Scientific Image Restoration Anywhere

Vibhatha Abeykoon*, Zhengchun Liu†, Rajkumar Kettimuthu†, Geoffrey Fox* and Ian Foster†‡
*School of Informatics, Computing and Engineering, IN 47405, USA
{vlabeyko, gcf}@iu.edu
†Data Science and Learning Division, Argonne National Laboratory, Lemont, IL 60439, USA
{zhengchung.liu, kettimut, foster}@anl.gov
‡Department of Computer Science, University of Chicago, Chicago, IL 60637, USA

***Abstract*—The use of deep learning models within scientific experimental facilities frequently requires low-latency inference, so that, for example, quality control operations can be performed while data are being collected. Edge computing devices can be useful in this context, as their low cost and compact form factor permit them to be co-located with the experimental apparatus. Can such devices, with their limited resources, can perform neural network feed-forward computations efficiently and effectively? We explore this question by evaluating the performance and accuracy of a scientific image restoration model, for which both model input and output are images, on edge computing devices. Specifically, we evaluate deployments of TomoGAN, an image-denoising model based on generative adversarial networks developed for low-dose x-ray imaging, on the Google Edge TPU and NVIDIA Jetson. We adapt TomoGAN for edge execution, evaluate model inference performance, and propose methods to address the accuracy drop caused by model quantization. We show that these edge computing devices can deliver accuracy comparable to that of a full-fledged CPU or GPU model, at speeds that are more than adequate for use in the intended deployments, denoising a 1024×1024 image in less than a second. Our experiments also show that the Edge TPU models can provide 3× faster inference response than a CPU-based model and 1.5× faster than an edge GPU-based model. This combination of high speed and low cost permits image restoration anywhere.**

***Index Terms*—Edge computing, Deep learning, Image restoration, Model quantization**

I. Introduction

Deep neural networks show considerable promise for the rapid analysis of data collected at scientific experiments, enabling tasks such as anomaly detection [1], image enhancement [2], and image reconstruction [3] to be performed in a fraction of the time required by conventional methods. However, the substantial computational costs of deep models are an obstacle to their widespread adoption. The graphical processing unit (GPU) devices that are typically used to run these models are too expensive to be dedicated to individual instruments, while dispatching analysis tasks to shared data centers can require substantial data movement and incur large round trip latencies.

Specialized "edge" inference devices [4] such as the NVIDIA Jetson Tx2 GPU (henceforth `TX2`) [5] and Google Edge TPU [6] are potential solutions to this problem. (The Edge TPU is distributed by Coral Inc. in two configurations: `Accelerator`, which relies on a host machine, such as a PC or single-board Raspberry Pi, and `Dev Board`, which comes with a 64-bit ARM system as host.) These edge devices use techniques such as reduced precision arithmetic to enable rapid execution of deep models with a low price point (and low power consumption and compact form factor) that makes it feasible to embed them within scientific instruments.

The question remains, however, as to whether these edge inference devices can execute the deep models used in science with sufficient speed and accuracy. Models originally developed to run on GPUs that support 32-bit floating point arithmetic must be adapted to run on edge devices that may support only lower-precision integer arithmetic, a process that typically employs a technique called model quantization [7–9]. Google and NVIDIA have developed implementations of such schemes, allowing inference with integer-only arithmetic on integer-only hardware [10–12]. They report benchmark results showing that edge devices can perform inference as rapidly as a powerful PC, at much lower cost [13, 14]. However, questions remain when it comes to using such devices in scientific settings. The resulting models will be more compact, but will they be sufficiently accurate? And will the edge device run the models rapidly enough to meet scientific goals?

We explore these questions here by studying how a specific scientific deep learning model, TomoGAN[1] [15, 16], can be adapted for edge deployment. TomoGAN uses generative adversarial network (GAN) methods [17] to enhance the quality of low-dose X-ray images via a denoising process. Although diverse object detection and classification applications have been implemented on edge devices, image restoration with a complex image generative model has not previously been attempted on them. We adapt TomoGAN to run on the Google Edge TPU (both `Accelerator` and `Dev Board`) and `TX2`, and compare the accuracy and computational performance of the resulting models with those of other implementations. We also describe how to mitigate accuracy loss in quantized models by applying a lightweight "fine-tuning" convolutional neural network to the results of the quantized TomoGAN.

The rest of this paper is as follows. In §II, we describe how we adapt a pre-trained deep learning model, TomoGAN, for the Edge TPU. Next in §III, we present experiments used to evaluate edge computing performance and model accuracy, both with and without the fine-tuning component. In §IV we review related work, and in §V we summarize our results and outline directions for future work.

[1]Code available at git@github.com:ramsesproject/TomoGAN.git

II. Methodology

We next describe how we adapt the TomoGAN model for the `Edge-TPU` (i.e., `Accelerator` and `Dev Board`). Specifically, we describe the steps taken to improve the accuracy of the enhanced images, the datasets used, and our performance and accuracy evaluations.

A. Quantization

We consider two approaches to quantizing the TomoGAN model for the Edge TPU: `Post-Quantization` and `Quantization-Aware`. In both methods, the first step is to design a non-quantized model with the expected features unique to both `Post-Quantization` and `Quantization-Aware` models.

1) Post-Quantization-Based Inference Model: The steps followed to generate the `Post-Quantization`-based inference model are shown in Figure 1.

We first train a `Post-Quantization`-based model, which differs from the standard TomoGAN model only in the input tensor shape, which is 64×64×3 rather than 1024×1024×3. The partitioning of each 1024×1024 input image into multiple 64×64 subimages is needed because of limitations on the output size permitted by the `Edge-TPU-Compatible`. See §III-A for details on training data. The average training time for 40,000 iterations was around 24 hours on a single NVIDIA V100 GPU.

The resulting trained `Post-Quantization` model is then converted to a `Mobile-Compatible` model (see §II-B1), which is used in term to generate an `Edge-TPU-Compatible` model (see §II-B2).

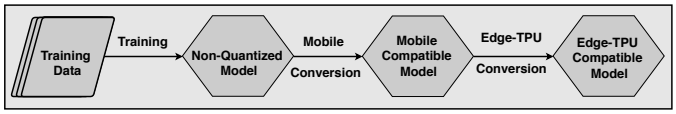

Fig. 1. The `Post-Quantization` model generation workflow involves a training step, conversion of the trained model to a mobile form, and then conversion of the mobile form to an Edge TPU model.

2) Quantization-Aware Based Inference Model: This second approach (§2) differs from the first only in the method used to generate the trained model. In order to attenuate the accuracy loss that may result from the quantization of trained weights in the inference stage, a more complex model is trained that induces fake quantization layers to simulate the effect of quantization.

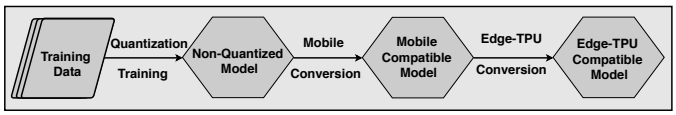

Fig. 2. The `Quantization-Aware` model generation workflow differs from the `Post-Quantization` workflow only in its training step.

The major drawback of this methodology is that the introduction of fake quantization layers leads to a much longer training time, extending to dozens of days. We thus conclude that this method is not feasible for larger models like TomoGAN, and adopt the `Post-Quantization` approach for our TomoGAN-based image restoration system.

B. Model Generation

There are specific model generation schemes to provide mobile-computing and edge-computing friendly models.

1) Mobile-Compatible Model Generation: A mobile-compatible model accepts quantized `unsigned int8` inputs and generates quantized `unsigned int8` outputs. A quantized *int8_value* representation is related to the corresponding *real_value* as follows

$$real\_value = (int8\_value - zero\_point) * scale,$$

where *zero-point* and *scale* are parameters. Prior to mobile-compatible model generation, we process a representative dataset to estimate the value range of the data that are to be quantized, and choose appropriate values for these parameters.

2) Edge-TPU Compatible Model Generation: In order to exploit the `Accelerator` and `Dev Board`, the quantized model must be converted into an `Edge-TPU-Compatible` model by compiling it with the Edge TPU runtime. `Edge-TPU-Compatible` model generation is done by using a compiler deployed with `Edge-TPU` firmware libraries. This compiler enables a conversion of a `Mobile-Compatible` model into an `Edge-TPU-Compatible` model.

C. Inference Workflow

We describe in turn the inference workflows used when running on a CPU, `Edge-TPU`, and Edge GPU. The CPU related experiments are carried out with the trained model with no quantization and the `Edge-TPU` and Edge-GPU experiments are carried out with the trained model with quantization.

1) CPU Inference: The CPU inference workflow, shown in Figure 3, uses the non-quantized model. We feed the required inputs, non-quantized model, and noisy image of size 1×1024×1024×3 to the CPU-based inference API, which returns a de-noised image with dimension 1024×1024.

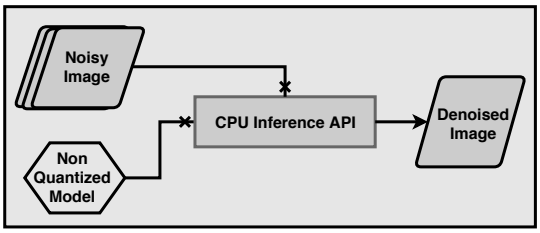

Fig. 3. The CPU inference workflow applies the non-quantized TomoGAN direclty, with no pre- or post-processing required.

2) Edge TPU Inference: The Edge TPU inference workflow, shown in Figure 4, applies the `Post-Quantization` (II-A1) or `Quantization-Aware` (II-A2) models to pre-processed images. We use customized versions of the `Accelerator` and `Accelerator BasicEngine` inference API and `TensorflowLite` API [12] for this purpose.

Each input image has shape $1\times1024\times1024\times3$, where the dimension 3 results from grouping with each image two adjacent images, as used in TomoGAN to improve output quality. Each image is partitioned into 256 *subimages* of shape $1\times64\times64\times3$, due to `Edge-TPU-Compatible` restrictions; after inference, processed subimages are buffered in memory, and once all have been processed, are stitched back together to form the de-noised output image.

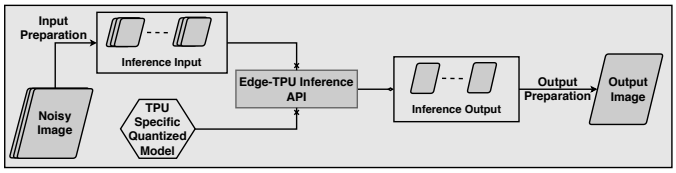

Fig. 4. The Edge TPU inference workflow partitions each input image into 256 subimages, processes the subimages, and stitches the corresponding output subimages together to create the output image.

3) Edge-GPU Based Inference Workflow: The Edge GPU inference workflow, shown in Figure 5, is similar to the CPU inference workflow, except for the part of using a GPU specific quantized model. Each input image, with shape $1\times1024\times1024\times3$, is passed to the Edge GPU-specific quantized model (produced with TensorRT [18] from the non-quantized model), which produces a de-noised image with shape 1024×1024 as output.

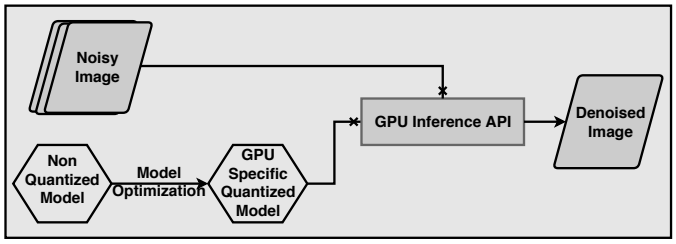

Fig. 5. The Edge GPU inference workflow differs from the CPU inference workflow in its application of model optimization to the non-quantized model.

D. Fine-Tuning Workflow

With `Post-Quantization`-enabled inference, some accuracy may be lost due to model quantization. We observed this effect in our preliminary results: the non-quantized model produced better output than the `Post-Quantization` Edge-TPU quantized model. To improve image quality in this case, we designed a shallow convolutional neural network (referred to as the *Fine-Tune network* in the rest of the paper) to be applied to the output of the quantized TomoGAN: see Figure 6. We use output from the `Edge-TPU-Compatible` model (see §II-C3 and §III-A) to train this network. The target labels are the corresponding target images for each inferred image from the mentioned portion of the training dataset. At the inference stage, we applied the `Edge-TPU` inference workflow (see §II-C3) and used its output as input to the Fine-Tune model. We shall see in §III-C that this Fine-Tune network improves image quality to match that of the images generated from the CPU inference workflow

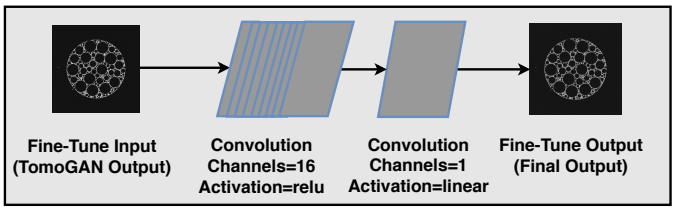

Fig. 6. The Fine-Tune network is applied to the output from the quantized TomoGAN to improve image quality.

III. Experiments

In order to evaluate both computing throughput and model inference performance, we conducted a set of experiments on quantized inference and enhanced the output from direct inference with a shallow Fine-Tune network. We compared the performance and evaluated the image quality on CPU, GPU, and TPU devices individually.

A. Datasets

We used two datasets for our experiments. Each dataset comprises 1024 pairs of 1024×1024 images, each pair being a noisy image and a corresponding ground-truth image, as described in Liu et al. [15]. Ground truth images are obtained from normal-dose X-ray imaging and noisy images from low-dose X-ray imaging of the same sample. We used one dataset for training and the other for testing.

B. Performance Evaluation

We evaluated both inference performance (i.e., throughput) on different hardware platforms and the quality of the resulting images. For inference performance, we studied a laptop CPU (§III-B1), the `Accelerator` and `Dev Board` Edge TPU (§III-B2), and the `TX2` Edge GPU (§III-B3), applying for each the workflow of §II-C to a series of images and calculating the average inference latency.

1) CPU Inference Performance Evaluation: Standard CPU-based experiments were conducted by using the non-quantized model with a personal computer comprising an Intel Core i7-6700HQ CPU@2.60GHz with 32GB RAM. The supported operating system was Ubuntu 16.04 LTS distribution. The non-quantized model takes an average inference time of 1.537 seconds per image: see `i7@2.6GHz` in Figure 7.

2) TPU Inference Performance Evaluation: We evaluated Edge TPU performance on two platforms with different configurations: the `Accelerator` with an Edge TPU coprocessor connected to the host machine (a laptop with Intel i7 CPU) via a USB 3.0 Type-C (data and power) interface, and the `Dev Board` with Edge TPU coprocessor and a 64-bit ARM CPU as host. Columns `Accelerator` and `Dev Board` of Table I provide timing breakdowns for these two devices. The first component is the time to run the quantized TomoGAN model: 0.435 and 0.512 seconds per image for `Accelerator` and `Dev Board`, respectively. The second component, "Stitching," is due to an input image size limit of 64×64 imposed by the Edge TPU hardware and compiler

that we used in this work. Processing a single 1024×1024 image thus requires processing 256 individual 64×64 images, which must then be stitched together to form the complete output image. This stitching operation takes an average of 0.12 and 0.049 seconds per image on the `Dev Board` and `Accelerator`, respectively.

The third component, "Fine-Tune," is the quantized fine-tune network used to improve image quality to match that of the non-quantized model, as discussed in §III-C; this takes an average of 0.070 and 0.166 seconds per image on `Accelerator` and `Dev Board`, respectively. We note that model compilation limitations associated with the current Edge TPU hardware and software require us to run the quantized TomoGAN and Fine-Tune networks separately, which adds extra latency for data movement between host memory and Edge TPU. We expect to avoid this extra cost in the future by chaining TomoGAN and Fine-Tune to execute as one model. (While the quantized TomoGAN requires 301 billion operations to process a 1024×1024 image, Fine-Tune takes only 621 million: a negligible 0.2% of TomoGAN.)

TABLE I
PERFORMANCE BREAKDOWN ON INFERENCE IN EDGE DEVICES. TOPS REFERS TRILLION OPERATIONS PER SECOND AND TFLOPS DENOTES TRILLION FLOATING POINT OPERATIONS PER SECOND.

	Accelerator	**Dev Board**	**Jetson Tx2**
Quantized TomoGAN (s)	**0.435**	0.512	0.880
Stitching (s)	**0.049**	0.120	-
Fine-Tune (s)	**0.070**	0.166	-
Total (s)	**0.554**	0.798	0.880
Power Consumption (w)	2	2	7.5
Peak Performance	4 TOPS	4 TOPS	1.3 TFLOPS

3) Edge GPU Inference Performance Evaluation: The original TomoGAN can process a 1024×1024 pixel image in just 44ms on a NVIDIA V100 GPU card. As our focus here is on edge devices, we evaluated TomoGAN performance on the `TX2`, which has a GPU and is designed for edge computing. Column Jetson Tx2 in Table I shows results. We see an average inference time per image of 0.88 seconds for the `TX2`. We compare in Figure 7 this time with the quantized TomoGAN Edge TPU times (not including stitching and fine tuning).

We note that in constructing the model for the `TX2`, we used NVIDIA's TensorRT toolkit [11, 18] to optimize the operations of TomoGAN. We also experimented with 16-bit floating point, 32-bit floating-point, and unsigned int8, and observed similar performance for each, which we attribute to the lack of Tensor cores in the `TX2`'s NVIDIA Pascal architecture for accelerating multi-precision operations.

4) Performance Discussion: We find that inference is significantly faster on the `Edge-TPU` than on the CPU or `TX2`, and faster on `TX2` than on the CPU. `Accelerator` is faster than `Dev Board`, because the former has a more powerful host (the laptop with i7) with better memory throughput than the latter (the 64-bit SoC ARM platform). These performance differences may appear small, but we should remember that a single light source experiment can generate thousands of images, each larger than 1024×1024, e.g., 2560×2560, and thus any acceleration is valuable.

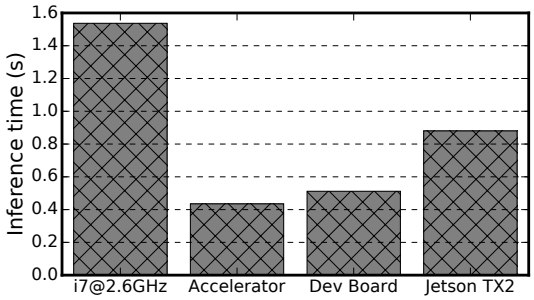

Fig. 7. Inference time for a 1024×1024 pixel image on a CPU, the two Edge TPU platforms, and `TX2`.

C. Image Quality Evaluation

We used structural similarity index (SSIM) [19] to evaluate image quality. We calculated this metric for images enhanced from the original TomoGAN, the quantized TomoGAN, and the quantized TomoGAN plus Fine-Tune network, with results shown in Figure 8. We observe that SSIM for the quantized TomoGAN+Fine-Tune is comparable to that of the original (non-quantized) TomoGAN.

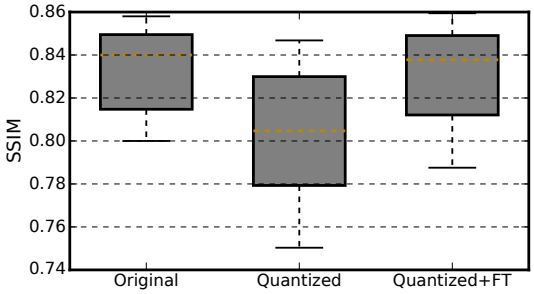

Fig. 8. SSIM image quality scores for original TomoGAN, quantized TomoGAN, and quantized TomoGAN+Fine-Tune, each applied to our 1024 test images. (The boxes show 25th to 75th percentile values, and the dashed lines indicate the mean.)

IV. RELATED WORK

The opportunities and challenges of edge computing have received much attention [20, 21]. Methods have been proposed for both co-locating computing resources with sensors, and for offloading computing from mobile devices to nearby edge computers [22–24].

Increasingly, researchers want to run deep neural networks on edge devices [25–27], leading to the need to adapt computationally expensive deep networks for resource-constrained environments. Quantization, as discussed above, is one such approach [10, 28–30]. Others include the use of neuromorphic hardware [31], specialized software [32], the distribution of deep networks over cloud, local computers, and edge computers [33], and mixed precision computations [34].

Various deep networks have been developed or adapted for edge devices, including Mobilenet [35], VGG [36], and

Resnet [37]. However, that work focuses on image classification and object detection. In contrast, we are concerned with image translation and image-to-image mapping to provide an enhanced image. Also, we are applying our image restoration model on edge devices, an approach that has not been discussed in the literature.

Our use of a fine-tuning network to improve image quality is an important part of our solution, allowing us to avoid the excessive training time required for the quantization-aware model. We are not aware of prior work that has used such a fine-tuning network, although it is conceptually similar to the use of gradient boosting in ensemble learning [38].

V. Conclusion

We have reported on the adaption for edge execution of TomoGAN, an image-denoising model based on generative adversarial networks developed for low-dose x-ray imaging. We ported TomoGAN to the Google Coral Edge TPU devices (`Dev Board` and `Accelerator`) and NVIDIA Jetson `TX2` Edge GPU. Adapting TomoGAN for the Edge TPU requires quantization. We mitigate the resulting loss in image quality, as measured via the SSIM image quality metric, by applying a fine-tune step after inference, with negligible computing overhead. We find that `Dev Board` and `Accelerator` provide $3\times$ faster inference than a CPU, and that `Accelerator` is $1.5\times$ faster than `TX2`. We conclude that edge devices can provide fast response at low cost, enabling *scientific image restoration anywhere*.

The work reported here focused on image restoration. However, before images can be enhanced with TomoGAN, they must be reconstructed from the x-ray images, for example by using filtered back projection (FBP) [39]. FBP is not computationally intensive: processing the images considered here using the TomoPy implementation [40] takes about 400ms per image on a laptop with an Intel i7 CPU. Nevertheless, for a complete edge solution, we should also run FBP on the edge device. We will tackle that task in future work.

Acknowledgment

This work was supported in part by the U.S. Department of Energy, Office of Science, Advanced Scientific Computing Research, under Contract DE-AC02-06CH11357. This research was accomplished when V. Abeykoon is an intern at Argonne National Laboratory under the supervision of Z. Liu. We thank the JLSE management team at the Argonne Leadership Computing Facility and the Google Coral team for their assistance.

References

[1] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly detection: A survey," *ACM Computing Surveys*, vol. 41, no. 3, p. 15, 2009.

[2] Q. Yang, P. Yan, Y. Zhang, H. Yu, Y. Shi, X. Mou, M. K. Kalra, Y. Zhang, L. Sun, and G. Wang, "Low-dose CT image denoising using a generative adversarial network with Wasserstein distance and perceptual loss," *IEEE Transactions on Medical Imaging*, vol. 37, no. 6, pp. 1348–1357, 2018.

[3] J. Schlemper, J. Caballero, J. V. Hajnal, A. N. Price, and D. Rueckert, "A deep cascade of convolutional neural networks for dynamic MR image reconstruction," *IEEE Transactions on Medical Imaging*, vol. 37, no. 2, pp. 491–503, 2017.

[4] X. Xu, Y. Ding, S. X. Hu, M. Niemier, J. Cong, Y. Hu, and Y. Shi, "Scaling for edge inference of deep neural networks," *Nature Electronics*, vol. 1, no. 4, p. 216, 2018.

[5] (2019, Sept.) Nvidia Jetson Tx2. [Online]. Available: https://www.nvidia.com/en-us/autonomous-machines/embedded-systems/jetson-tx2/

[6] "Google Coral," July 2019. [Online]. Available: https://coral.withgoogle.com/

[7] E. Park, S. Yoo, and P. Vajda, "Value-aware quantization for training and inference of neural networks," in *European Conference on Computer Vision*, 2018, pp. 580–595.

[8] L. Hou and J. T. Kwok, "Loss-aware weight quantization of deep networks," *arXiv preprint arXiv:1802.08635*, 2018.

[9] R. Krishnamoorthi, "Quantizing deep convolutional networks for efficient inference: A whitepaper," *arXiv preprint arXiv:1806.08342*, 2018.

[10] B. Jacob, S. Kligys, B. Chen, M. Zhu, M. Tang, A. Howard, H. Adam, and D. Kalenichenko, "Quantization and training of neural networks for efficient integer-arithmetic-only inference," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 2704–2713.

[11] S. Migacz, "Nvidia 8-bit inference width TensorRT," in *GPU Technology Conference*, 2017.

[12] "TensorFlowLite for mobile based deep learning," Sept. 2019. [Online]. Available: https://www.tensorflow.org/lite

[13] "Edge TPU benchmark," Sept. 2019. [Online]. Available: https://coral.withgoogle.com/docs/edgetpu/benchmarks/

[14] "NVIDIA benchmarks on deep learning," Sept. 2019. [Online]. Available: https://developer.nvidia.com/deep-learning-performance-training-inference

[15] Z. Liu, T. Bicer, R. Kettimuthu, D. Gursoy, F. De Carlo, and I. Foster, "TomoGAN: Low-dose x-ray tomography with generative adversarial networks," *arXiv preprint arXiv:1902.07582*, 2019.

[16] Z. Liu, T. Bicer, R. Kettimuthu, and I. Foster, "Deep learning accelerated light source experiments," in *Proceedings of the IEEE/ACM Third Workshop on Deep Learning on Supercomputers (DLS)*, 2019.

[17] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Advances in Neural Information Processing systems*, 2014, pp. 2672–2680.

[18] "TensorRT," July 2019. [Online]. Available: https://docs.nvidia.com/deeplearning/sdk/tensorrt-developer-guide/index.html

[19] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, "Image quality assessment: From error visibility to structural similarity," *IEEE Transactions on Image Processing*, vol. 13, no. 4, pp. 600–612, April 2004.

[20] P. Garcia Lopez, A. Montresor, D. Epema, A. Datta, T. Higashino, A. Iamnitchi, M. Barcellos, P. Felber, and E. Riviere, "Edge-centric computing: Vision and challenges," *ACM SIGCOMM Computer Communication Review*, vol. 45, no. 5, pp. 37–42, 2015.

[21] M. Satyanarayanan, "The emergence of edge computing," *Computer*, vol. 50, no. 1, pp. 30–39, 2017.

[22] X. Chen, Q. Shi, L. Yang, and J. Xu, "ThriftyEdge: Resource-efficient edge computing for intelligent IoT applications," *IEEE Network*, vol. 32, no. 1, pp. 61–65, 2018.

[23] M. Chen and Y. Hao, "Task offloading for mobile edge computing in software defined ultra-dense network," *Journal on Selected Areas in Communications*, vol. 36, no. 3, pp. 587–597, 2018.

[24] X. Chen, L. Jiao, W. Li, and X. Fu, "Efficient multi-user computation offloading for mobile-edge cloud computing," *IEEE/ACM Transactions on Networking*, vol. 24, no. 5, pp. 2795–2808, 2015.

[25] J. Chen and X. Ran, "Deep learning with edge computing: A review," *Proceedings of the IEEE*, vol. 107, no. 8, pp. 1655–1674, 2019.

[26] Z. Zhou, X. Chen, E. Li, L. Zeng, K. Luo, and J. Zhang, "Edge intelligence: Paving the last mile of artificial intelligence with edge computing," *arXiv preprint arXiv:1905.10083*, 2019.

[27] N. D. Lane and P. Georgiev, "Can deep learning revolutionize mobile sensing?" in *16th International Workshop on Mobile Computing Systems and Applications*. ACM, 2015, pp. 117–122.

[28] I. Hubara, M. Courbariaux, D. Soudry, R. El-Yaniv, and Y. Bengio, "Quantized neural networks: Training neural networks with low precision weights and activations," *Journal of Machine Learning Research*, vol. 18, no. 1, pp. 6869–6898, 2017.

[29] M. Kim and P. Smaragdis, "Bitwise neural networks," *arXiv preprint arXiv:1601.06071*, 2016.
[30] I. Hubara, M. Courbariaux, D. Soudry, R. El-Yaniv, and Y. Bengio, "Binarized neural networks," in *Advances in Neural Information Processing Systems*, 2016, pp. 4107–4115.
[31] O. Krestinskaya, A. P. James, and L. O. Chua, "Neuromemristive circuits for edge computing: A review," *IEEE Transactions on Neural Networks and Learning Systems*, 2019.
[32] N. D. Lane, S. Bhattacharya, P. Georgiev, C. Forlivesi, L. Jiao, L. Qendro, and F. Kawsar, "DeepX: A software accelerator for low-power deep learning inference on mobile devices," in *15th International Conference on Information Processing in Sensor Networks*. IEEE Press, 2016, p. 23.
[33] S. Teerapittayanon, B. McDanel, and H.-T. Kung, "Distributed deep neural networks over the cloud, the edge and end devices," in *37th International Conference on Distributed Computing Systems*. IEEE, 2017, pp. 328–339.
[34] I. Chakraborty, D. Roy, A. Ankit, and K. Roy, "Efficient hybrid network architectures for extremely quantized neural networks enabling intelligence at the edge," *CoRR*, vol. abs/1902.00460, 2019. [Online]. Available: http://arxiv.org/abs/1902.00460
[35] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," *arXiv preprint arXiv:1704.04861*, 2017.
[36] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.
[37] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.
[38] L. Breiman, "Arcing the edge," Technical Report 486, Statistics Department, University of California Berkeley, Tech. Rep., 1997.
[39] A. C. Kak, M. Slaney, and G. Wang, "Principles of computerized tomographic imaging," *Medical Physics*, vol. 29, no. 1, pp. 107–107, 2002.
[40] D. Gürsoy, F. De Carlo, X. Xiao, and C. Jacobsen, "TomoPy: A framework for the analysis of synchrotron tomographic data," *Journal of synchrotron radiation*, vol. 21, no. 5, pp. 1188–1193, 2014.

LICENSE