The Importance of High Speed Storage in Deep Learning Training

Solene Bechelli
Department of Biomedical Engineering
University of North Dakota
Grand Forks, ND USA
solene.bechelli@und.edu

David Apostal
Computational Research Center
University of North Dakota
Grand Forks, ND 58202-8399
david.apostal@und.edu

Aaron Bergstrom
Computational Research Center
University of North Dakota
Grand Forks, ND 58202-8399
aaron.bergstrom@und.edu

***Abstract*—With the increase of computational power and techniques over the past decades, the use of Deep Learning (DL) algorithms in the biomedical field has grown significantly. One of the remaining challenges to using deep neural networks is the proper tuning of the model's performance beyond its simple accuracy. Therefore, in this work, we implement the combination of the NVIDIA DALI API for high-speed storage access alongside the TensorFlow framework, applied to the image classification task of skin cancer. To that end, we use the VGG16 model, known to perform accurately on skin cancer classification. We compare the performance between the use of CPU, GPU and multi-GPU devices training both in terms of accuracy and runtime performance. These performances are also evaluated on additional models, as a mean for comparison.**

Our work shows the high importance of model choice and fine tuning tailored to a particular application. Moreover, we show that the use of high-speed storage considerably increases the performance of DL models, in particular when handling images and large databases which may be a significant improvement for larger databases.

***Index Terms*—High-speed storage, deep learning, image classification, cancer application, application specific storage**

I. Introduction

With the development of new DL algorithms in various fields, related challenges have evolved into making more efficient workflows both in terms of computer runtime, and validity of the output. As such I/O performance is becoming an important area of research.

The use of High-Performance Computing (HPC) clusters in the biomedical field is constantly increasing. In 2013, HPC systems were primarily used for simulations related to brain activity, heat capacity calculations, molecular dynamics simulations, etc. With the emergence of machine learning in this field, HPC usage has shifted toward classification, segmentation, and prediction tasks using machine learning and deep learning algorithms. [1] A main focus of current studies is toward improving the rate of success of predictions, which, if highly relevant to avoid misdiagnosis, is also more time consuming with the addition of data augmentation, and the requirement for larger databases. To this end, benchmarking HPCs for biomedical applications allows for better tailoring algorithms such as machine learning and deep learning to the type of hardware used, to improve both scaling and time computing performances. [2] Studies have been performed to improve model performance, however, both hardware and software structure constantly evolve which in turn forces our approach to deep learning algorithms to change and fit such structures. [2] Therefore, the number of benchmarking studies for GPU storage has greatly increased since 2017 to provide a comprehensive overview of the available computer power and their related code structure for deep learning applications. [3]

In addition to the ever changing HPC architecture and hardware, many models are used in biomedical researches such as VGG16, AlexNet, ResNet50, etc. However, if some have been highly benchmarked and a significant number of studies is available, [4], [5] this knowledge remains sparse for other models. Benchmarking is not easily extended from one model to another and needs to be tailored to the specificity of each DL models. Moreover, a model's efficiency is shown to be sensitive to the type of hardware used. [4] This makes the study of deep learning model performance on both CPU and GPU of high relevance to further improve future discoveries in many fields. Most of the current research focuses on the achievement of high performance for a given task to the detriment of a model's runtime and computing efficiency leading to long training time and heavy memory workload. [4] However, appropriate benchmarking of deep learning frameworks need to consider both the training time and the overall performance of one's model. [6]

One of the main runtime and performance limitations for DL algorithms resides in access to the training database. These databases are often sizable and for both loading and preprocessing operations, this task remains time consuming under most systems. [7] Due to the large number of entries required to obtain accurate model predictions, the time spent loading data and performing data augmentation can be significant. [8]

This study aims to transfer part of the workload handled by the CPU to the GPU available during the pre-processing steps. This is enhanced by the use of an AI400 storage unit from DataDirect Networks, but also to reduce the training time by transferring images directly to GPU devices memory, without needing to use CPUs. Thus, we discuss the performance of

Research presented in this paper was supported by the National Science Foundation under NSF EPSCoR Track-1 Cooperative Agreement OIA #1946202. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

the NVIDIA DALI API combined with an AI400 high-speed storage unit for DL models in the biomedical field. We analyze the performance of several deep learning models as well as the effect of some of the tuning parameters on the performance of said models. We first discuss previous findings in this field. We then introduce the simulation method and the machine used including the structure and hardware of the HPCs. Eventually, we compare the performance of multiple deep learning models on both CPUs and GPUs devices while accessing high-speed storage through DALI.

II. Related work

Storage options and input optimization have been the focus of several studies either for local physical storage options, [9], [10] or for cloud computing approaches. [11] One study has demonstrated that the training time of a DL model is greatly impacted by the implementation of the input. [12] In the attempt of improving the dataset access, TensorFlow has developed its *tf.data* framework which has showed higher performance than other approaches. [12] Additionally, many studies have used the DALI API by NVIDIA, [13], [14] or similar API, [15] to enhance data loading.

One of the challenges of I/O optimization is its dependency to the deep learning framework used (TensorFlow, PyTorch, etc.). By decoupling the I/O from the DL framework, Macedo et al. significantly decreased the training time required for several frameworks. [9]

Following the observation that framework can lead to various performances, some work showed the comparison of platform performance, from TensorFlow to CAFFE and PyTorch, [2], [6], [16], [17] with Pytorch currently preferred in academia and TensorFlow in industry. [16]

The comparison of CPU and GPU performance in deep learning has also been evaluated by several studies showing great improvement made with GPU usage. [2], [18], [19]

Moreover, as previously mentioned, most of the current studies of deep learning in the biomedical field focus on the improvement of accuracy and metrics related to the prediction rather than on the runtime efficiency of a model. [20], [21]

The purpose of this work is not to improve the accuracy of skin cancer prediction because studies have already shown the tremendous improvement that have been achieved in the past years, but more so to enhance the runtime performance ability of such models. [22]

III. Software and Hardware used

A. HPC structure and hardware

In order to assess the effect of high-speed storage on deep learning models, we use both CPU, and GPU devices, with the following characteristics: (1): the CPU processors are Intel Xeon Gold 6140 processors with 36 cores per node and 192Gb RAM. (2): The GPU devices are 8 NVIDIA Tesla V100 GPUs per node with 32GB of HBM2 VRAM each (5120 CUDA cores per card, 640 Tensor cores per card).

The access to the AI400 high-speed storage unit is done through the NVIDIA DALI API 1.14.0. [23]

B. TensorFlow and deep learning models

In this work, we use Python 3.6 and Tensorflow 2.6's framework to train three different models: InceptionV3, ResNet50 and VGG16. Deep learning models are well suited to study GPU performances, because most of their calculations are done on vectors. [2], [16] The choice of the models is motivated in part by their different structures, but also by their wide usage in biomedical researches. We use the ResNet50 model to allow for comparison with literature work as this model time performance has been widely studied. [7], [24], [25] The image database is separated into training and testing/validation with a 80/20 ratio.

C. The database and related performance

For this work, we use the dermoscopic database provided by the Kaggle community. [26] This database is a combination of approximately 3000 images separated into benign and malignant categories. This database has been used in various works and has shown to provide accurate results in the prediction of skin cancers. Its smaller size allows us to obtain quick results and is easily scalable for further studies on larger datasets.

IV. Evaluation

A. CPU calculations

First we look into the overall performance of the deep learning models chosen on CPUs. This will be used as benchmarking base to measure the performance improvement provided by our other approaches. Table I provides the time performance results for each model as well as the number of related parameters. Each result is normalized either in terms of numbers of images per epochs, or in terms of time taken for one image to be handled which is the time to forward propagate one image during training.

TABLE I
Performance time and model's structure for each of the model studied.

Model types	Time/image/epochs (s)	# of parameters	# of layers
Inception V3	0.093	$\approx$ 24 millions	311
ResNet50	0.101	$\approx$ 23 millions	50
VGG16	0.167	$\approx$ 138 millions	16

We note that the difference in performance is based on a combination of the number of layers and the related number of trainable parameters. The InceptionV3 and ResNet50 models have similar performance in terms of time and a close number of parameters. Their difference lies in the number of layers which may be an explanation for the slightly slower time provided by the Inception model. On the other hand, the VGG16 model with about 138 millions trainable parameters led to the largest performance time of 0.167 s/image/epoch.

In biomedical studies, data augmentation is widely used to provide larger database leading higher prediction performance with fewer mistakes. Therefore, we compare the models performance using different types of data augmentation: random flip, random zoom, and random rotation. Fig. 1, 2, and 3 show

that if data augmentation may affect the accuracy of a model, it does not impact the running time of small databases by much. However, because this process consists of scaling the original database, the bigger the original training set, the more impact from the data augmentation process will be observed. This difference may become significant in larger databases studies and must be taken into account then.

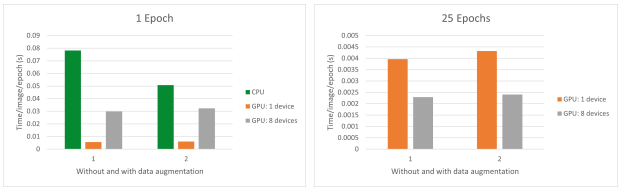

Fig. 1. Hardware performance for the InceptionV3 model for both 1 and 25 epochs.

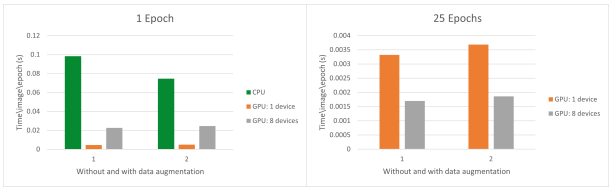

Fig. 2. Hardware performance for the ResNet50 model for both 1 and 25 epochs.

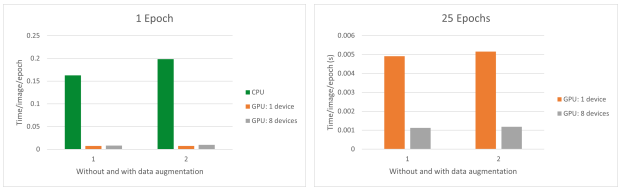

Fig. 3. Hardware performance for the VGG16 model for both 1 and 25 epochs.

B. Single GPU calculations

In this section, we implement the NVIDIA DALI API to assess the impact of the use of the AI400 high-speed storage device. This API has provided the best results when used in conjunction with GPU devices, therefore, we compare the performance of the use of a single GPU device with and without high-speed storage access.

First, Fig. 1, 2, and 3 all show that single- and multi-GPU usage provide significant improvement from the CPU calculations. We also note that using data augmentation had less impact on runtime performance on GPU than on the CPU. Table II provides the ratio of time decrease between CPU and single-GPU device usage.

Table II shows that the use of GPU devices provide significant improvement of the training time for each of the models. However, we also note that the increase is not the same for each models. This finding confirms the results of Zhu et al. showing that hardware architecture directly correlates with the

TABLE II
PERFORMANCE DIFFERENCES BETWEEN CPU AND SINGLE GPU PERFORMANCES FOR BOTH THE FIRST EPOCH AND THE FOLLOWING 24 EPOCHS. THE NUMBERS ARE A RATIO OF THE NUMBER OF IMAGES PER SECOND PER EPOCH.

Model types	1st epoch: 1GPU/CPU	24 epochs: 1GPU/CPU
Inception V3	≈ 14	≈ 23
ResNet50	≈ 22	≈ 31
VGG16	≈ 22	≈ 34

model's performance. [4] For instance, the VGG16 model, which has the largest number of trainable parameters is the one with the largest increase and therefore is the one we will focus on for the remainder of this work.

Lastly, Fig. 1, 2, and 3 also demonstrate that the first epoch is significantly slower than the following ones. This phenomenon has been the topic of many discussions and has so far been attributed to the amount of disk work required during the first epoch, which is not perform during the following epochs. This is in part due to the fact that some layers (particularly the final layer of softmax distribution for binary prediction) is initialized with nodes of random value for which the gradient descent performed during the first epoch of training may require time before reaching convergence.

We now turn to the results obtained when using the NVIDIA DALI API to access our database. Combining AI storage access with GPU calculation significantly increases the speed of the training overall. For the VGG16 model, the performance goes from ≈ 203 image/s compared to up to ≈ 804 image/s on high-speed storage when using the *model.fit* method provided in TensorFlow. We also compared the performance of a custom training loop that along side high-speed storage still performed better than classical storage (211 image/s), but remain less time efficient that the *model.fit* method.

In addition, we analyze the effect of the batch size on the performance of the VGG16 model. Decreasing the batch size leads to fewer images handled per second. This result is expected because the workload on each of the GPU significantly increases, which has been thoroughly explained by Kouchura et al. [27] However, we observed that the batch size influences the performance of each methodology with an optimum batch size found at 16 to 32, which is in agreement with literature work and current common practices in deep learning training. This may be further explained as we kept the learning rate constant throughout the workflow. As mentioned by He et al., the monitoring of the learning rate as a linear function of the batch size may significantly increase the models' performance. [28] Similar works have show the importance of the learning rate choice particularly with learning rate warm-ups. [29]

Thus, we study the impact of batch sizing and learning rate tuning on the single GPU performance. To do so, we compare the runtime performance of the change of batch size keeping a fixed learning rate with the change of batch size and a varying learning rate. The learning rate is tuned using the same method as He et al.: $0.0001 * b/16$ with b the new batch size. [28]

TABLE III
COMPARISON OF LEARNING RATE TUNING PERFORMANCE

	Fixed		Tuned	
Batch size	1st	average	1st	average
1	15.13	12.44	12	9.155
16	7.25	4.18	7.25	4.18
32	8	4	8	3.8

Table III shows that tuning the learning rate as a function of the batch size may slightly improve the time performance of a model in the case of smaller batch sizes (batch size of 1). However, the change is less significant in the case of larger batch sizes. In addition, we monitor the change in accuracy for both approaches. If no overfitting is observed, the accuracy of batch size 1 is significantly decreased by $4-5\%$. This is not observed when using fixed learning rate, and therefore shows that the runtime performance increase comes to the detriment of prediction accuracies.

C. Multi-GPU approach

We now turn to the use of multiple GPUs in the same HPC compute node. In this section we analyze the impact of using multiple GPUs (2 to 8 devices) and investigate the performance on the VGG16 model.

First, we look at Fig. 1 through 3, which show that the model's performance strongly depends on its CPU/GPU training. This finding is in agreement with previous work on CPU/GPU comparison for deep learning applications. [19]

Fig. 4 shows that the first epoch is more impacted by the change in batch size than the remaining part of the training. which is similar to what is observed for single GPU usage.

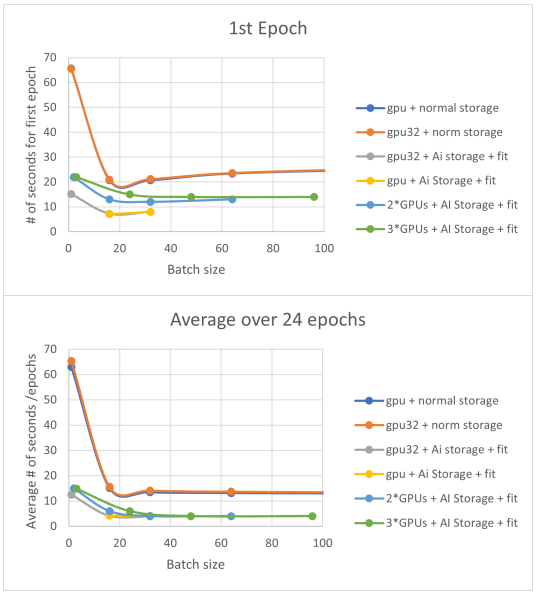

Fig. 4. Time performance evolution of the VGG16 model using different architecture approaches as a function of the batch size.

To emphasize this finding, Fig. 4 also shows the evolution of the performance for the first and averaged epochs using several processors as a function of the batch size. We note that for batch size greater than 64, the performance remained rather constant. We can attribute this behavior to the small database used. This has been widely studied, in particular in correlation with learning rate tuning in order to find the most suitable approach to obtain the most efficient training for the VGG16 model. [30], [31]

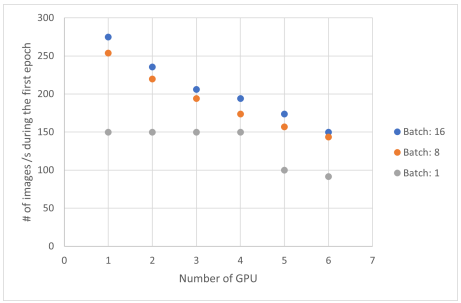

Fig. 5. Performance evolution as a function of the number of GPU devices used for smaller batch size. The learning rate is not updated and kept at 0.0001.

In order to assess the best architecture possible, we also analyze the impact of the number of threads used during the loading and processing of the database. If using a larger number of threads does not improve the time performance of a model, it instead increases the ability to use multi-GPU devices.

Lastly, Fig. 5 shows that the increase in batch size per number of GPUs, does not necessarily increase the performance of a model. On the contrary, we note that regardless of the batch size chosen, increasing the number of GPUs for different batch size tends to decrease the number of images handled per second per epoch. This could also be tied to the need for multiple access to the database and the storage.

Eventually, we turn to the accuracy performance, which remains the same when using normal storage and AI storage for 1 GPU, however, the performance decreases when looking at 8 GPU devices with AI storage. The average accuracy value for most of the models is 0.80 with a standard deviation +- 0.006. The validation accuracy is close with 0.81 on average. On the other hand, the accuracy for the multi-GPU + AI combination is 0.85/0.90 for the training and 0.80 for the validation set which shows overfitting and misrepresentation of the dataset. We attribute this results to the sharding of the dataset to appropriately use all GPUs.

V. CONCLUSION

In this work, we have shown different approaches to improve deep learning efficiency using high-speed storage through the NVIDIA DALI API. We have shown that the use of this API can significantly improve the performance of deep learning models. We have also demonstrated that the performance of a model will be altered differently depending on the CPU/GPU architecture that is chosen. We note that the combination of multi-GPU approach with direct access to the AI 400 storage unit through the DALI API provides

the best results in terms of time efficiency. It should be kept in mind that the choice of the approach used may have dramatic consequences on the accuracy performance of a model particularly when using sharding approach to separate the dataset. Future work should emphasize on scaling this work to larger database to possibly analyze data starvation issue and optimization of such algorithms.

REFERENCES

[1] S. Bastrakov, I. Meyerov, V. Gergel, A. Gonoskov, A. Gorshkov, Efimenko, Evgeny, M. Ivanchenko, M. Kirillin, A. Malova, G. Osipov, and others, "High performance computing in biomedical applications," Procedia Comput. Sci., vol. 18, pp. 10–19, 2013.

[2] S. Shams, R. Platania, K. Lee, and S.-J. Park, "Evaluation of deep learning frameworks over different HPC architectures," IEEE 37th International Conference on Distributed Computing Systems (ICDCS), pp. 1389–1396, 2017.

[3] S. Dong, and D. Kaeli, "Dnnmark: A deep neural network benchmark suite for gpus," pp 63–72, 2017.

[4] H. Zhu, M. Akrout, B. Zheng, A. Pelegris, A. Jayarajan, A. Phanishayee, B. Schroeder, and G. Pekhimenko, "Benchmarking and analyzing deep neural network training," IEEE International Symposium on Workload Characterization (IISWC), pp 88–100, 2018.

[5] T. D. Le, H. Imai, Y. Negishi, and K. Kawachiya, "Automatic gpu memory management for large neural models in tensorflow," Proceedings of the 2019 ACM SIGPLAN International Symposium on Memory Management, pp 1–13, 2019.

[6] L. Liu, Y. Wu, W. Wei, W. Cao, S. Sahin, and Q. Zhang, "Benchmarking deep learning frameworks: Design considerations, metrics and beyond," IEEE 38th International Conference on Distributed Computing Systems (ICDCS), pp 1258–1269, 2018.

[7] A. Aizman, G. Maltby, and T. Breuel, "High performance I/O for large scale deep learning," IEEE International Conference on Big Data (Big Data), pp 5965–5967, 2019.

[8] M. Zolnouri, X. Li, and V.P. Nia,"Importance of data loading pipeline in training deep neural networks," arXiv preprint arXiv:2005.02130, 2020.

[9] R. Macedo, C. Correia, M. Dantas, C. Brito, W. and Xu, Y. Tanimura, J. Haga, and J. Paulo, "The Case for Storage Optimization Decoupling in Deep Learning Frameworks," IEEE International Conference on Cluster Computing (CLUSTER), pp 649–656, 2021.

[10] J. M. Wozniak, P. E. Davis, T. Shu, J. Ozik, N. Collier, M. Parashar, I. Foster, T. Brettin, and R. Stevens, "Scaling deep learning for cancer with advanced workflow storage integration,"IEEE/ACM Machine Learning in HPC Environments, MLHPC 2018, pp. 114–123, 2018.

[11] H. C. Kaskavalci, and S. Gören, "A deep learning based distributed smart surveillance architecture using edge and cloud computing," International Conference on Deep Learning and Machine Learning in Emerging Applications (Deep-ML), pp. 1–6, 2019.

[12] D. G. Murray, J. Simsa, A. Klimovic, and I. Indyk, "tf. data: A machine learning data processing framework," arXiv preprint arXiv:2101.12127, 2021.

[13] V. G. T. da Costa, E. Fini, M. Nabi, N. Sebe, and E. Ricci, "solo-learn: A Library of Self-supervised Methods for Visual Representation Learning," J. Mach. Learn. Res., vol. 23, pp. 56–1, 2022.

[14] L. Wang, Q. Luo, and S. Yan, "Accelerating Deep Learning Tasks with Optimized GPU-assisted Image Decoding," IEEE 26th International Conference on Parallel and Distributed Systems (ICPADS), pp. 274–281, 2020.

[15] J. Mohan, A. Phanishayee, A. Raniwala, and V. Chidambaram, "Analyzing and mitigating data stalls in dnn training," arXiv preprint arXiv:2007.06775, 2020.

[16] H. Dai, X. Peng, X. Shi, L. He, Q. Xiong, and H. Jin, "Reveal training performance mystery between TensorFlow and PyTorch in the single GPU environment," Sci. China Inf. Sci., vol. 65, pp 1–17, 2022.

[17] G. Al-Bdour, R. Al-Qurran, M. Al-Ayyoub, A. Shatnawi, " Benchmarking open source deep learning frameworks," Int. J. Electr. Comput., vol. 10, pp 5479, 2020.

[18] K. Aida-Zade, E. Mustafayev, and S. Rustamov, "Comparison of deep learning in neural networks on CPU and GPU-based frameworks," IEEE 11th International Conference on Application of Information and Communication Technologies (AICT), pp. 1–4, 2017.

[19] A. Jayasimhan,P. Pabitha, "A comparison between CPU and GPU for image classification using Convolutional Neural Networks," International Conference on Communication, Computing and Internet of Things (IC3IoT), pp. 1–4, 2022.

[20] M. Wang, and X. Gong, "Metastatic cancer image binary classification based on resnet model," IEEE 20th International Conference on Communication Technology (ICCT), pp. 1356–1359, 2020.

[21] Liu, Min and Hu, Lanlan and Tang, Ying and Wang, Chu and He, Yu and Zeng, Chunyan and Lin, Kun and He, Zhizi and Huo, Wujie, "A Deep Learning Method for Breast Cancer Classification in the Pathology Images," IEEE J. Biomed. Health Inform., 2022.

[22] Y. Chen, J. Deng, and T. Wang, "Skin Cancer Diagnosis and Medical Service System Based on Deep Learning Models," 3rd International Conference on Electronic Communication and Artificial Intelligence (IWECAI), pp. 367–371, 2022.

[23] J. A. Guirao, K. Lecki, J. Lisiecki, S. Panev, M. Szolucha, A. Wolant, and . Zientkiewicz, "Fast AI Data Preprocessing with NVIDIA DALI," https://devblogs.nvidia.com/fast-ai-data-preprocessing-with-nvidia-dali, 2019.

[24] W. Brewer, G. Behm, A. Scheinine, B. Parsons, W. Emeneker, and R.P. Trevino, "Inference benchmarking on HPC systems,"IEEE High Performance Extreme Computing Conference (HPEC), pp 1–9, 2020.

[25] A. Jain, AA. Awan, H. Subramoni, and D. K. Panda, "Scaling tensorflow, pytorch, and mxnet using mvapich2 for high-performance deep learning on frontera," IEEE/ACM Third Workshop on Deep Learning on Supercomputers (DLS), pp 76–83, 2019.

[26] "Skin Cancer: Malignant vs. Benign—Kaggle.," Available online: https://www.kaggle.com/fanconic/skin-cancer-malignant-vs-benign, 2021.

[27] Y. Kochura, Y. Gordienko, V. Taran, N. Gordienko, A. Rokovyi, O. Alienin, and S. Stirenko, "Batch size influence on performance of graphic and tensor processing units during training and inference phases," in International Conference on Computer Science, Engineering and Education Applications, 2019, pp. 658–668.

[28] T. He, Z. Zhang, H. Zhang, Z. Zhang, J. Xie, and M. Li, "Bag of tricks for image classification with convolutional neural networks," Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp 558–567, 2019.

[29] P. Goyal, P. Dollár, R. Girshick, P. Noordhuis, L. Wesolowski, A. Kyrola, A. Tulloch, Y. Jia, and K. He, "Accurate, large minibatch sgd: Training imagenet in 1 hour," arXiv preprint arXiv:1706.02677, 2017.

[30] I. Kandel, and M. Castelli, "The effect of batch size on the generalizability of the convolutional neural networks on a histopathology dataset," ICT Express, vol. 6, pp. 312–315, 2020.

[31] D. Masters,and C. Luschi, "Revisiting Small Batch Training for Deep Neural Networks," arXiv preprint arXiv:1804.07612, 2018.