Characterizing and Accelerating End-to-End EdgeAI Inference Systems for Object Detection Applications

Yujie Hui
hui.82@osu.edu
The Ohio State University
Columbus, Ohio, USA

Jeffrey Lien
jlien@novumind.com
NovuMind Inc
Santa Clara, California, USA

Xiaoyi Lu
xiaoyi.lu@ucmerced.edu
University of California, Merced
Merced, California, USA

ABSTRACT

Modern EdgeAI inference systems still have many crucial limitations. In this paper, we holistically consider implications and optimizations of EdgeAI inference systems for object detection applications in efficiency and accuracy. We summarize three intrinsic limitations of current-generation EdgeAI inference systems based on our observations (i.e., less compute capabilities, restrictions of operations, and accuracy loss due to numerical precision). Then we propose three approaches to improve end-to-end performance and prediction accuracy: 1) Utilizing parallel computing designs and methods to solve computational bottlenecks; 2) Applying domain-specific optimizations to mostly eliminate accuracy loss; 3) Using higher-quality input data to saturate the processors and accelerators. We also provide five recommendations for end-to-end EdgeAI solution deployments, which are usually neglected by EdgeAI users. In particular, we deploy and optimize two real object detection applications (2D and 3D) on two EdgeAI inference systems (NovuTensor and Nvidia Xavier) with widely used datasets (i.e., MS-COCO, PASCAL-VOC, and KITTI). The results show that runtime performance can be accelerated by up to 2X on NovuTensor and the mean average precision (mAP) can be increased by 46% through applying our proposed methods.

ACM Reference Format:
Yujie Hui, Jeffrey Lien, and Xiaoyi Lu. 2021. Characterizing and Accelerating End-to-End EdgeAI Inference Systems for Object Detection Applications. In *The Sixth ACM/IEEE Symposium on Edge Computing (SEC '21), December 14–17, 2021, San Jose, CA, USA.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3453142.3491294

1 INTRODUCTION

In recent years, we see many datacenter-based Artificial Intelligence (AI) systems are taking advantage of various technologies (e.g., multi-core CPUs, accelerators, and high-speed interconnects) to accelerate the Deep Learning training process [20, 33]. Different from the requirement of training, Deep Learning inference tasks need much less computing resources and energy, which is because the input data for inference is only processed once in the forward path of a trained network and it does not need to go through the time-consuming and resource-hungry forward-backward computation processes iteratively. Hence, deploying Deep Learning inference systems on edge computing platforms has become a promising approach.

EdgeAI systems utilize popular AI solutions like neural networks to process real-time data. Many EdgeAI processors or accelerators are designed by different vendors for inference systems and manufactured on various platforms. For instance, Nvidia Xavier [5] is equipped with ARM CPU, Volta GPU, and a customized Deep Learning accelerator. EdgeAI inference systems can benefit from the heterogeneous computing of various processors. Compared with the well-studied datacenter-based AI training systems [20, 24, 33], the insufficiency of understanding and optimization of EdgeAI inference systems may cause problems of under-utilizing EdgeAI hardware platforms, leading to low performance and accuracy for AI inference applications. In particular, this paper summarizes that there are at least three intrinsic limitations of current-generation EdgeAI inference systems, which are:

(1) Less compute capabilities due to low power consumption. Current-generation processors and accelerators designed for EdgeAI inference systems usually consume very low power (10 Watt ~ 50 Watt) compared with that of GPGPUs. Delivering real time performance with limited energy and compute capabilities is challenging.

(2) Restrictions of supported AI inference operations. Many EdgeAI accelerators are designed to only support a fixed set of AI inference operations due to technology and market trade-offs. As a result, some deep neural networks with complex operations cannot be deployed on these EdgeAI platforms. This leads to a critical dilemma: *should we choose to deploy complex AI models on high-end but expensive EdgeAI platforms? Or should we choose affordable approaches but can still maintain similar performance and accuracy as running on the high-end EdgeAI platforms?*

(3) Accuracy loss due to lower numerical precision. Reducing numerical precision via quantization, is a common method to speed up Deep Learning inference [42]. However, accuracy will suffer because of potential information loss during quantization.

Due to these limitations, this paper first quantitatively characterizes EdgeAI inference systems from performance and accuracy angles. Our analysis in Section 3 demonstrates the harmful effects of these limitations. Based on our observations, we propose several approaches to accelerate the runtime performance and increase the prediction accuracy of EdgeAI systems.

In terms of end-to-end runtime performance, we apply parallel computing designs to optimize the object detection applications on EdgeAI inference systems. We also exploit the parallelism of EdgeAI inference hardware features. Concerning prediction accuracy, this

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
SEC '21, December 14–17, 2021, San Jose, CA, USA
© 2021 Association for Computing Machinery.
ACM ISBN 978-1-4503-8390-5/21/12…$15.00
https://doi.org/10.1145/3453142.3491294

Authorized licensed use limited to: Univ of Calif Merced. Downloaded on August 17,2022 at 23:58:13 UTC from IEEE Xplore. Restrictions apply.

paper proposes a hybrid calibration method for quantization, which mostly eliminates accuracy loss.

More interestingly, we analyze the relations between the resolutions of input images and prediction accuracy. We wondrously find that running simple neural networks with higher resolution input images may get higher or similar accuracy than using complex neural networks with regular resolution images for some object detection scenarios. This implies that we can avoid deploying complex neural networks on expensive EdgeAI hardware platforms. Alternatively, we can choose affordable AI hardware with simpler models and feed higher resolution images. This approach essentially implies that we need an efficient EdgeAI system design, which is not only optimizing the model computation but also data movement (i.e., communication). In this case, more computing capabilities can be used to optimize this workflow, which essentially can solve the dilemma discussed above. With our approaches, we can alleviate the burden of designing complex neural networks. The details are discussed in Section 5.2. Energy efficiency is another crucial factor for EdgeAI systems. Section 6.1 discusses the effect on power consumption of our proposed methods.

This paper makes the following contributions:

- We deploy the killer computer-vision application for edge (2D&3D object detection) on two EdgeAI inference systems (i.e., NovuTensor and Nvidia Xavier) based on real datasets (e.g., MS-COCO, PASCAL-VOC, and KITTI). We argue that the EdgeAI accelerators on these architectures have little potential if they are not coupled with hardware and software methods to reduce the computation and communication overhead. (Section 3)
- We present a fresh view that leveraging parallel computing designs and methods on EdgeAI inference systems is essential for the end-to-end performance based on our characterization. We demonstrate that exploiting the EdgeAI accelerators is not straightforward but requires domain-specific knowledge on software and hardware. Our methods for EdgeAI can make a real difference in end-to-end performance (Section 4). Our methods can save up to 20% energy consumption and maintain the power consumption on NovuTensor. (Section 6.1)
- We propose a domain-specific hybrid calibration method, which mostly eliminates the accuracy loss due to uniform quantization. The proposed method is evaluated on an EdgeAI inference system deployed with a LiDAR 3D object detection application. The evaluation results show that our method outperforms other conventional calibration methods, which increases the mAP by 64% at most. (Section 5.1)
- We propose a methodology to improve the accuracy of object detection on EdgeAI platforms, which converts the challenges of EdgeAI inference systems from AI algorithms to computing capabilities. In particular, deploying simpler neural networks with higher resolution input images can deliver higher or similar accurate results than deploying more complicated neural networks that are unsupported by EdgeAI systems (Section 5.2). Our methodology is application specific but it could be useful for many other important computer-vision related applications.

In a nutshell, this paper provides an important guidance (summarized as five recommendations highlighted in the paper), which are usually neglected by EdgeAI users. Our recommendations aim to improve object detection tasks on EdgeAI inference systems.

2 BACKGROUND

This section introduces some necessary background information.

2.1 Object Detection

Object detection is the most popular Deep Learning task in industry for EdgeAI accelerators, which aims to classify and localize objects of interest. Object detection tasks can be grouped into two genres: *2D object detection* and *3D object detection*. An object detection system is able to predict several bounding boxes to localize the objects in the 2D images or 3D point clouds. MS-COCO [27] and PASCAL-VOC [14] are two widely used datasets for 2D object detection tasks. In the past decade, CNN based object detection systems have been emerging since deep convolutional neural networks are able to learn robust features from input data [25].

3D object detection usually needs more information to predict 3D bounding boxes. LiDAR sensors provide accurate 3D point clouds from surrounding environments. Most works on 3D object detection rely on LiDAR point clouds [34, 35, 48]. CNNs are capable of processing the images encoded from LiDAR point clouds. KITTI [17] is a popular dataset, which provides LiDAR point clouds as well as their corresponding 2D images.

2.2 EdgeAI Processors

GPGPU plays an important role in Deep Learning training due to its powerful compute capability. However, Deep Learning inference tasks might not need GPGPUs, since the cost and power consumption of GPUs are too high. Hence, custom EdgeAI processors have emerged recently, which are able to perform Deep Learning inference tasks in an energy efficient way. Many EdgeAI processors with specialized designs target CNN processing due to the recent popularity of CNNs [43]. In addition, co-designs of CNN models and hardware help to further increase throughput and reduce power consumption. The co-design methods include reducing numerical precision and reducing the number of operations. In this paper, we conduct our experiments on two EdgeAI inference systems that aim to accelerate CNN operations (i.e., Nvidia Xavier and NovuTensor) as shown in Table 1.

Nvidia Xavier is an embedded system on a module, containing a 512-core Volta GPU, two Deep Learning Accelerators (DLA), a Carmel ARM CPU, and 16GB memory [15]. Nvidia Xavier integrated high-end GPGPUs with CUDA cores and Tensor cores. Xavier also provides low-end DLAs with a power consumption of only 0.5 - 1.5 Watt that target processing CNN inference. Nvidia Xavier is supported by TensorRT [8], an inference library that offers model optimization and runtime inference acceleration. Nvidia Xavier supports multiple numerical precisions (i.e., FP32, FP16, and INT8). Users can configure power modes at 10W, 15W, and 30W. The processing steps of inference are conducted by the processors and accelerators equipped within the platform. Table 1 shows the combined CPU and accelerator power consumption specification.

Authorized licensed use limited to: Univ of Calif Merced. Downloaded on August 17,2022 at 23:58:13 UTC from IEEE Xplore. Restrictions apply.

NovuMind's NovuTensor is an ASIC that focuses on accelerating various CNN inference applications, while it has an order of magnitude less cost than the Xavier module. NovuTensor is equipped with two neural processing unit (NPU) cores. The main advantage of NovuTensor is its native tensor processing [29], in which the 3D convolutional operations can be processed without the typical General Matrix Multiply (GEMM) method of unfolding the 3D tensors to 2D. Without this unfolding and folding overhead, significant improvements in Silicon Area utilization and thus power efficiency, can be achieved. Additionally, since neural network computation is deterministic and predictable, simplifications in data flow and memory hierarchy can further improve key metrics desirable by edge devices. This domain-specific design enables NovuTensor to deliver up to 15 Tera Operations Per Second (TOPS) of CNN based compute with a power consumption of 15W. NovuTensor is manufactured with a PCIe interface with 2GB device memory to connect to a host CPU. Note that NovuTensor is a PCIe-based card, which needs the co-processing of the host. To conduct the end-to-end evaluation, the host CPU will be involved in the pre-/post-processing. Then it is necessary to consider the power consumption of host CPU for PCIe-based AI accelerators. The thermal design power (TDP) of the host CPU used in our experiments is 91W [1]. Nevertheless, the actual power consumption may not achieve the reported TDP during the inference. The detailed evaluations about power consumption will be discussed in Section 6.1. To achieve optimal performance, EdgeAI inference systems typically only process one input data at a time. NovuTensor can process data in batch of two without any performance sacrifice due to its native batch-based design.

Table 1: Specifications of EdgeAI Platforms

Specifications	NovuTensor	Nvidia Xavier
Precisions	INT8	FP32/FP16/INT8
# of NPU Cores	2	512-Core GPU
TOPS (up to)	15	32
Memory	2 GB	32 GB (shared)
Peak Power (Watt)	15	30/15/10
Peak Host Power (Watt)	91	

2.3 Deployment Flow for EdgeAI Platforms

To perform Deep Learning inference tasks on an edge computing platform, several steps are required to deploy the trained neural network on the EdgeAI system. The overview of a deployment process is illustrated in Figure 1. We summarize three steps to generate a deployable runtime engine for EdgeAI systems. First, since the neural network model can be trained by a variety of frameworks such as TensorFlow [9] and Caffe [22], developers need to convert the format of the model to the platform desired format. Therefore users need to convert the model format prior to feeding it into the hardware. The SDKs may help to parse the undesired model format. For example, NovuTensor requires Caffe [22] model as the input while TensorRT provides a parser function to parse Caffe model to the desired format. Second, the trained model may need to be quantized to a lower numerical precision. Numerical precision is a trade-off between prediction accuracy and computing speed that developers should be aware of. Third, using SDK of the EdgeAI platform compiles the quantized model into a deployable runtime engine. The runtime engine can be deployed on the hardware. Runtime libraries from SDKs provide APIs for user applications to launch computing and prediction.

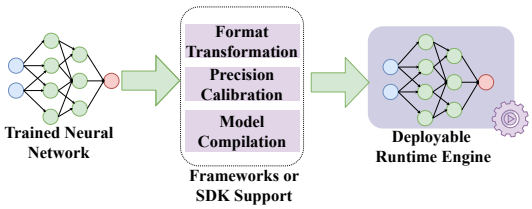


Figure 1: Deep Learning Inference Applications Deployment Flow for EdgeAI Platforms

3 CHARACTERIZATION AND MOTIVATION

This section introduces the characteristics of EdgeAI systems for Deep Learning inference tasks and presents the motivation of further optimizations.

3.1 Limitations of EdgeAI Platforms

EdgeAI platforms with Deep Learning accelerators aim to accelerate computationally intensive tasks. However, some intrinsic limitations exist and can affect the performance of Deep Learning applications. Based on our experience, this section presents three major intrinsic limitations of current-generation EdgeAI platforms.

Power Consumption vs. Computing. EdgeAI inference systems are usually designed to consume very low power. However, they may not be suitable for computationally intensive tasks. In addition, pre-processing large input data on the CPUs of edge platforms can be a non-negligible overhead. To achieve an acceptable inference latency with limited power, co-designs of hardware and software for EdgeAI systems are important and challenging.

Restrictions of supported AI inference operations. Current-generation EdgeAI platforms are limited to several Deep Learning operations due to their hardware designs. Considering those limitations, some complex neural networks cannot be deployed on the EdgeAI platforms or need to be adjusted. For instance, the *Reorg* and *Route* layers in YOLOv2 [38], which reshape and concatenate the tensors, are not supported by NovuTensor and Nvidia Xavier's DLA. Hence, a hardware friendly model that replaces those unsupported layers is needed. Nvidia Xavier can use GPU if a layer is not supported by the DLA. Modifying and retraining a hardware friendly model can be time consuming for developers, which raises the barrier for deploying neural networks on EdgeAI platforms.

Accuracy loss due to lower numerical precision. Reducing precision, via quantizing floating point numbers (FP32) to fixed point numbers (INT8) for computation, is a method to speed up Deep Learning computation by reducing energy and silicon area costs, and thus allowing for more computational units. Deep Learning inference can benefit from quantization in two aspects: 1) reducing the parameter storage size of neural networks and 2) speeding up the computation and reducing power consumption by taking advantage of integer computation [45]. Nevertheless, due to lower

Authorized licensed use limited to: Univ of Calif Merced. Downloaded on August 17,2022 at 23:58:13 UTC from IEEE Xplore. Restrictions apply.

numerical precision representations, the accuracy of inference applications might drop. Many existing research studies propose different quantization algorithms, which are reasonably effective for Deep Learning inference workloads [40, 42].

However, the accuracy loss is still not negligible, though previous work shows that the accuracy loss can be controlled within 1% [45]. According to our experimental results, the accuracy can suffer without any fine tuning. The details are discussed in Section 3.3.

3.2 Performance Characterization

Deep Learning inference applications at the edge are latency critical. Users need to get the prediction results from the edge-side as soon as possible. Hence, achieving real-time performance is one of the most essential design targets for EdgeAI inference systems. To obtain human understandable prediction results, an end-to-end AI application generally needs three steps, including pre-processing, inference, and post-processing:

- **Pre-processing:** parses and manipulates the raw input data, which will be fed into the Deep Learning networks.
- **Inference:** forwards the input data through different Deep Learning network topologies (i.e., CNNs and RNNs) and performs the computation on the input data.
- **Post-processing:** extracts the output from Deep Learning networks and generates the final human understandable prediction results.

The computation of Deep Learning has been improved significantly with the help of AI accelerators (i.e., GPGPUs and TPUs) over the decades. But pre-processing could be a new bottleneck of AI applications due to the diversity of the raw data. For example, pre-processing raw LiDAR data for 3D object detection could be a computationally intensive task. Table 2 compares the runtime performance of some related work [31] (measured on an NVIDIA 1080Ti GPU). To get a better sense of the numbers in Table 2, we need to be aware of the acceptable latency for autonomous cars to detect potential hazards. A common perception of real-time is the ability to process 30 frames per second, which is similar to human eyes [10]. In other words, it would be acceptable for an end-to-end inference application if it can process an image in 33ms.

Table 2: Runtime Performance Comparison for 3D Object Detection Tasks on KITTI [31]

Methods	Inference (ms)	Pre/Post Process (ms)	Total (ms)
LaserNet[31]	12	18	30
PIXOR[46]	35	27	62
VoxelNet[48]	190	35	225
MV3D[11]	-		360
AVOD[26]	80	20	100

Raw LiDAR data is a set of 3D points captured by 3D scanners in the surrounding environment. Each point is stored in (x, y, z) coordinates with an additional axis (i.e., reflectant value and color). One way to process raw LiDAR data is encoding and discretizing the point clouds into some fixed grids [13]. Complex-Yolo [41] encodes the raw point cloud data into a single birds-eye-view(BEV) RGB-map. The encoded RGB input feature map has three channels (i.e., density, height, and intensity channel).

An experiment is conducted on Nvidia Xavier, which is set to 15W mode, to demonstrate that pre-processing is a bottleneck of LiDAR processing AI applications. Pre-processing, inference, and post-processing latencies are collected separately in our experiments and the experimental results are illustrated and broken down in Figure 2. Performance increases from the reduction of numerical precision due to the lesser computation costs. But the pre-processing accounts for 68% of the end-to-end processing latency when computes in INT8 precision. Hence, the pre-processing becomes the bottleneck of this AI application.

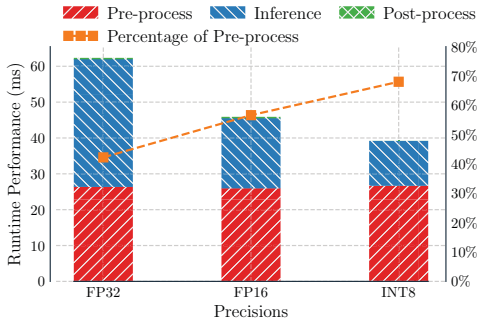


Figure 2: End-to-end Latency Breakdown of LiDAR Processing on Nvidia Xavier with Varied Precisions

Another problem that could impact the application performance on EdgeAI platforms is the overhead of transferring data from/to host memory to/from device memory. The data transfer latency and the computation latency on device make up the whole inference latency in Figure 2. We conduct experiments on different AI platforms (i.e., NovuTensor, Nvidia Xavier, and Nvidia 1080Ti) to demonstrate the necessity to optimize the data transfer. A model of YOLOv2 is deployed and the resolution of input feature map is 3×608×608. The experimental results are listed in Table 3. Since the memory of Nvidia Xavier is shared between the ARM processors and GPU, the data transfer speed of the GPU is not limited by the PCIe bus like the case of using a traditional GPU. On the other hand, data will be transferred between host and device memory through the PCIe bus for NovuTensor and Nvidia 1080Ti. Since NovuTensor's PCIe interface has fewer lanes than Nvidia 1080Ti, it takes more time to transfer the input data to NovuTensor. Hence, it is necessary to optimize the data transfer overhead for PCIe based EdgeAI inference systems such as NovuTensor.

Table 3: Breakdown of Data Transfer and Execution

Devices	Data Transfer (ms)	Execution (ms)	Percentage of Data Transfer
Xavier DLA	1	21	4.5%
NovuTensor	23	12.9	64%
Nvidia 1080Ti	0.35	6.8	5%

3.3 Accuracy Characterization

One of the most important targets of EdgeAI platforms is predicting accurate and reliable results based on real-world input data.

Authorized licensed use limited to: Univ of Calif Merced. Downloaded on August 17,2022 at 23:58:13 UTC from IEEE Xplore. Restrictions apply.

However, computing in lower numerical precision (e.g., INT8) can sometimes deliver poor prediction results which are unreliable. This section introduces a general quantization procedure for different EdgeAI platforms and exemplifies the negative effect on accuracy from quantization based on our experimental results.

The quantization procedure ultimately delivers a mapping relation between 32-bit floating point numbers and 8-bit integers with the pre-defined scale factors. The mapping function can be defined as follows [40]:

$$Q(w) = sgn(w) \cdot \Delta \cdot min(\lfloor \frac{|w|}{\Delta} + 0.5 \rfloor, \frac{M-1}{2}) = \Delta \cdot z \quad (1)$$

where the $sgn(\cdot)$ is the sign function, Δ is the scale factor, M is the level of representations, w is the floating-point weights or activations, and z is the represented integer. For instance, if the numerical precision is INT8, the range of z is from -128 to 127. The weights or activations can be represented by $-128\Delta, -127\Delta, \ldots -\Delta, 0, \Delta, \ldots, 127\Delta$.

Appropriate scale factors should be determined before deployment to deliver optimal prediction accuracy. Calibration is a process to determine optimal scale factors. EdgeAI platforms may apply different standards to select scale factors. For example, NovuTensor selects the scale factors that minimize the Mean Squared Error (MSE) between the original values and the quantized values. Equation 2 is applied to determine the MSE after quantization where N is the number of weights or activations of each layer and w_i is the weights or activations.

$$E = \frac{1}{N} \sum_{i=1}^{N} (Q(w_i) - w_i)^2 \quad (2)$$

The SDK of NovuTensor and TensorRT [8] for Nvidia Xavier both provide quantization tools to generate the scale factors automatically. A small set of training data (calibration data) are used to perform calibration. A histogram of each layer's activations and weights is collected based on the calibration data. More calibration methods will be discussed in Section 5.1.

We find that computing in lower numerical precision can incur a large accuracy drop in some cases. In our experiments, a 3D object detection application based on KITTI [17] is deployed on NovuTensor and Nvidia Xavier and mean average precision (mAP) is chosen as the metric to evaluate accuracy. An identical model is deployed on NovuTensor, Nvidia Xavier, and an Nvidia 1080 Ti GPU. According to the experimental results, the mAP drops 88% and 46% on Nvidia Xavier and NovuTensor, respectively, when the activations and weights are quantized to 8-bit integers by using their default SDKs. Though INT8 quantization has been widely deployed, huge accuracy degradation can still occur. We suspect when training with batch-normalization, the channel-to-channel dynamic range difference is large. Then the layer-wise quantization may struggle.

3.4 Motivation

Deploying Deep Learning inference applications on EdgeAI platforms can bring several unexpected problems based on our empirical experience. Firstly, the end-to-end runtime performance can suffer from the computational overhead of pre-/post-processing. Secondly, data transferring between device memory and host memory is a non-negligible overhead. Thirdly, the reliability of the inference applications can be affected by numerical precision. These observations are easily neglected by EdgeAI users, who usually choose to rely on using the default SDKs with default configurations to achieve agile but not efficient development.

In this paper, we propose methods to optimize the runtime performance and increase the prediction accuracy of AI inference applications deployed on EdgeAI platforms. Our proposed methods can guide designs and deployment for EdgeAI inference systems. Though some of our optimizations might be well-known, if users do not use them they will lose performance. This paper demonstrates that the end-to-end deployment and optimization processes with finer granularities are needed even if EdgeAI SDKs are available for different platforms.

4 PERFORMANCE ACCELERATION

We aim to improve the runtime efficiency of EgdeAI systems at first. This section introduces the designs and techniques proposed in this paper to accelerate the runtime performance of end-to-end EdgeAI applications in two aspects. Results from our experiments show that the proposed methods are able to effectively optimize runtime performance. Our methods are also general and helpful for different EdgeAI platforms as well as different Deep Learning tasks.

4.1 Pre-processing Optimization

To accelerate the pre-processing of raw LiDAR data, the encoding process is analyzed and profiled in our experiments. As we discuss in Section 3.2, encoding the LiDAR point clouds is computationally intensive and each pixel is operated upon independently. It is very intuitive to apply parallel computing designs to the encoding process. Particularly, we put OpenMP [6] and Intel MKL [4] into practice to parallelize the encoding. We apply compiler directives provided by OpenMP to create parallel blocks for accelerating encoding. We use vector mathematics functions provided by Intel MKL in our implementation, which optimize the computation of each of the vector elements. In particular, *vsAdd*, *vsLn*, and *vsDiv* are utilized.

Our evaluation results show that applying our approaches to Deep Learning applications is promising to optimize the runtime performance. We refer to an implementation of encoding raw LiDAR data into BEV images [11] as the baseline. We optimize the memory allocation and usage at the beginning. Then, OpenMP and MKL are applied to optimize the pre-processing individually. The experiments are conducted on NovuTensor and Nvidia Xavier. The specifications of the environments are listed in Table 4.

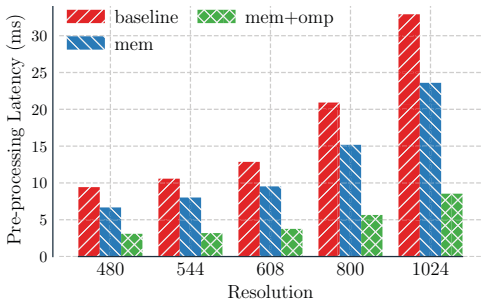
The evaluation results are shown in Figure 3. Note that the x axis in each subfigure denotes the resolution of the encoded images. For example, if the value of x is 480, the resolution of encoded image is $3 \times 480 \times 480$. In general, the performance of pre-processing on NovuTensor performs better than that on Nvidia Xavier, due to the more powerful CPU with higher frequency. Three optimizations applied to the encoding process for encoding raw LiDAR data are compared: 1) *mem*: The LiDAR pre-processing implementation [11] is inefficient on the memory allocation and usage. Buffers will be re-allocated and re-initialized every time for incoming raw LiDAR data. We avoid this overhead by designing and using a page-size

Authorized licensed use limited to: Univ of Calif Merced. Downloaded on August 17,2022 at 23:58:13 UTC from IEEE Xplore. Restrictions apply.

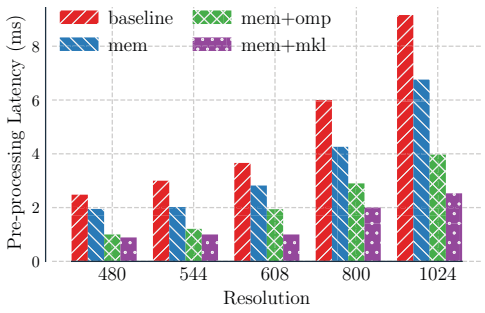
aligned circular buffer pool instead of the original design. In specific, page-size aligned buffers are allocated from the circular buffer pool and can be reused. 2) *mem + omp*: memory optimization together with OpenMP, and 3) *mem + mkl*: memory optimization together with MKL. The experimental results are shown in Figure 3. Overall, utilizing OpenMP and MKL on the encoding process has huge benefits. Particularly, the performance for pre-processing improves up to 81% by utilizing MKL, while up to 74% improvement is gained by using OpenMP on Nvidia Xavier. On NovuTensor, the latency of processing raw LiDAR data into BEV image with resolution of 480×480 is only $0.89ms$ in the best-case scenario. The evaluation results indicate that our proposed designs and employed techniques are able to effectively improve the performance of pre-processing during runtime on different EdgeAI platforms. The designs and techniques for optimizing the pre-processing can also be applied to other EdgeAI inference systems.

Table 4: Specifications of experimental environments

Specification	NovuTensor	Nvidia Xavier
Processor	Intel Core i7-7700K	Carmel ARMv8
# of Cores	4	8
Frequency	4.2GHZ	1.2GHZ
RAM (DDR)	32 GB	32 GB (shared)
OS	Ubuntu 16.04	Ubuntu 18.04



(a) Nvidia Xavier



(b) NovuTensor

Figure 3: Performance Comparisons for Encoding Raw LiDAR data with varied environments, input resolutions, and optimization methods

Recommendation #1: The combination of EdgeAI inference systems with parallel computing designs is promising for end-to-end inference tasks. Even leveraging simple designs and libraries can boost the end-to-end performance significantly.

4.2 Pipelining Design

The overhead of transferring data between host and device is non-negligible for some EdgeAI systems as mentioned in Section 3.2. Performing neural network computations on AI accelerators generally follows an execution sequence: copy data from host to device → compute on the device → copy computed data from device to host. Copying data between host and device memory could be a huge overhead with respect to the volume of input and output data of the neural networks. For example, hundreds of MBs of data per second will be transferred between host and device when the input is a high definition (HD) image. We characterize the latency, as listed in Table 3, the transferring of an input feature map with the dimension of $3 \times 608 \times 608$ from host DRAM to device memory via DMA ($23ms$) is much slower than the computation performed by the process engines on the device ($12.9ms$) for NovuTesnor. In order to accelerate the runtime performance, a design is required to resolve the bottleneck of data transfer.

An intuitive optimization method is incorporating the concurrent execution approach, which overlaps the data transfer and the neural network computation. The execution pipeline design of NovuTensor is illustrated in Figure 4. NovuTensor communicates with the CPU via the PCIe interface. A DMA will be issued to transfer the input/output data between host memory and device memory when the data is ready. The optimal system parameters such as the number of threads and buffer size need to be tuned due to the bottleneck of PCIe-based data movement on the NovuTensor system (fewer lanes). To achieve the best overlapping efficiency, we use four threads to do the pre-processing, data transfer, NPU execution, and post-processing separately. Pipelining the CPU tasks (pre-/post-processing), data transfer, and NPU tasks can boost the runtime throughput of Deep Learning inference applications. Our proposed pipeline design speeds up the runtime performance by 29% compared with the sequential execution mode. The pipeline design can also be adopted to Nvidia Xavier. TensorRT provides an asynchronous API called *enqueue* cooperated with asynchronous CUDA memory copy API to perform inference. Two threads can be launched to pipeline the procedure. The thread will be blocked when the input queue is full or *cudaStreamSynchronize* is invoked.

Recommendation #2: It is important to consider implications and optimizations on the **end-to-end** EdgeAI inference applications during deployment. Overlapping data movements and computation can make a real difference in the end-to-end performance of inferences and data processing in particular.

Authorized licensed use limited to: Univ of Calif Merced. Downloaded on August 17,2022 at 23:58:13 UTC from IEEE Xplore. Restrictions apply.

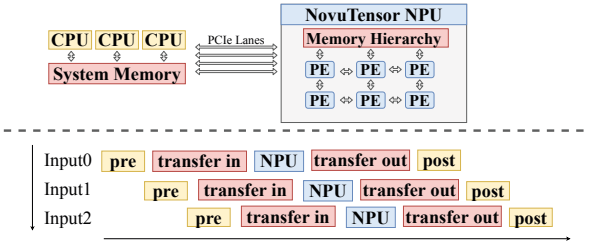


Figure 4: Pipeline Design for NovuTenosr's Architecture

4.3 Hardware & Software Co-design

4.3.1 Batch Size. NovuTensor and Nvidia Xavier are designed to support various batch sizes for Deep Learning inference workloads. Even though increasing the batch size can improve the throughput of applications, Deep Learning inference is a latency critical task that is different from Deep Learning training. To obtain the optimal performance on both latency and throughput, experiments are conducted on NovuTensor and Xavier with various batch size from 1 to 32. Input feature maps with dimension of $3 \times 608 \times 608$ are fed into a YOLOv2 network. To fairly compare performance, we only evaluate the performance of Xavier in INT8 precision. Figure 5 illustrates the experimental results. The latencies of batch 16 and 32 of Xavier are not shown because they are too slow for inference tasks in the autonomous driving domain. We hypothesize that the throughput decreasing of batch 8 results from imperfect optimization of TensorRT. Nvidia Xavier's accelerators are able to process 196 encoded images per second at most in the max power configuration with batch 4 and 8. Since NovuTensor is designed to support batch-2 natively, the latency will not suffer when computing in the batch-2 mode compared with computing in the batch-1 mode. Note that NovuTensor only supports batch-1 and batch-2 modes. While these platforms can deliver better throughput while processing more images at one time, it is more crucial to complete the tasks with low latency. Hence, we configure the NovuTensor to batch-2 mode and Nvidia Xavier to batch-1 mode for all the subsequent experiments in order to obtain the lowest latency and good throughput.

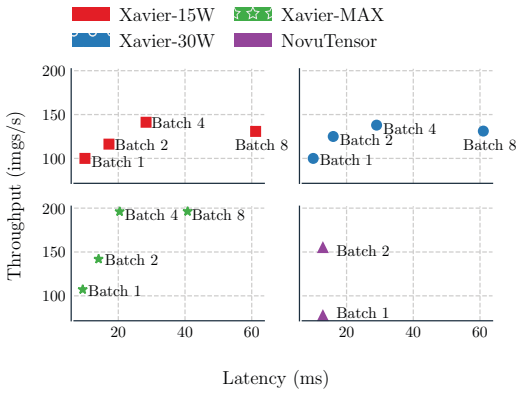


Figure 5: Throughput and Latency for NovuTensor and Xavier with Increased Batch Size from 1 to 8

4.3.2 Hardware Configuration. Users are allowed to select the precisions (i.e., FP32, FP16, and INT8) on which Nvidia Xavier performs the computations. In addition, users can configure the platform to execute on different power modes (e.g., max power, 30W, and 15W), where all the computing units will compute at different clock frequencies. Reducing the numerical precision and configuring the power modes can deliver better runtime performance as shown in Figure 6. In the best case, the runtime performance reaches 20*ms* by applying our proposed designs and optimizations when the device computes in the max power mode and INT8 precision.

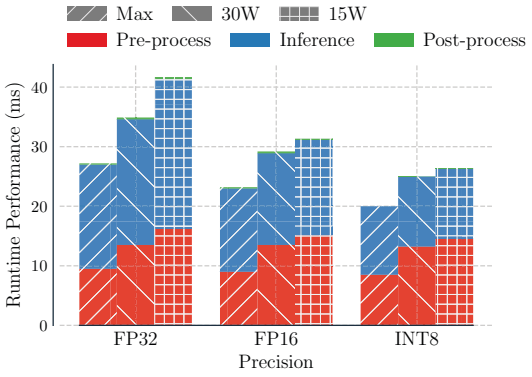


Figure 6: End-to-end Runtime Performance Comparison for BEV object detection with Different Precisions and Power Modes on Nvidia Xavier

Recommendation #3: Domain-specific knowledge on both hardware and software are required to fully exploit the capabilities of EgdeAI inference accelerators.

4.4 Overall Improvement

Real time performance is crucial for Deep Learning inference applications because users need to get the prediction results from the EdgeAI platforms as soon as possible to make optimal decisions in time. Two optimization methods have been discussed in Section 4.1 and Section 4.2. We apply the methods to 3D object detection application on NvouTesnor and Nvidia Xavier. The backbone of the application is YOLOv2 [38] neural network trained with KITTI dataset [17]. 6000 files of cloud points are used as the training set and 1481 files are used for inference. Table 5 shows the comparison of runtime latency and throughput between the baseline and the results taken by combining our proposed methods. Nvidia Xavier is configured to batch-1 mode and NovuTensor is configured to batch-2 mode to make sure that they can deliver the optimal performance. Column *Baseline* shows the runtime results without any optimizations as Column *Proposed* shows the performance optimized by the combined methods discussed in the previous sections. Our methods decrease the runtime latency for Nvidia Xavier by 50.1% at most. For NovuTensor, our proposed methods achieve 37.5% improvement. NovuTensor is able to process 66 images per second by applying our proposed methods, which outperforms Xavier due to its native batch-2 design. Table 5 demonstrates the overall improvement by applying all proposed optimizations in this paper (i.e., pre-processing optimization and pipelining). The independent benefits of pre-processing optimization can be referred in Figure 3.

Authorized licensed use limited to: Univ of Calif Merced. Downloaded on August 17,2022 at 23:58:13 UTC from IEEE Xplore. Restrictions apply.

Table 5: Overall Improvement on Latency and Throughput under Different Configurations for 3D Object Detection Tasks by Combining Proposed Methods

Devices	Device Power	Precision	Latency (ms)		Throughput (imgs/s)	
			Baseline	*Proposed*	*Baseline*	*Proposed*
Xavier	15W	INT8	49.5	26.3	20.2	38.0
		FP16	57.4	31.4	17.4	31.8
		FP32	73	41.7	13.7	23.9
	30W	INT8	49	25	20.4	40
		FP16	56	29.2	17.9	34.2
		FP32	70	34.9	14.3	28.7
	Max	INT8	31	**20**	32.3	50
		FP16	38.2	23.2	26.2	43.1
		FP32	54	27.2	18.5	36.8
NovuTensor	15W	INT8	48	30	41.6	**66**

5 ACCURACY IMPROVEMENT

This section introduces the accuracy optimizations of Deep Learning inference applications on EdgeAI platforms. Since the accuracy may suffer from quantizing the model weights and activations to lower numerical precisions, as discussed in Section 3.3, it is necessary to eliminate or reduce the accuracy loss caused by the quantization. In this paper, we propose two optimization methodologies to get higher accuracy prediction results.

5.1 Hybrid Calibration

Quantization for INT8 precision demands calibration, introduced in Section 3.3, to determine a mapping function from real numbers to 8-bit integers. In our evaluations, we randomly picked 100 data out of the training set as the calibration data. The calibration process collects the output of each layer by feeding the calibration data into neural network. The scale factors will be determined by the calibration policy. To avoid severe accuracy loss, optimal scale factors are needed to minimize the information loss after quantization, which can be defined as Equation 2. Four calibration methods commonly considered for quantization [45] are listed below:

- *Max:* Select the maximum value as the threshold for each layer during calibration [44].
- *Entropy:* Minimize the loss of information by measuring the amount of information loss based on Kullback-Leiber (KL) divergence [32].
- *Percentile:* Determine a percentile of histogram during calibration and select the value at the percentile as the mapping threshold [30]. For instance, if the percentile is 99%, then 1% largest values will be clipped.
- *Sampling-Brute-Force (SBF):* Determine the optimal threshold by brute-force searching a sampled search space. For example, 100 thresholds within a search space can be selected and the one that causes minimum information loss during calibration will be selected as the threshold. The search space can be 99.9% to 100% with step size of 1×10^{-5}.

Determining the optimal mapping thresholds of each layer in the neural network needs to collect the histogram of activation values for all the calibration data. A threshold will be determined for each layer to be mapped to 127, the max value represented by an 8-bit integer, based on different calibration methods. *Max-based* and *Percentile-based* methods can generate the scale factors in constant time but may not be able to deliver the optimal one.

NovuTensor's SDK calculates the thresholds by the *Percentile-based* method and provides a quantization tool that uses SBF to search an optimal percentile. TensorRT [8] for Nvidia Xavier provides an *Entropy Calibrator* for calibration. However, employing the calibrator directly could cause severe accuracy loss as discussed in Section 3.3. The mAP decreases 46% and 88%, when reducing the numerical precisions from FP32 to INT8, for 3D object detection tasks on NovuTensor and Nvidia Xavier, respectively.

In order to minimize the accuracy loss from quantization, we attempt to use *SBF* method to determine the thresholds in the beginning. The mAP is increased by 20% after brute-force searching the thresholds. But the major drawback of *SBF* policy is that searching a range for each layer is extremely time consuming. Then, we analyze the histograms of activations from each layer in Yolov2 network. It is unexpected that the absolute values of activations from all the layers except the last layer (Layer 1 to Layer 22) are smaller than 127. Hence, we attempt the *Max-based* policy, mapping the maximum absolute value in activations to 127, for determining the thresholds of all the layers. The mAP increases 22% compared with that using *Percentile-based* calibration method. Our experiments in Figure 7 precisely demonstrate the comparison of *SBF* policy and *Max-based* policy. They are able to deliver similar accuracy results because the search space of *SBF* policy includes the maximum absolute value in activations of the network. Meanwhile, *SBF* policy may not select the best thresholds for some layers due to the interference of previous layers. Then the mAP of using *SBF* is slightly lower than of using *Max-based* policy.

To further analyze the *Max-based* method, we evaluate the cumulative accuracy loss from each layer. For instance, to measure the cumulative accuracy loss caused by Layer 1 to Layer 12, the Layer 12's output activations which are calculated in INT8 precision are collected. Then, the collected activations are forwarded into Layer 13 as input, but are calculated in FP32 precision. We observe that the mAP only drops heavily after Layer 21, 22, and 23. Therefore, we propose a hybrid calibration method, which combines *Max-based* method and *Percentile-based* method. Algorithm 1 shows our proposed hybrid calibration procedure. Line 1 obtains the histograms of activations from each convolutional layer and sorts them by ascending order. Assume a neural network has N convolutional layers and the maximum absolute number that can be represented by the numerical precision is T. If the maximum absolute value during calibration is larger than T, an optimal scale factor will be searched as show in Line 6. On the other hand, the scale factor will be set to the maximum absolute value if it is smaller than T. But the program will search the optimal scale factor if the information loss is higher than a predefined threshold E after evaluation (Line 8-13). The *search space* includes the percentiles from 99.9% to 100% with a step of 1×10^{-5}. Hence, 100 percentiles are in the *search space* in total, and the quantization loss can be calculated as Equation 2.

Our hybrid calibration policy increases the mAP by 64% compared with the baseline and only has a loss of 0.08 mAP compared with that computed in the FP32 precision (i.e., no quantization and best case). The accuracy evaluation in this paper is illustrated in Figure 7. Our calibration policy is robust for any randomly selected calibration data from training set, and is not sensitive to the selection of calibration data, as long as there are enough samples to accurately represent the distribution of the data. The inference

Authorized licensed use limited to: Univ of Calif Merced. Downloaded on August 17,2022 at 23:58:13 UTC from IEEE Xplore. Restrictions apply.

Algorithm 1: Hybrid Calibration Policy

Result: Determine the optimal scale factor Δ_i for each convolutional layer $Conv_i$.

// Get the histogram of each convolutional layer

1 Initialize array: $H[0], H[1], H[2], ...H[N-1]$ and sort $abs(H[i])$ by ascending order;
2 **for** $i \leftarrow 0 < N$ **do**
3 max_value $\leftarrow max(abs(H[i]))$;
4 **if** *max_value* $> T$ **then**
5 **for** δ *in Search Space* **do**
6 $\Delta_i \leftarrow \delta$ that minimizes the quantization error;
7 **end**
8 **else if** *max_value* $\leq T$ **then**
9 $\Delta_i \leftarrow$ max_value;
10 Evaluate the accuracy loss e_i;
11 **if** $e_i > E$ **then**
12 Go back to line 5;
13 **end**
14 **end**
15 **end**
16 **return** $\Delta_0, \Delta_1, ...\Delta_{N-1}$;

process will not use the calibration data. Fine tuning the scale factors is a time-consuming process, given that the model is tuned layer by layer. The optimization for Nvidia Xavier is not included in this paper due to space constraints. Also note that TensorRT does not provide the hybrid calibration method. However, our proposed optimization methodology for calibration process can be used for other EdgeAI platforms as well.

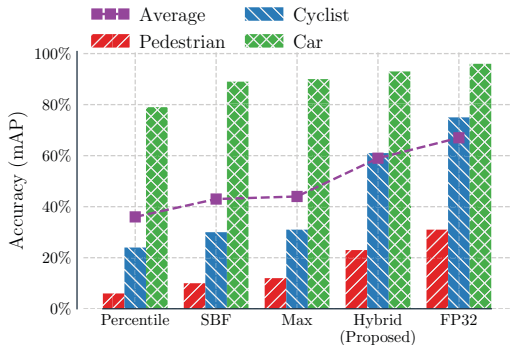


Figure 7: Accuracy Comparison for BEV Object Detection with Different Calibration Methods

Recommendation #4: Eliminating the accuracy loss from quantization and calibration needs careful designs. Analyzing each layer's histograms of a neural network can help to determine the optimal scale factors as achieved by our proposed hybrid calibration policy.

5.2 Higher Input Resolution

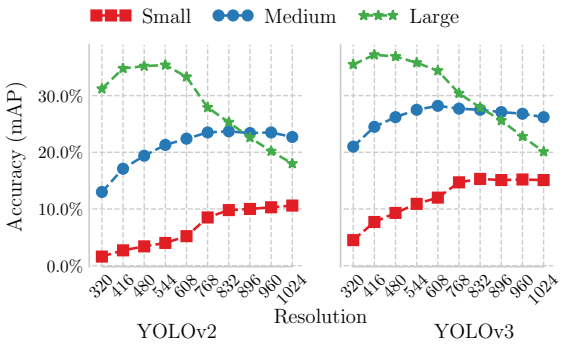
The input data for object detection could be images or points clouds captured by different devices such as cameras, LiDAR, etc. The qualities of the input data will influence the accuracy of prediction results. For example, it would be easier to detect small objects in a high resolution image rather than in a low resolution image. Hence, increasing the resolution of input image should be helpful for improving the prediction accuracy. On the other hand, forwarding larger input data through the neural network requires more computing resources and consumes more energy. In this case, EdgeAI platforms can take advantage of their application specific processors to process the large input data in an energy efficient way.

Experiments are conducted with varied neural networks and resolutions of input images, to understand the correspondence between accuracy and resolutions. YOLOv2 [38] and YOLOv3 [39] are chosen because they are widely used for 2D object detection tasks and Yolov2 can be deployed on most EdgeAI platforms. YOLOv3, an improved version of YOLOv2 which uses three different scales to predict the bounding boxes, performs better on detecting small objects than YOLOv2 [23]. The evaluation results are shown in Figure 8a. The mAPs for small, medium, and large objects increase when the resolutions of input images are increasing. The small objects can be detected more easily by increasing the size of objects. However, the mAP of medium and large objects starts to decrease after reaching the peak because the model does not trained with images out of the training range. Darknet [37], the training framework used for our experiments, will randomly resize the input images to different resolutions within a range (320×320 to 608×608) during training, in order to make the model compatible with input images with various size objects.

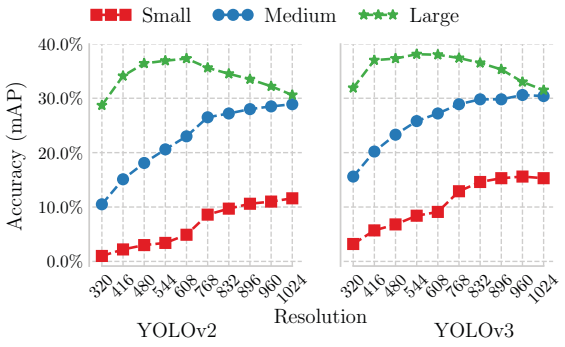
Since the accuracy of medium and large objects are restricted by the default training range, we enlarge the training range to 736×736 to 1024×1024 and retrain models with the same hyper-parameters. The evaluation results with the retrained models are illustrated in Figure 8b. Benefits are gained for detecting medium objects with retraining the models because those resized objects are fitted during training. And huge benefits for small objects can be obtained by increasing the resolutions of input images. Large objects detection accuracy, however, tends to decrease with the increase in input image sizes despite re-training. This is likely attributed to the fixed receptive field for each of the network topologies. The receptive field of the neural network describes the size of the input image region which is non-linearly transformed to a given output value. Here, while we are increasing the input image size for each of the networks, we are keeping the topologies fixed and thus the receptive fields fixed. The networks with large input resolutions thus will have a reduced context from which to make object detection decisions. It is expected that methods to increase the receptive field of these networks, would further improve large object accuracy for large input resolutions.

We also train models on another popular dataset for object detection, PASCAL-VOC [14]. PASCAL-VOC dataset is simpler than MSCOCO with respect to the image size and the number of objects, especially the small objects in a single picture [47]. On average, there exist 7.4 objects to be detected in an image of MSCOCO compared with 2.4 objects of PASCAL-VOC. The evaluation results are

Authorized licensed use limited to: Univ of Calif Merced. Downloaded on August 17,2022 at 23:58:13 UTC from IEEE Xplore. Restrictions apply.



(a) w/o-retraining



(b) w-retraining

Figure 8: Accuracy Comparison on MSCOCO

illustrated in Figure 9. Note that 608-models are trained with the default training range, 1024-models are trained with the enlarged range. More benefits are gained for MSCOCO, which has images that are more complex and contain more objects, compared with PASCAL-VOC, when increasing the input image resolution. Regarding the neural network structures, significant improvements are seen for YOLOv2 with MSCOCO when increasing the resolution of input images. Furthermore, the 1024-model performs on par with 608-model for both YOLOv2 and YOLOv3 on PSACAL-VOC due to its simplicity.

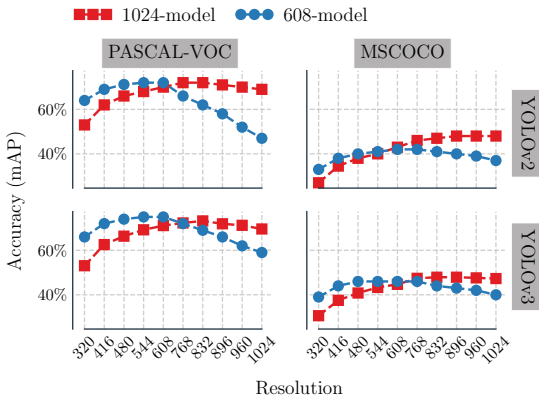


Figure 9: Accuracy Comparison for with Varied Input Resolutions, Neural Network Structures, and Datasets

A surprising observation is that the combination of high resolution input and YOLOv2 network can achieve similar or even better mAP than the combination of low resolution input and YOLOv3 network. For instance, we compare the mAP between YOLOv2 and YOLOv3 with various input sizes as shown in Table 6. To fairly compare, we choose the combinations that have similar number of computing operations. Using YOLOv2 with 1024×1024 input images achieves better mAP than using YOLOv3 at the lower resolution. Hence, deploying simple neural networks on EdgeAI inference systems can also achieve the desired accuracy, with a sacrifice of increased computational complexity.

Table 6: Two Combinations of Various Networks and Input Sizes

Networks	Input Resolution	GFLOPs	mAP
YOLOv2	1024x1024	178.53	0.48
YOLOv3	608x608	140.69	0.46

In conclusion, increasing the resolution of input images is a convincing method to improve the accuracy of object detection tasks according to our experimental results. Nevertheless, there still exist some restrictions as follows: (1) Retraining may be required because the accuracy will suffer if the resolutions of input feature maps are out of the training range. (2) Additional benefits can be obtained by using the model trained in higher resolution images if the dataset contains more complex and small objects. Though these restrictions exist, increasing the resolution of input images, which augments the information of input data, can help Deep Learning accelerators to predict more accurate results for users.

Recommendation #5: Data quality matters a lot for EdgeAI inference applications, which is usually overlooked. In our experiment, increasing the resolutions of input feature maps can improve the accuracy for object detection tasks. This implies that with high-quality data, simpler neural networks can achieve higher or similar accuracy compared with complex neural networks that might not be supported by some low-cost EdgeAI platforms.

6 DISCUSSION

This section discusses power consumption of end-to-end EdgeAI inference systems evaluated in this paper (NovuTensor and Nvidia Xavier). We demonstrate the effect of our proposed methods on the power consumption. The feasibility of extending our methods to other EdgeAI systems will be discussed in Section 6.2.

6.1 Power Consumption

Energy efficiency is another important factor for EdgeAI systems excluding runtime performance and accuracy discussed previously. Inference tasks usually consume much less energy than Deep Learning training tasks because Deep Learning training requires high-end GPUs from datacenters like Tesla V100, which consumes 300 Watts per GPU at peak. We evaluate two types of EdgeAI systems in terms of their connection approaches with the host machine. NovuTensor is a PCIe-based system equipped with Deep Learning accelerators. Therefore, to perform end-to-end inference tasks, the host machine

Authorized licensed use limited to: Univ of Calif Merced. Downloaded on August 17,2022 at 23:58:13 UTC from IEEE Xplore. Restrictions apply.

mainly includes CPU and DRAM needs to participate. Nvidia Xavier is an embedded system on a module, which contains GPU, DL accelerators, CPU, and memory. It performs inference tasks individually without involvement of the host machine.

Due to the different organizations of NovuTensor and Xavier, we need to analyze their power consumptions in different ways. Figure 10 demonstrates the breakdown of power consumption on NovuTensor running 3D object detection application. To monitor the energy usage on the host machine, we use CPU Energy Meter [2] for Intel CPUs, which is able to monitor power consumption of CPU socket and memory during application runtime. CPU Energy Meter will report the duration of program in second (s) and the whole energy usage in joule (J). Then we follow equation $W = \frac{J}{s}$ to obtain the actual power consumption in Watt (W) of host machine. As shown in Figure 10, the host machine consumes 5.6 Watt on idle time. Since NovuTensor does not provide a tool to monitor the actual power consumption, we report 15 Watt from its specification. The end-to-end inference application on NovuTensor consumes ~40 Watt in total. To analyze the effect of our proposed methods on the power consumption, we conduct experiments w/wo our proposed methods with various input resolutions. The experiments show that our proposed design can save up to 20% energy while have slightly higher power consumption. This is mainly because our design can use CPU cores efficiently with higher parallelism as discussed in Section 4. We also find that processing higher resolution images is more power efficient. For Nvidia Xavier, although users are allowed to configure power mode, we cannot monitor the actual power consumption of each component. So we consider 30/15/10 Watt as the actual power consumption.

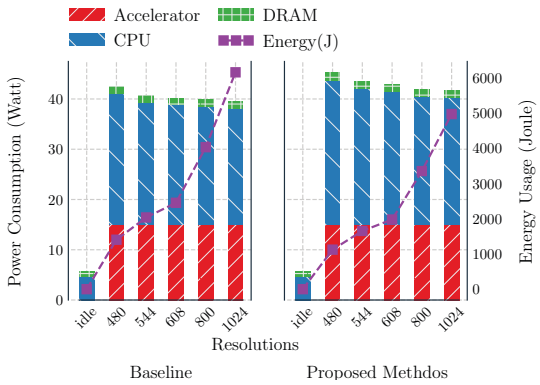


Figure 10: Power and Energy Consumption Comparison of NovuTensor for 3D Object Detection with Different Input Resolutions

6.2 Extend to other EdgeAI Systems

To extend our proposed methods to other EdgeAI systems, two factors should be considered for deployment: 1) Since EdgeAI platforms require various frameworks and provide different SDKs to developers, the programs may need to be rewritten. For example, EdgeTPU uses TensorFlow-Lite [7]. Though programming languages and frameworks are different, runtime performance can still benefit from our proposed methods. 2) The determined scale factors from our proposed calibration method can be reused for other EdgeAI systems if user defined scale factors are allowed by the framework. Otherwise, Algorithm 1 needs to be applied to determine the optimal scale factors.

7 RELATED WORK

Object Detection on EdgeAI Systems. Deep Learning based methods for 2D Object Detection can be grouped into two categories: two-stage detection and one-stage detection. Two-stage detection systems such as RCNN [19], and Fast RCNN [18], separate the localization and classification processes. One-stage detection system only needs one step to classify and locate the objects, which improves the inference runtime performance [28, 38, 39]. The performance of EdgeAI inference systems for 2D object detection workloads is discussed in this paper [21]. To predict 3D bounding boxes, most existing works use LiDAR point clouds as the input of the detector. LiDAR point clouds provide more information of the surrounding environment than 2D images. There are mainly two ways to process LiDAR cloud points. The first method is to directly process the point clouds in 3D [34]. The other way to process LiDAR point clouds is to encode and discretize the point clouds into some fixed grids [13, 31, 41, 46].

Accelerations for AI Inference. In the past decade, researchers and engineers have put much effort into boosting AI performance. The innovations of machine learning and advances in hardware architecture have been discussed by Reagen et al. [36]. A computing platform with powerful GPUs can accelerate the AI training and data processing [20, 24]. The demanding use of computing resources for AI workloads forces the convergence of AI and high-end computing. More research work focus on the Deep Learning inference due to the demanding of edge-side AI applications. Gao et al. study the performance of mobile GPUs for inference [16]. Many processors and accelerators that provide high-performance computation capabilities for Deep Learning inference have been proposed in recent years [3, 12]. Our work is different than these studies, since we focus on proposing guidance for developers to improve the performance of object detection tasks on EdgeAI inference systems.

8 CONCLUSION

In this paper, we identify three limitations of EdgeAI inference systems for object detection applications, which are: 1) less compute capabilities due to low power consumption, 2) restrictions of supported AI inference operations, and 3) accuracy loss due to lower numerical precision. These limitations lead to two sub-optimal problems of EdgeAI inference systems according to our characterizations, which are the bottlenecks of pre-/post-processing and accuracy loss. Motivated by these observations, we propose five different optimization guides for EdgeAI inference systems to achieve better runtime performance and accuracy while maintaining low energy consumption. Utilizing technologies under our guidance can significantly improve the performance of EdgeAI inference systems. In the meantime, we also find that, in order to achieve higher accuracy, we do not necessarily need more complex neural networks, which often require more expensive systems. Instead, feeding higher resolution images to simpler neural networks could achieve desired accuracy. We believe our recommendations can benefit current- and next-generation EdgeAI inference systems for object detection applications as well as other applications.

Authorized licensed use limited to: Univ of Calif Merced. Downloaded on August 17,2022 at 23:58:13 UTC from IEEE Xplore. Restrictions apply.

ACKNOWLEDGMENTS

We want to thank the anonymous reviewers for their insightful comments and suggestions. This work was supported in part by the NSF research grant CCF #1822987 and CCF #2132049.

REFERENCES

[1] 2021. https://ark.intel.com/content/www/us/en/ark/products/97129/intel-core-i7-7700k-processor-8m-cache-up-to-4-50-ghz.html
[2] 2021. CPU Energy Meter. https://github.com/sosy-lab/cpu-energy-meter
[3] 2021. Edge TPU. https://cloud.google.com/edge-tpu
[4] 2021. Intel Math Kernel Library. https://software.intel.com/content/www/us/en/develop/tools/math-kernel-library.html
[5] 2021. NVIDIA Jetson AGX Xavier. https://developer.nvidia.com/embedded/jetson-agx-xavier-developer-kit
[6] 2021. OpenMP. https://www.openmp.org/
[7] 2021. TensorFlow-Lite. https://www.tensorflow.org/lite
[8] 2021. TensorRT. https://developer.nvidia.com/tensorrt
[9] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. 2015. TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems. https://www.tensorflow.org/ Software available from tensorflow.org.
[10] Amer Al-Rahayfeh and Miad Faezipour. 2013. Enhanced frame rate for real-time eye tracking using circular hough transform. In *2013 IEEE Long Island Systems, Applications and Technology Conference (LISAT)*. IEEE, 1–6.
[11] bostondiditeam. 2021. MV3D Implementation. https://github.com/bostondiditeam/MV3D
[12] Yunji Chen, Tianshi Chen, Zhiwei Xu, Ninghui Sun, and Olivier Temam. 2016. DianNao Family: Energy-Efficient Hardware Accelerators for Machine Learning. *Communications of the ACM* 59, 11 (2016), 105–112.
[13] Chen, Xiaozhi and Ma, Huimin and Wan, Ji and Li, Bo and Xia, Tian. 2017. Multi-view 3d object detection network for autonomous driving. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 1907–1915.
[14] Mark Everingham, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. 2010. The Pascal Visual Object Classes (VOC) Challenge. *International Journal of Computer Vision* 88, 2 (2010), 303–338.
[15] Dustin Franklin. 2021. NVIDIA Jetson AGX Xavier Delivers 32 TeraOps for New Era of AI in Robotics. https://developer.nvidia.com/blog/nvidia-jetson-agx-xavier-32-teraops-ai-robotics/
[16] Gao, Cao and Gutierrez, Anthony and Rajan, Madhav and Dreslinski, Ronald G and Mudge, Trevor and Wu, Carole-Jean. 2015. A study of mobile device utilization. In *2015 ieee international symposium on performance analysis of systems and software (ispass)*. IEEE, 225–234.
[17] Andreas Geiger, Philip Lenz, and Raquel Urtasun. 2012. Are we ready for autonomous driving? the kitti vision benchmark suite. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, 3354–3361.
[18] Ross Girshick. 2015. Fast R-cnn. In *Proceedings of the IEEE International Conference on Computer Vision*. 1440–1448.
[19] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. 2014. Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 580–587.
[20] Hazelwood, Kim and Bird, Sarah and Brooks, David and Chintala, Soumith and Diril, Utku and Dzhulgakov, Dmytro and Fawzy, Mohamed and Jia, Bill and Jia, Yangqing and Kalro, Aditya and others. 2018. Applied machine learning at facebook: A datacenter infrastructure perspective. In *2018 IEEE International Symposium on High Performance Computer Architecture (HPCA)*. IEEE, 620–629.
[21] Hui, Yujie and Lien, Jeffrey and Lu, Xiaoyi. 2019. Early Experience in Benchmarking Edge AI Processors with Object Detection Workloads. In *International Symposium on Benchmarking, Measuring and Optimization*. Springer, 32–48.
[22] Jia, Yangqing and Shelhamer, Evan and Donahue, Jeff and Karayev, Sergey and Long, Jonathan and Girshick, Ross and Guadarrama, Sergio and Darrell, Trevor. 2014. Caffe: Convolutional Architecture for Fast Feature Embedding. In *Proceedings of the 22nd ACM International Conference on Multimedia* (Orlando, Florida, USA) *(MM '14)*. Association for Computing Machinery, New York, NY, USA, 675–678. https://doi.org/10.1145/2647868.2654889
[23] Jiao, Licheng and Zhang, Fan and Liu, Fang and Yang, Shuyuan and Li, Lingling and Feng, Zhixi and Qu, Rong. 2019. A survey of deep learning-based object detection. *IEEE Access* 7 (2019), 128837–128868.
[24] Jakub Konečnỳ, Brendan McMahan, and Daniel Ramage. 2015. Federated optimization: Distributed optimization beyond the datacenter. *arXiv preprint arXiv:1511.03575* (2015).
[25] Krizhevsky, Alex and Sutskever, Ilya and Hinton, Geoffrey E. 2017. Imagenet classification with deep convolutional neural networks. *Commun. ACM* 60, 6 (2017), 84–90.
[26] Ku, Jason and Mozifian, Melissa and Lee, Jungwook and Harakeh, Ali and Waslander, Steven L. 2018. Joint 3d proposal generation and object detection from view aggregation. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 1–8.
[27] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. 2014. Microsoft COCO: Common Objects in Context. In *European Conference on Computer Vision*. Springer, 740–755.
[28] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg. 2016. SSD: Single Shot Multibox Detector. In *European Conference on Computer Vision*. Springer, 21–37.
[29] Chien-Ping Lu and Yu-Shuen Tang. [n.d.]. Native Tensor Processor, and Partitioning of Tensor Contractions. https://patentscope.wipo.int/search/en/detail.jsf?docId=US225521272&tab=NATIONALBIBLIO.
[30] McKinstry, Jeffrey L and Esser, Steven K and Appuswamy, Rathinakumar and Bablani, Deepika and Arthur, John V and Yildiz, Izzet B and Modha, Dharmendra S. 2018. Discovering low-precision networks close to full-precision networks for efficient embedded inference. *arXiv preprint arXiv:1809.04191* (2018).
[31] Meyer, Gregory P and Laddha, Ankit and Kee, Eric and Vallespi-Gonzalez, Carlos and Wellington, Carl K. 2019. Lasernet: An efficient probabilistic 3d object detector for autonomous driving. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 12677–12686.
[32] Szymon Migacz. 2021. 8-bit Inference with TensorRT.
[33] Ovtcharov, Kalin and Ruwase, Olatunji and Kim, Joo-Young and Fowers, Jeremy and Strauss, Karin and Chung, Eric S. 2015. Toward accelerating deep learning at scale using specialized hardware in the datacenter. In *2015 IEEE Hot Chips 27 Symposium (HCS)*. IEEE Computer Society, 1–38.
[34] Qi, Charles R and Su, Hao and Mo, Kaichun and Guibas, Leonidas J. 2017. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 652–660.
[35] Qi, Charles Ruizhongtai and Yi, Li and Su, Hao and Guibas, Leonidas J. 2017. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In *Advances in neural information processing systems*. 5099–5108.
[36] Reagen, Brandon and Adolf, Robert and Whatmough, Paul and Wei, Gu-Yeon and Brooks, David. 2017. Deep learning for computer architects. *Synthesis Lectures on Computer Architecture* 12, 4 (2017), 1–123.
[37] Joseph Redmon. 2021. Darknet. https://github.com/pjreddie/darknet
[38] Redmon, Joseph and Farhadi, Ali. 2017. YOLO9000: better, faster, stronger. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 7263–7271.
[39] Redmon, Joseph and Farhadi, Ali. 2018. Yolov3: An incremental improvement. *arXiv preprint arXiv:1804.02767* (2018).
[40] Shin, Sungho and Hwang, Kyuyeon and Sung, Wonyong. 2016. Fixed-point performance analysis of recurrent neural networks. In *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 976–980.
[41] Simony, Martin and Milzy, Stefan and Amendey, Karl and Gross, Horst-Michael. 2018. Complex-yolo: An euler-region-proposal for real-time 3d object detection on point clouds. In *Proceedings of the European Conference on Computer Vision (ECCV)*. 0–0.
[42] Sung, Wonyong and Shin, Sungho and Hwang, Kyuyeon. 2015. Resiliency of deep neural networks under quantization. *arXiv preprint arXiv:1511.06488* (2015).
[43] Sze, Vivienne and Chen, Yu-Hsin and Yang, Tien-Ju and Emer, Joel S. 2017. Efficient processing of deep neural networks: A tutorial and survey. *Proc. IEEE* 105, 12 (2017), 2295–2329.
[44] Vanhoucke, Vincent and Senior, Andrew and Mao, Mark Z. 2011. Improving the speed of neural networks on CPUs. (2011).
[45] Wu, Hao and Judd, Patrick and Zhang, Xiaojie and Isaev, Mikhail and Micikevicius, Paulius. 2020. Integer Quantization for Deep Learning Inference: Principles and Empirical Evaluation. *arXiv preprint arXiv:2004.09602* (2020).
[46] Yang, Bin and Luo, Wenjie and Urtasun, Raquel. 2018. Pixor: Real-time 3d object detection from point clouds. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*. 7652–7660.
[47] Yang, Yang and Deng, Hongmin. 2020. GC-YOLOv3: You Only Look Once with Global Context Block. *Electronics* 9, 8 (2020), 1235.
[48] Zhou, Yin and Tuzel, Oncel. 2018. Voxelnet: End-to-end learning for point cloud based 3d object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 4490–4499.

Authorized licensed use limited to: Univ of Calif Merced. Downloaded on August 17,2022 at 23:58:13 UTC from IEEE Xplore. Restrictions apply.