CNNs can vary across different real-world conditions, such as variations in lighting, angles, or environmental noise. For example, an image captured in bright sunlight may yield different results compared to one taken in low light. This variability makes it essential for TPU-accelerated models to efficiently handle diverse input data, ensuring robustness during real-time execution. To achieve this, models must be trained on a wide variety of scenarios and data to prepare for real-world applications.
3. Challenges with Unseen Data. Inference on unseen data remains a significant challenge in the field of machine learning. Even high-performing CNNs may struggle when exposed to new inputs that were not part of the training data. This emphasizes the need for ongoing updates and incremental learning on the device. Implementing techniques such as continual learning allows models to adapt to new data inputs, thus improving accuracy and performance over time.
4. Need for Human Oversight. Certain industries may still require human validation for sensitive applications, such as surveillance or healthcare. While TPU accelerators can enhance CNN performance and improve efficiency, human oversight remains critical in ensuring accuracy and ethical deployment. For instance, in medical applications, a CNN may identify potential health issues, but a trained healthcare professional must validate the results before any diagnosis or treatment is administered.

Several projects have demonstrated the potential of integrating Coral TPU for CNN-based tasks, showcasing the technology's versatility and effectiveness. For instance, a research team from MIT successfully optimized a CNN specifically for detecting traffic signs in autonomous vehicles. They employed transfer learning with a MobileNet architecture, achieving remarkable inference times of less than 10 milliseconds per image on the Coral TPU. This efficiency allowed for real-time recognition without compromising vehicle safety, an essential factor in the deployment of autonomous driving systems where split-second decisions can be crucial.

In another innovative project, a group from Stanford University developed a TPU-accelerated CNN for drone-based environmental monitoring. Their model analyzed over 50,000 aerial images, identifying plant health and crop stress with impressive accuracy. By offloading computation to the Coral TPU, the drone was able to provide real-time feedback, making the solution scalable for agricultural use. This capability not only enhances productivity in farming but also supports sustainable practices by enabling timely interventions based on the health of crops.

Key Components of TPU-Based Optimization:

1. Model Optimization Techniques. Many CNN architectures are optimized through advanced techniques such as pruning and quantization. These methods reduce the model size without sacrificing performance, making it feasible to deploy them on edge devices with limited resources. The Coral TPU supports quantized models, which significantly improve energy efficiency and inference speed, crucial for battery-operated devices or those with restricted power availability. This optimization enables real-time applications in settings where power consumption is a major concern.
2. Instant Predictions. Models deployed on Coral TPU devices achieve instant predictions, making them ideal for real-time applications like robotics or smart surveillance, where response times are critical. For example, in smart surveillance systems, the ability to quickly analyze video feeds and identify potential threats enhances security measures and allows for prompt responses to incidents.
3. On-device Learning Capabilities. The Coral TPU enables on-device learning through fine-tuning, allowing models to be updated incrementally based on new data. This feature is particularly useful for dynamic environments where the system needs to adapt rapidly to new conditions. In industries like finance, where market conditions fluctuate frequently, the ability to update models in real-time ensures that predictions remain accurate and relevant.

The system utilized a lightweight CNN to identify authorized personnel with high precision, providing near-instant verification. This setup replaced traditional cloud-based solutions, significantly minimizing latency and eliminating privacy concerns associated with remote data storage. By processing data locally, organizations can enhance security and protect sensitive information, ensuring compliance with regulations regarding data privacy.

Another innovative project involved using the Coral TPU in wildlife monitoring. Researchers in Canada deployed camera traps powered by TPUs to detect endangered species in remote areas. The CNN-based model achieved over 95% accuracy in identifying animals in various lighting conditions, enabling real-time notifications to conservation teams. This timely information allows for more effective conservation efforts, as teams can respond swiftly to protect endangered species from threats like poaching or habitat loss.

Overall, the integration of specialized accelerators such as the Google Coral TPU has revolutionized the optimization of convolutional neural networks, allowing them to perform effectively in real-world scenarios. By addressing the unique challenges of deploying CNNs on hardware accelerators and demonstrating their capabilities across various applications, these advancements pave the way for more intelligent, efficient, and responsible machine learning solutions in the future.

Improving results of CNN

The process of creating a model optimized for Google Coral Edge TPU involves systematically handling data, applying transformations, and training models using tools like Roboflow. Below are the key steps to create an efficient image classification model for Coral devices:

1. Start by determining the specific aim of your dataset. For instance, if your goal is to identify various real-world objects like animals, vehicles, or plants, ensure that your dataset is tailored to this objective. This focus will direct the subsequent steps in preparing your dataset.
2. Compile a diverse array of images that depict each category or class you plan to include in your model. If your focus is on classifying animals, gather pictures of different types, such as cats, dogs, and birds.
3. Aim for an even distribution of images across classes to prevent bias in the training phase. This means collecting a similar number of images for each category.
4. Create separate folders for each class (e.g., one for cats, another for dogs, etc.) and place the relevant images into these folders. This organization is vital for effective data management and training.
5. Ensure that all images are accurately labeled according to their respective folders. Proper labeling is crucial for the model’s learning process, as it directly influences how well it can distinguish between classes.
6. For models designed for the Edge TPU, opt for TensorFlow Lite (TFLite) as the dataset format. This format is specifically optimized for mobile and edge devices, making it suitable for use with the Coral Edge TPU.
7. Download the structured dataset along with the labeling file, which helps map class labels and ensures the model accurately connects images to their respective labels during training.
8. Import the images and labels into TensorFlow, which will be the framework used to construct your model.

9. Select a lightweight architecture that is compatible with the Edge TPU, such as MobileNetV2 or EfficientNet. These models are designed for efficiency, making them ideal for edge devices with limited processing power.

10. Quantization reduces the model's size and enhances its inference speed by converting its weights from floating-point to integer format. This step is essential for optimizing the model's performance on the Edge TPU.

11. After quantization, use the Edge TPU Compiler to convert your TFLite model into a version that is compatible with the Edge TPU. This step further optimizes the model for efficient operation on Coral hardware.

12. Get your Coral USB Accelerator or Coral Dev Board ready by following the manufacturer's instructions for installation and connection, ensuring everything is configured properly to run your model.

13. Finally, evaluate the model's predictions using live inputs through the Coral TPU. This testing phase allows you to check the model's performance in real-time scenarios, confirming that it achieves the desired level of accuracy and responsiveness. .

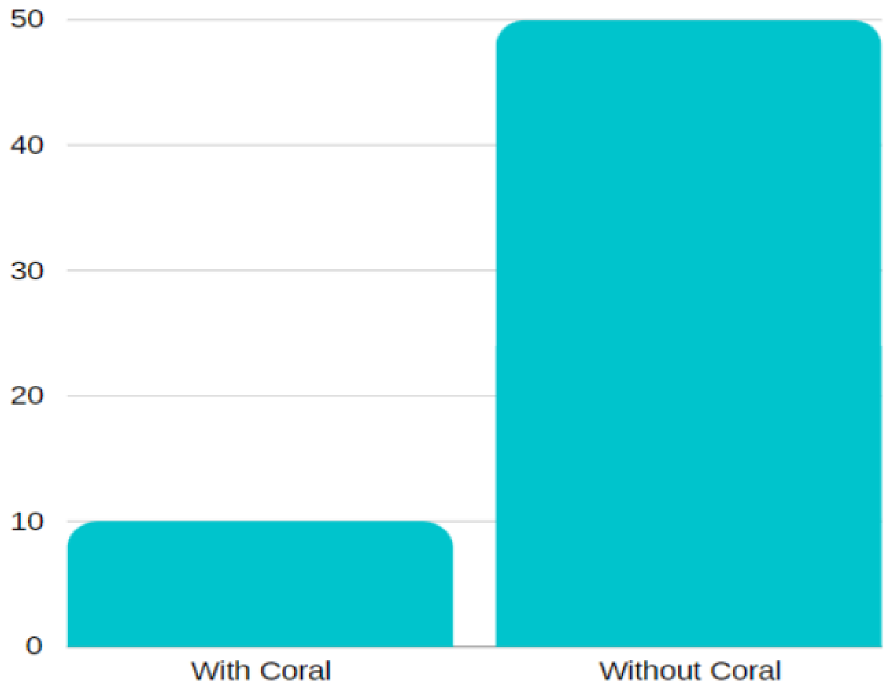

Fig. 1. Latency during the model's work

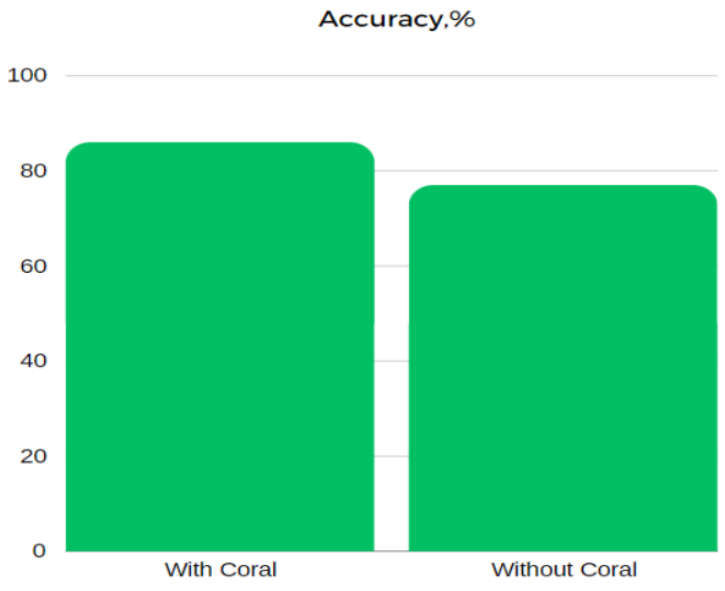

Fig. 2. Accuracy during model's work

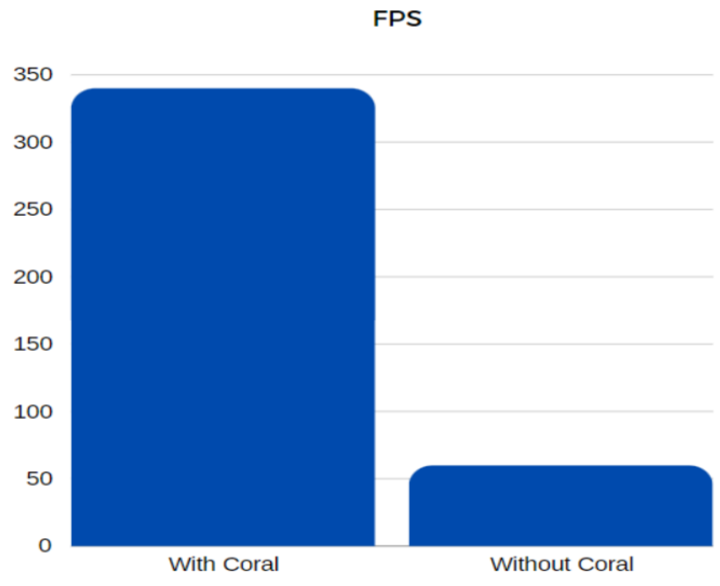

Fig. 3. Frames per second during model's work

The three graphs provide a comparative analysis of the performance of a machine learning model with and without the use of Google Coral for Edge TPU acceleration:

1) the first graph shows that when using Coral, the model achieves around 350 FPS, significantly outperforming the less than 50 FPS recorded without Coral. This stark difference indicates that Coral greatly enhances the speed of processing, making it suitable for real-time applications.

2) the second graph illustrates that the latency with Coral remains under 10 ms, whereas without Coral, it increases to approximately 50 ms. This lower latency with Coral means faster predictions, which is essential for applications that require immediate feedback, such as video analytics.

3) the third graph reveals that the model achieves around 82% accuracy when using Coral, compared to a lower accuracy when not utilizing the hardware acceleration. This means that Coral not only improves speed but also helps maintain a competitive level of accuracy, which is vital for effective model performance.

Overall, these graphs collectively emphasize the significant advantages of integrating Google Coral in machine learning tasks, particularly in terms of speed and responsiveness, which are crucial for applications demanding high performance.

Conclusions

The process of integrating Google Coral with machine learning models for enhanced performance presents a comprehensive approach to optimizing real-time applications. This integration begins with evaluating the baseline performance of models without hardware acceleration, which highlights the limitations in speed and latency. By incorporating Coral, users can significantly enhance frame rates, achieving impressive FPS rates that enable responsive applications.

Furthermore, the analysis of latency indicates that the use of Coral reduces processing times, which is crucial for applications requiring quick feedback, such as video analysis or object detection. This decrease in latency not only improves user experience but also opens up possibilities for real-time decision-making in various domains.

Accuracy assessments reveal that despite the substantial improvements in speed and latency, models utilizing Coral maintain competitive accuracy levels. This balance ensures that the integrity of the model's predictions is upheld while benefiting from the efficiency that hardware acceleration brings.

Overall, the combination of high-speed processing and low latency afforded by Google Coral allows for more effective deployment of machine learning models in practical scenarios. The ability to swiftly analyze data while maintaining accuracy makes it an invaluable asset in fields such as autonomous systems, robotics, and real-time analytics. This approach to model enhancement through hardware integration not only streamlines the workflow from development to deployment but also paves the way for innovative applications that demand both speed and reliability.

References

1. Prokscha R., Schneider M., Höß A. Efficient edge deployment demonstrated on YOLOv5 and coral edge TPU. In: Industrial Artificial Intelligence Technologies and Applications. River Publishers, 2023. Pp. 141-155.

2. Reidy B. C., et al. Efficient deployment of transformer models on edge TPU accelerators: A real system evaluation. In: Architecture and System Support for Transformer Models (ASSYST@ ISCA 2023). 2023.

3. Dubois E. Shared learning among distributed edge devices using Coral Edge TPU machine learning engines. 2021. PhD Thesis. Monterey, CA; Naval Postgraduate School.

4. Lentarıs G., et al. Performance and radiation testing of the Coral TPU co-processor for AI onboard satellites. In: 2023 European Data Handling & Data Processing Conference (EDHPC). IEEE, 2023. Pp. 1-4.
5. Cai H., et al. Coral-inspired asymmetrically porous radiative cooling biofilm with thermoplastic polyurethane-enhanced mechanical tolerance as building energy-saving envelope. ACS Applied Polymer Materials. 2023. Vol. 5, no. 12. Pp. 10053-10064.
6. Kovács B., et al. Object detection on TPU accelerated embedded devices. In: Computer Vision Systems: 13th International Conference, ICVS 2021, Virtual Event, September 22-24, 2021, Proceedings 13. Springer International Publishing, 2021. Pp. 82-92.
7. Mohammadi M., et al. Facial expression recognition at the edge: CPU vs GPU vs VPU vs TPU. In: Proceedings of the Great Lakes Symposium on VLSI 2023. 2023. Pp. 243-248.
8. Krauss D., et al. TrainUsIn – An AI training user interface for custom models on Coral Edge TPU. In: 2023 International Conference on Consumer Electronics-Taiwan (ICCE-Taiwan). IEEE, 2023. Pp. 809-810.
9. Drake D., Tang W. A self-supervised parking spot monitoring system using Google Coral Edge TPU. In: 2023 IEEE Sensors. IEEE, 2023. Pp. 1-4.
10. Joseph F. J., Nonsiri S., Monsakul A. Keras and TensorFlow: A hands-on experience. In: Advanced deep learning for engineers and scientists: A practical approach. 2021. Pp. 85-111.
11. Grattarola D., Alippi C. Graph neural networks in TensorFlow and Keras with Spektral [application notes]. IEEE Computational Intelligence Magazine. 2021. Vol. 16, no. 1. Pp. 99-106.
12. Sarang P. Artificial neural networks with TensorFlow 2. Apress: Berkeley, CA, USA, 2021.
13. Weber M., et al. Deeplab2: A TensorFlow library for deep labeling. arXiv preprint arXiv:2106.09748, 2021.
14. David R., et al. TensorFlow Lite Micro: Embedded machine learning for TinyML systems. Proceedings of Machine Learning and Systems. 2021. Vol. 3. Pp. 800-811.
15. Demosthenous G., Vassiliades V. Continual learning on the edge with TensorFlow Lite. arXiv preprint arXiv:2105.01946, 2021.
16. Adi S. E., Casson A. J. Design and optimization of a TensorFlow Lite deep learning neural network for human activity recognition on a smartphone. In: 2021 43rd Annual International Conference of the IEEE Engineering in Medicine & Biology Society (EMBC). IEEE, 2021. Pp. 7028-7031.
17. Manor E., Greenberg S. Custom hardware inference accelerator for TensorFlow Lite for microcontrollers. IEEE Access. 2022. Vol. 10. Pp. 73484-73493.
18. Konaite M., et al. Smart hat for the blind with real-time object detection using Raspberry Pi and TensorFlow Lite. In: Proceedings of the International Conference on Artificial Intelligence and its Applications. 2021. Pp. 1-6.
19. Pandey J., Asati A. R. Lightweight convolutional neural network architecture implementation using TensorFlow Lite. International Journal of Information Technology. 2023. Vol. 15, no. 5. Pp. 2489-2498.
20. Reda M., et al. Agroaid: A mobile app system for visual classification of plant species and diseases using deep learning and TensorFlow Lite. Informatics. MDPI. 2022. P. 55.
21. Breton S. N., et al. Deciphering stellar chorus: Apollinaire, a Python 3 module for Bayesian peakbagging in helioseismology and asteroseismology. Astronomy & Astrophysics. 2022. Vol. 663. A118.

Tymur Isaiev **Тимур Ісаєв**	Master Student of Computer Engineering & Information Systems Department, Khmelnytskyi National University e-mail: tymuri1112@gmail.com	Магістрант кафедри комп’ютерної інженерії та інформаційних систем, Хмельницький національний університет
Tetiana Kysil **Тетяна Кисіль**	Candidate of Physical and Mathematical Sciences, Associate Professor of Computer Engineering & Information Systems Department, Khmenlnytskyi National University https://orcid.org/0000-0002-4094-3500 e-mail: kysil_tanya@ukr.net	Кандидат фізико-математичних наук, доцент кафедри комп'ютерної інженерії та інформаційних систем, Хмельницький національний університет