


# Accelerating Deep Learning Model Training on Cloud Tensor Processing Unit

Cristiano A. Künas<sup>1</sup><sup>a</sup>, Edson L. Padoin<sup>2</sup><sup>b</sup> and Philippe O. A. Navaux<sup>1</sup><sup>c</sup>

<sup>1</sup>*Informatics Institute, Federal University of Rio Grande do Sul, Porto Alegre, Brazil*

<sup>2</sup>*Regional University of Northwestern Rio Grande do Sul, Ijuí, Brazil*

**Keywords:** Cloud Computing, TPU, High-Performance Computing, Diabetic Retinopathy.

**Abstract:** Deep learning techniques have grown rapidly in recent years due to their success in image classification, speech recognition, and natural language understanding. These techniques have the potential to solve complex problems and are being applied in various fields, such as agriculture, medicine, and administration. However, training large and complex models requires high-performance computational platforms, making accelerator hardware an essential tool and driving up its cost. An alternative solution is to use cloud computing, where users only pay for usage and have access to a wide range of computing resources and services. In this paper, we adapt a Diabetic Retinopathy neural network model for TPU-based training in the cloud and observe promising results, including reduced training time without code optimization. This demonstrates the potential of cloud computing in reducing the burden on local systems that are often overwhelmed by multiple running applications. This allows for training larger and more advanced models at a lower cost than local computational centers.

## 1 INTRODUCTION


Machine learning techniques have grown rapidly following their success in image classification. Currently, models for speech recognition and natural language understanding improve the functionality of smartphones, while autonomous vehicles are being tested and robotic consultants work in the financial market (Hatcher and Yu, 2018; Abiodun et al., 2018). These techniques have the potential to solve complex problems and are being applied in different areas, such as agriculture, medicine, forensic science, theoretical and applied physics, administration, and management.


Studies in machine learning and deep neural networks indicate a link between the size/complexity of models and their ability to generalize, learn and obtain efficient results in complex tasks (Amodei et al., 2016; Hestness et al., 2019; Sun et al., 2017). Hence, there is an increasing demand for ever bigger deep learning models with more parameters, trained with more data, and high-resolution data.


The training of increasingly large and complex models is making high-performance computational platforms an important tool. As a result, the hardware accelerator has become essential for speeding up the training of deep learning models because it is one of the tasks that consume the most computational resources and can take several days to complete without proper hardware support. However, the growing demand for this type of hardware is increasing its price, making it unfeasible for many researchers.

Cloud Computing is an alternative to high hardware costs for training deep learning models. With it, you only pay for usage (Roloff, 2013), and have access to a wide range of computing resources and services. In addition, cloud providers regularly update their resources, including GPUs and TPUs, which allows for training larger and more advanced models at a lower cost. This type of benefit is difficult to obtain with local computational centers since this equipment is costly, and the renewal of the computational center in small companies and universities does not occur with the frequency that new versions of this equipment are released.

In this context, we suggest offloading the model training to the cloud, reducing the burden on systems often overwhelmed by multiple running appli-

<sup>a</sup> <https://orcid.org/0000-0003-1080-7230>

<sup>b</sup> <https://orcid.org/0000-0002-4015-5619>

<sup>c</sup> <https://orcid.org/0000-0002-9957-5861>

cations<sup>1</sup>. As a case for evaluation, we adapted a Diabetic Retinopathy neural network model for TPU-based training and observed promising results, including reduced training time even without code optimization.

The rest of this paper is organized as follows. Section 2 covers existing work on cloud-based training of large neural networks using TPUs and TPU Pods. Section 3 provides an overview of the TPUv3 architecture used in the experiments and explains the topic of Diabetic Retinopathy (DR), our use case. In Section 4, we describe the methodology, detail the application, and outline the hardware and software setup used. The results of the performance evaluation are presented in Section 5. Finally, Section 6 concludes the paper and outlines future work.

## 2 RELATED WORK

You et al. investigate supercomputers' capability of speeding up DNN training (You et al., 2019b). The approach is to use a large batch size powered by the Layerwise Adaptive Rate Scaling (LARS) algorithm for efficient usage of massive computing resources. They empirically evaluate the effectiveness on five neural networks: AlexNet, AlexNet-BN, GNMT, ResNet-50, and ResNet-50-v2 trained with large datasets while preserving the state-of-the-art test accuracy. Using 2,048 Intel Xeon Phi 7250 Processors, they reduced the 90-epoch ResNet-50 training time from hours to 20 minutes. They implemented an approach on Google's cloud Tensor Processing Unit (TPU) platform, which verifies your previous success on CPUs and GPUs (You et al., 2018). They scaled the batch size of ResNet-50-v2 to 32K and achieved 76.3 percent accuracy. They applied the approach to Google's Neural Machine Translation (GNMT) application, which helps to achieve a 4x speedup on the cloud TPUs.

Wongpanich et al. explore techniques to scale up the training of EfficientNets on TPU-v3 Pods with 2048 cores, motivated by speedups that can be achieved when training at such scales (Wongpanich et al., 2021). Currently, EfficientNets can take on the order of days to train. EfficientNets are a family of state-of-the-art image classification models based on efficiently scaled convolutional neural networks. They discuss optimizations required to scale training to a batch size of 65536 on 1024 TPU-v3 cores, such as selecting large batch optimizers and learning rate schedules and utilizing distributed evaluation and

batch normalization techniques. Additionally, they presented timing and performance benchmarks for EfficientNet models trained on the ImageNet dataset to analyze the behavior of EfficientNets at scale. With the optimizations, they could train EfficientNet on ImageNet to an accuracy of 83% in 1 hour and 4 minutes.

Deep learning is computationally intensive, and hardware vendors have responded by building faster accelerators in large clusters. Training deep learning models requires overcoming both algorithmic and systems software challenges. Ying et al., discuss three systems-related optimizations: (1) distributed batch normalization to control per-replica batch sizes, (2) input pipeline optimizations to sustain model throughput, and (3) 2-D torus all-reduce to speed up gradient summation (Ying et al., 2018). They combined these optimizations to train ResNet-50 on ImageNet to 76.3% accuracy in 2.2 minutes on a 1024-chip TPU v3 Pod with a training throughput of over 1.05 million images/second and no accuracy drop.

The paper of Jouppi et al. evaluates a Tensor Processing Unit (TPU) (Jouppi et al., 2017). They compare the TPU to a server-class Intel Haswell CPU and an Nvidia K80 GPU. The workload, written in the high-level TensorFlow framework, uses production neural networks (NN) applications (MLPs, CNNs, and LSTMs) that represent 95% of NN inference demand. Despite low utilization for some applications, the TPU is, on average, about 15X to 30X faster than the GPU or CPU.

There is an industry-wide trend toward hardware specialization to improve performance, principally deep learning models which are compute-intensive. To systematically benchmark deep learning platforms, Wang et al. introduce ParaDnn, a benchmark suite for deep learning that generates models for fully connected (FC), convolutional (CNN), and recurrent (RNN) neural networks (Wang et al., 2019a). Along with six real-world models, they benchmarked Google's Cloud TPU v2/v3, NVIDIA's V100 GPU, and an Intel Skylake CPU platform. They deeply dive into TPU architecture, reveal its bottlenecks, and highlight valuable lessons learned for future specialized system design. They also provide a thorough comparison of the platforms and find that each has unique strengths for some types of models.

You et al. studied a principled layerwise adaptation strategy to accelerate the training of deep neural networks using large mini-batches (You et al., 2019a). Using this strategy, they developed a new layerwise adaptive large batch optimization technique called LAMB. The empirical results demonstrate the superior performance of LAMB across various tasks, such as BERT and ResNet-50 training, with very lit-

<sup>1</sup>Projects in progress on the SDumont Supercomputer: [https://sdumont.lncc.br/projects\\_statistics.php](https://sdumont.lncc.br/projects_statistics.php)

the hyperparameter tuning. In particular, for BERT training, their optimizer enables the use of huge batch sizes of 32,868 without any degradation of performance. By increasing the batch size to the memory limit of a TPUv3 Pod, BERT training time can be reduced from 3 days to just 76 minutes.

Most of these works address training large deep-learning models using TPU Pods. Differently from the related works, we used a single TPU with eight cores, and although we didn't explore optimization techniques, we achieved interesting results. In addition, we also performed a cost analysis and showed that the preemptive TPU can achieve better cost efficiency than the local cluster.

### 3 BACKGROUND

This Section provides an overview of the TPUv3 architecture used in the experiments and presents concepts about Diabetic Retinopathy (DR).

#### 3.1 The Google Cloud TPU

In this section, we introduce the TPU architecture developed by Google, which is utilized in our experiments. TPUs, or Tensor Processing Units, are application-specific integrated circuits (ASICs) that are specifically designed to accelerate machine learning workloads. As shown in Figure 1, a TPUv3 device has a structure of four internal chips, each of which comprises of two cores. Each core is equipped with scalar, vector, and matrix units (MXU) that are connected to the on-chip high bandwidth memory (HBM) of 16 GB per TPUv3 core. The TPUv3 offers a peak performance of 420 TFlops of floating point throughput (Ying et al., 2018). The cores of the TPU device perform calculations independently, and the high-bandwidth interconnections enable the chips to communicate with one another within the TPU device. These TPUs can be used to train and run large machine learning models and also can be used for other high-performance computing tasks (Google, 2023a).

When working with the Cloud TPU model, it is important to configure it correctly in order to take advantage of the distributed training capabilities of the device. One strategy for doing this is to scale the batch size by the number of TPU cores that are available. For example, if the batch size is 32, the global batch size will be 256 (8 cores  $\times$  32 = 256) (Künas et al., 2021). This means that each core will process a batch of 32 examples, and the results will be combined across all cores to form the final out-

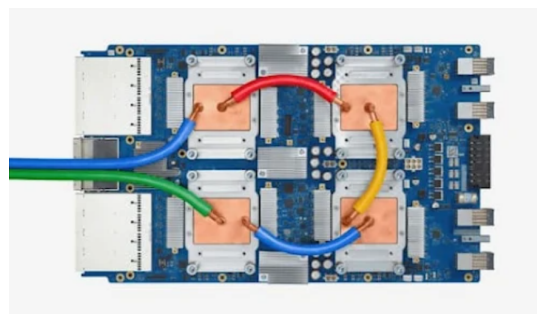


Figure 1: The architecture of TPUv3 device with four chips, 420 TFlops of peak floating point throughput and 128 GB of HBM.

put. The global batch size is then automatically fragmented across all replicas, which allows for efficient processing of large data sets. This approach allows for the parallel processing of multiple examples at once, which can greatly speed up the training process.

#### 3.2 Diabetic Retinopathy

*Diabetes Mellitus* is a metabolic disorder characterized by an abnormal increase in blood sugar levels. The patient will be subject to complications such as heart attack, stroke, kidney failure, hard-to-heal injuries, and vision problems when not appropriately treated (Zheng et al., 2018). Vision problems occur because diabetes affects the circulatory system, including progressive vascular ruptures, and can develop regardless of the severity of the patient. One specific vision problem caused by diabetes is diabetic retinopathy (DR) (Janghorbani et al., 2000). DR can be seen in Figure 2, which illustrates a comparison between a healthy retina and a retina affected by the disease.

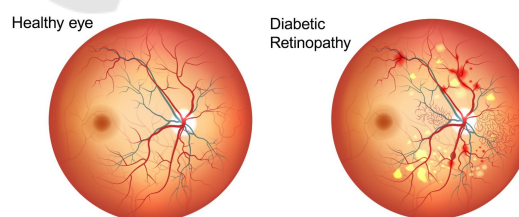


Figure 2: Comparison of a healthy retina and retina with diabetic retinopathy.

The *Global Diabetic Retinopathy Project Group* (Wilkinson et al., 2003) has proposed a five-stage classification protocol for PDR and NPDR. The stages are as follows:

- **No Apparent Retinopathy:** no abnormalities are present.
- **Mild Non-Proliferative Diabetic Retinopathy:**

the presence of retinal microaneurysms.

- **Moderate Non-Proliferative Diabetic Retinopathy:** more than just microaneurysms, but less severe than stage IV.
- **Severe Non-Proliferative Diabetic Retinopathy:** the presence of more than 20 intra-retinal hemorrhages in each of the four quadrants, venous beading in at least two quadrants, and intra-retinal microvascular abnormalities in at least one quadrant, in the absence of PDR.
- **Proliferative Diabetic Retinopathy:** characterized by neovascularization and vitreous or pre-retinal hemorrhage.

Regular eye examinations are crucial in tracking the severity level of diabetic retinopathy (DR) for people with diabetes mellitus. Timely diagnosis and treatment of DR are essential (Network, 2010), as the condition can progress to advanced stages without producing any immediate symptoms, thereby increasing the risk of vision loss (Stütt et al., 2016).

## 4 METHODOLOGY

The goal of this research is to offload the DL model training by using the cloud. Our inspiration codebase is the Voets reproduction<sup>2</sup> (Voets et al., 2019). Our model uses the Inception v3 architecture to transfer learning. We initialized the network with imagenet weights. In Figure 3, the 42 layers of the Inception v3 architecture are detailed.

After loading the imagenet weights, we add a *Global Average Pooling 2D* layer and two *Dense* layers, the first fully connected with 1024 units using *Rectified Linear Unit* (ReLU) activation function, and the second using *Softmax* activation function and five units, one to each of 5 classes.

The model uses Adam optimizer, a gradient descent algorithm based on the adaptive estimation of first and second-order moments. The learning rate value was 0.0014. The accuracy was collected to judge the model. The loss function calculates the logarithmic loss between actual and predicted labels. In this paper, we use the Sparse Categorical Crossentropy function.

As input, we used Kaggle's EyePACS dataset<sup>3</sup>. This database is commonly used for deep learning applications in DR detection and is divided into two

<sup>2</sup><https://github.com/mikevoets/jama16-retina-replication>

<sup>3</sup><https://www.kaggle.com/competitions/diabetic-retinopathy-detection>

subsets, train and test. The training dataset contains 35,126 images, where 25,810 have no signs of disease; 2,443 present mild retinopathy; 5,292 present moderate retinopathy; 873 present grave retinopathy; and 708 present proliferative retinopathy. We split these images into training and validating datasets. Thus, we perform a 5-class classification.

First, we process all images by locating the center and radius of the eye fundus and redimensioning every picture to 256x256 pixels. The images dataset was converted to TFRecord format and then uploaded to the bucket on Google Storage for the training in the TPU device. Each TFRecord file contains 2,000 images (except the last one which has 1,126 images). We split the TFRecord files into training (80%) and validating (20%) datasets. Therefore, the training dataset consists of 28,000 images, and the validating dataset consists of 7,126 images. The dataset is public and available on Kaggle at <https://kaggle.com/datasets/cristianokunas/diabetic-tfrecords256>.

TFRecord, TensorFlow's custom data format, is a powerful tool. It's natively supported by the high-performance `tf.data` API, can handle distributed datasets and takes advantage of parallel I/O. Working with large datasets can greatly benefit from using a binary file format for storage. Binary data consumes less space on disk, is faster to transfer, and can be read more efficiently. Using a binary file format can lead to a faster import pipeline and ultimately reduce the training time for your model. In addition to performance benefits, the TFRecord file format is also optimized for use with TensorFlow. It simplifies combining multiple datasets, and seamlessly integrates with the data import and preprocessing features of the library. This is particularly useful for datasets that are too large to fit in memory (88.29 GB for the EyePACS dataset), as only the necessary data (e.g. a batch) is loaded from the disk and processed at a time. Overall, the TFRecord file format provides a convenient and efficient way to work with large datasets in TensorFlow.

### 4.1 Software Setting

A recent survey showed that Python remains the top language for deploying, executing, and integrating ML/DL algorithms and related tasks like data transformation (Wang et al., 2019b). Its popularity stems from its ease of learning, fast implementation, and rich environment, including popular ML/DL frameworks like Caffe, Tensorflow, Torch, and MXNet.

Our application was deployed using Python 3.7.3 and the embedded frameworks Tensorflow (2.6.0) and

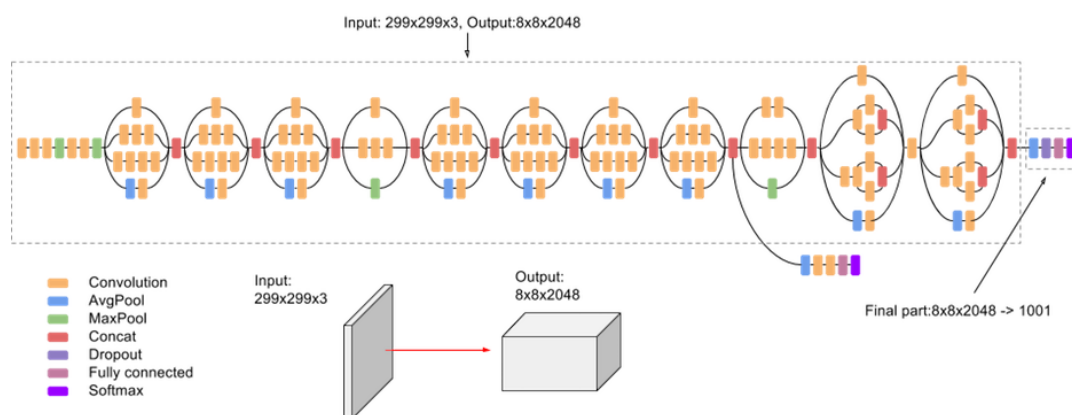


Figure 3: Inception V3 architecture. Source: <https://cloud.google.com/static/tpu/docs/images/inceptionv3onc-overview.png>.

Keras (2.6.0). We used CUDA Toolkit 11.8 and cuDNN 8.7 for GPU versions, following each developer's recommended installation procedures.

## 4.2 Experimental Platforms

The experiments described in this paper were conducted on the computational resources available at the Google Cloud<sup>4</sup>, in the PCAD infrastructure at INF/UFRGS<sup>5</sup>, and Santos Dumont Supercomputer (SDumont)<sup>6</sup> at the National Laboratory for Scientific Computing (LNCC)<sup>7</sup>:

- **Cloud TPUv3:** We use a single TPUv3 with 8 cores and 128 GB memory. The TPU device provides 420 Teraflops performance. This environment is named **TPUv3** throughout the rest of this paper.
- **Blaise:** A single compute node composed of two Intel Xeon E5-2699 v4 Broadwell (2.2GHz) CPU, 44 physical cores (22 per socket), 256 GB of RAM, and four NVIDIA Tesla P100-SXM2-16GB. All the experiments conducted used only one GPU. This environment is named **P100** throughout the rest of this paper.
- **Bull Sequana X1120 (GPU):** A single compute node composed of two Intel Xeon Cascade Lake Gold 6252 (2.1GHz) CPU, 48 physical cores (24 per socket), 384 GB of RAM, and four NVIDIA Tesla V100-SXM2-32GB. All the experiments conducted also used only one GPU. This environment is named **V100** throughout the rest of this paper.

<sup>4</sup><https://cloud.google.com/>

<sup>5</sup><http://gppd-hpc.inf.ufrgs.br/>

<sup>6</sup><https://sdumont.lncc.br>

<sup>7</sup><https://www.lncc.br>

## 5 RESULTS

In this section, we showcase the performance evaluation results obtained from the experimental platform mentioned in the previous section. We present metrics for execution time for different architectures. The results presented are an average of at least 10 runs, with a relative error of less than 5% and a 95% confidence level using the t-Student distribution. We also present the accuracy achieved by our model and perform a cost efficiency analysis when using TPUs compared to a local cluster.

### 5.1 Performance Evaluation

As mentioned previously, we use the Inception V3 architecture. After loading and initializing the network with the imagenet weights and adding all layers described in the previous section, we trained the model on 28,000 samples and validated it on 7,126 samples, with a batch size of 32 and limiting to 25 epochs. For the TPU, we use the strategy presented in Section 3.1, where each core processes 32 examples, resulting in a global batch size of 256.

The performance results of our study are presented in Fig. 4, showcasing that the V100 outperformed the P100 in terms of average training time, with a 1.63× improvement. The TPUv3, on the other hand, showed a remarkable performance with an average training time that was 3.48× faster than the V100. This result is even more significant when compared to the P100, where the TPUv3 demonstrated an improvement of 5.63× in terms of average training time.

This gain is achieved without any code optimization. Furthermore, although there is a time spent to transfer the dataset to the cloud to run on the TPU device, the performance of TPUv3 is still quite con-

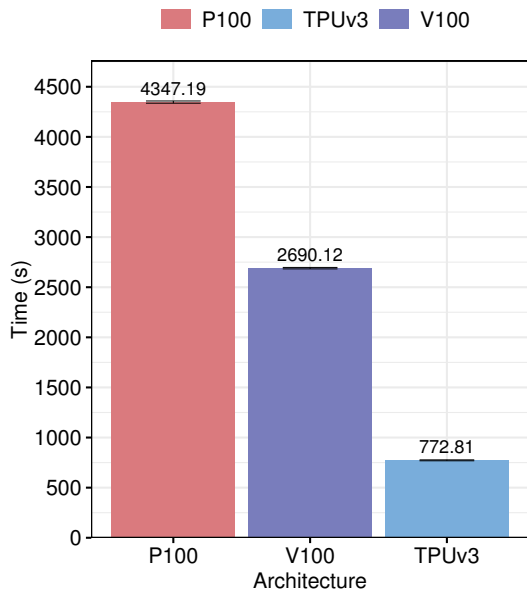


Figure 4: Neural Network Training times in different hardware.

siderable, about  $4.95\times$  and  $3.06\times$  more effective than P100 and V100, respectively. The information about the transfer is presented in Table 1. The measurements demonstrate that the average throughput was  $\approx 12.2$  MBytes/second, with an average size of 20.73 MBytes per TFRecord file.

Table 1: Dataset transfer measurements to the cloud. The edge location is in the PCAD infrastructure at INF/UFRGS. The cloud location is in the Google Cloud Storage - Iowa-us-central1.

Parameter	Edge to Cloud
Time	106.510 seconds
Average throughput	12.2 MBytes/second
Total Tranferred	932.2 MBytes

The results of this study are important in the field of deep learning, where the ability to process large amounts of data in a timely and efficient manner is crucial. TPUs have a higher computation density, meaning they can perform more operations in a smaller space. This allows for more efficient use of the chip’s power and results in faster training times. Our findings demonstrate that the TPUv3 outperforms the other devices in average training time, making it an attractive option for researchers and practitioners looking to improve their deep-learning models.

Increasing the global batch size is necessary to fully utilize the TPU cores when training deep learning models. This is because the TPU cores operate on the XLA memory layout (Google, 2023b), which

requires each tensor’s batch dimension to be a multiple of 8 (Jouppi et al., 2017) to more optimally utilize the memory of each TPU core and increase throughput. However, it’s important to note that training with large batch sizes can lead to a degradation in model quality due to the ”generalization gap” (Keskar et al., 2016). This has been observed compared to models trained with smaller batch sizes.

Although we did not explore optimization techniques for scaling training to large batch sizes, such as selecting large batch optimizers and learning rate schedules, as well as utilizing distributed evaluation techniques and batch normalization, we still achieved good performance on the TPUv3.

## 5.2 Accuracy Evaluation

This section compares the accuracy collected from ten executions of the Neural Network model on the P100, V100, and TPUv3 architectures. All values shown are averages, and the t-test (Kim, 2015) is used to compare them. The model’s average accuracy achieves 85.45%, 81.59% and 81.43% for TPUv3, V100, and P100, respectively, with a standard deviation of less than 1% on both architectures. Fig. 5 depicts the accuracy when running in each architecture.

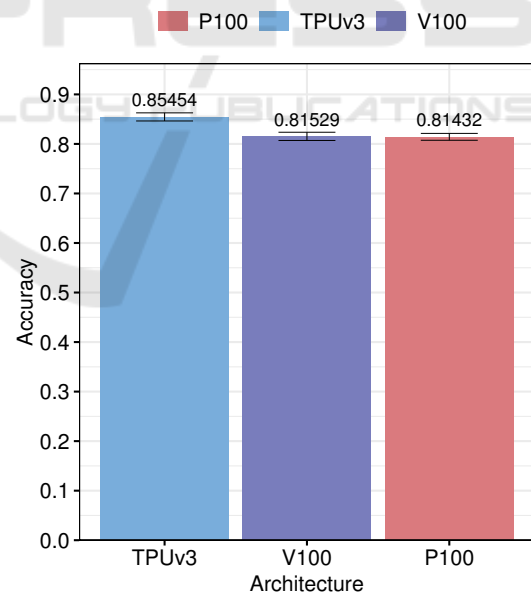


Figure 5: Neural Network Accuracy in different hardware.

The 85% accuracy of the diabetic retinopathy detection model is deemed appropriate. This is reinforced by the fact that other studies have shown similar accuracy rates (Lin et al., 2018; Ghosh et al., 2017), including using the Inception V3 architecture (Mohammadian et al., 2017). It is important to high-

light that the early detection of diabetic retinopathy is crucial to avoid serious vision complications. A model with an accuracy of 85% can provide reliable results and significantly contribute to the accurate diagnosis of the disease.

### 5.3 Cost Evaluation

We also analyze the *cost efficiency* of using the cloud for model training by scaling the performance value with the price per hour. Table 2 shows the cost per hour for a local cluster and the Google Cloud TPU<sup>8</sup>. The value for the cluster was calculated as follows. We consider the hardware cost for a machine of \$25,000 and that the machine will be used for one year, so we arrive at a hardware cost per hour of \$2.85.

We do not measure the price for facilities, personnel, and power consumption and do not include them in the total price. The TPUv3 cost per hour is about  $\approx 2.8\times$  higher than the local cluster. However, the performance achieved was  $\approx 3.48\times$  better.

Table 2: Cost (in Dollar/hour) of each solution.

Device	Cost
TPUv3-8	\$8.00
Local cluster	\$2.85

On the local cluster, our estimated cost would be \$2.13 to train the model. On the other hand, TPUv3, costing \$8 per hour, gives us an average cost per training of \$1.72. TPU is about 19% more efficient, i.e., it costs 19% less to perform the same amount of work in the cloud than in the local cluster. This indicates that Cloud TPU can be a good choice for training deep learning models, especially for our case, the Diabetic Retinopathy model.

Additionally, the cost per training can be further reduced by using preemptible TPUs. Preemptible TPUs cost much less than non-preemptible ones, about 70% less. However, it can be interrupted at any time. In this case, the application must be restart-resilient to save model checkpoints regularly and restores the most recent one upon restart. In our case study, the estimated cost of using preemptible TPUs is \$0.52. This is  $3.3\times$  better than the on-demand TPU and represents a cost efficiency of around 75% compared to the local cluster.

<sup>8</sup><https://cloud.google.com/tpu/pricing>

## 6 CONCLUSION AND FUTURE WORK

TPUs are specialized hardware devices designed to accelerate machine learning workloads, especially matrix operations used in deep learning algorithms. TPUs are considered better for deep learning tasks because they provide a high computational performance, efficient and cost-effective solution for accelerating these workloads.

In this paper, we sought to evaluate the performance of the Diabetic Retinopathy model training by asynchronously offloading the training to the cloud using TPU devices. Such an approach aids in alleviating the contention for high-demanded local HPC resources, allowing them to be focused on running applications. We adjusted the neural network model to be trained on TPUs, and have seen encouraging outcomes, including shorter training time, with gains of up to  $5.63\times$  in the best case, even without any optimizing the code. Our results provide a good starting point for those interested in improving the performance of their deep-learning models.

Future work will extend the performance evaluation to the Cloud TPUv4 and the TPU Pods, exploring optimization techniques, such as selecting large batch optimizers and learning rate schedules, to scale training to large batch sizes.

## ACKNOWLEDGEMENTS

This study was partially supported by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior – Brasil (CAPES) – Finance Code 001, by Petrobras grant n.º 2020/00182-5, by CNPq/MCTI/FNDCT - Universal 18/2021 under grants 406182/2021-3, MCTIC/CNPq - Universal 28/2018 under grants 436339/2018-8, by CIARS RITEs/FAPERGS project and by CI-IA FAPESP-MCTIC-CGI-BR project. Some experiments in this work used the PCAD infrastructure, <http://gppd-hpc.inf.ufrgs.br>, at INF/UFRGS. The authors acknowledge the National Laboratory for Scientific Computing (LNCC/MCTI, Brazil) for providing HPC resources of the SDumont supercomputer, which have contributed to the research results reported within this paper. URL: <http://sdumont.lncc.br>.

## REFERENCES

- Abiodun, O. I., Jantan, A., Omolara, A. E., Dada, K. V., Mohamed, N. A., and Arshad, H. (2018). State-of-the-

- art in artificial neural network applications: A survey. *Heliyon*, 4(11):e00938.
- Amodei, D., Ananthanarayanan, S., Anubhai, R., Bai, J., Battenberg, E., Case, C., Casper, J., Catanzaro, B., Cheng, Q., Chen, G., et al. (2016). Deep speech 2: End-to-end speech recognition in english and mandarin. In *International conference on machine learning*, pages 173–182. PMLR.
- Ghosh, R., Ghosh, K., and Maitra, S. (2017). Automatic detection and classification of diabetic retinopathy stages using cnn. In *2017 4th International Conference on Signal Processing and Integrated Networks (SPIN)*, pages 550–554. IEEE.
- Google (2023a). Cloud tpu system architecture. [Accessed Jan. 23, 2023].
- Google (2023b). Xla: Google’s accelerated linear algebra library. [Accessed Jan. 26, 2023].
- Hatcher, W. G. and Yu, W. (2018). A survey of deep learning: Platforms, applications and emerging research trends. *IEEE Access*, 6:24411–24432.
- Hestness, J., Ardalani, N., and Diamos, G. (2019). Beyond human-level accuracy: Computational challenges in deep learning. In *Proceedings of the 24th Symposium on Principles and Practice of Parallel Programming*, pages 1–14.
- Janghorbani, M., Jones, R. B., and Allison, S. P. (2000). Incidence of and risk factors for proliferative retinopathy and its association with blindness among diabetes clinic attenders. *Ophthalmic Epidemiology*, 7(4):225–241.
- Jouppi, N. P., Young, C., Patil, N., Patterson, D., Agrawal, G., Bajwa, R., Bates, S., Bhatia, S., Boden, N., Borchers, A., et al. (2017). In-datacenter performance analysis of a tensor processing unit. In *Proceedings of the 44th annual international symposium on computer architecture*, pages 1–12.
- Keskar, N. S., Mudigere, D., Nocedal, J., Smelyanskiy, M., and Tang, P. T. P. (2016). On large-batch training for deep learning: Generalization gap and sharp minima. *arXiv preprint arXiv:1609.04836*.
- Kim, T. K. (2015). T test as a parametric statistic. *Korean journal of anesthesiology*, 68(6):540.
- Künas, C. A., Serpa, M. S., Bez, J. L., Padoin, E. L., and Navaux, P. O. (2021). Offloading the training of an i/o access pattern detector to the cloud. In *2021 International Symposium on Computer Architecture and High Performance Computing Workshops (SBAC-PADW)*, pages 15–19. IEEE.
- Lin, G.-M., Chen, M.-J., Yeh, C.-H., Lin, Y.-Y., Kuo, H.-Y., Lin, M.-H., Chen, M.-C., Lin, S. D., Gao, Y., Ran, A., et al. (2018). Transforming retinal photographs to entropy images in deep learning to improve automated detection for diabetic retinopathy. *Journal of ophthalmology*, 2018.
- Mohammadian, S., Karsaz, A., and Roshan, Y. M. (2017). Comparative study of fine-tuning of pre-trained convolutional neural networks for diabetic retinopathy screening. In *2017 24th National and 2nd International Iranian Conference on Biomedical Engineering (ICBME)*, pages 1–6. IEEE.
- Network, S. I. G. (2010). Management of obesity: a national clinical guideline. *Scottish Intercollegiate Guidelines Network: Edinburgh*, 20.
- Roloff, E. (2013). Viability and performance of high-performance computing in the cloud. Master’s thesis, Federal University of Rio Grande do Sul.
- Stitt, A. W., Curtis, T. M., Chen, M., Medina, R. J., McKay, G. J., Jenkins, A., Gardiner, T. A., Lyons, T. J., Hammes, H.-P., Simo, R., et al. (2016). The progress in understanding and treatment of diabetic retinopathy. *Progress in retinal and eye research*, 51:156–186.
- Sun, C., Shrivastava, A., Singh, S., and Gupta, A. (2017). Revisiting unreasonable effectiveness of data in deep learning era. In *Proceedings of the IEEE international conference on computer vision*, pages 843–852.
- Voets, M., Møllersen, K., and Bongo, L. A. (2019). Reproduction study using public data of: Development and validation of a deep learning algorithm for detection of diabetic retinopathy in retinal fundus photographs. *PloS one*, 14(6):e0217541.
- Wang, Y. E., Wei, G.-Y., and Brooks, D. (2019a). Benchmarking tpu, gpu, and cpu platforms for deep learning. *arXiv preprint arXiv:1907.10701*.
- Wang, Z., Liu, K., Li, J., Zhu, Y., and Zhang, Y. (2019b). Various frameworks and libraries of machine learning and deep learning: a survey. *Archives of computational methods in engineering*, pages 1–24.
- Wilkinson, C., Ferris III, F. L., Klein, R. E., Lee, P. P., Agardh, C. D., Davis, M., Dills, D., Kampik, A., Pararajasegaram, R., Verdager, J. T., et al. (2003). Proposed international clinical diabetic retinopathy and diabetic macular edema disease severity scales. *Ophthalmology*, 110(9):1677–1682.
- Wongpanich, A., Pham, H., Demmel, J., Tan, M., Le, Q., You, Y., and Kumar, S. (2021). Training efficient-nets at supercomputer scale: 83% imagenet top-1 accuracy in one hour. In *2021 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*, pages 947–950. IEEE.
- Ying, C., Kumar, S., Chen, D., Wang, T., and Cheng, Y. (2018). Image classification at supercomputer scale. *arXiv preprint arXiv:1811.06992*.
- You, Y., Li, J., Reddi, S., Hseu, J., Kumar, S., Bhojanapalli, S., Song, X., Demmel, J., Keutzer, K., and Hsieh, C.-J. (2019a). Large batch optimization for deep learning: Training bert in 76 minutes. *arXiv preprint arXiv:1904.00962*.
- You, Y., Zhang, Z., Hsieh, C.-J., Demmel, J., and Keutzer, K. (2018). Imagenet training in minutes. In *Proceedings of the 47th International Conference on Parallel Processing*, pages 1–10.
- You, Y., Zhang, Z., Hsieh, C.-J., Demmel, J., and Keutzer, K. (2019b). Fast deep neural network training on distributed systems and cloud tpus. *IEEE Transactions on Parallel and Distributed Systems*, 30(11):2449–2462.
- Zheng, Y., Ley, S. H., and Hu, F. B. (2018). Global aetiology and epidemiology of type 2 diabetes mellitus and its complications. *Nature reviews endocrinology*, 14(2):88–98.