Deep Neural Network Discipline and Consequence to be Achieved through Internet Implementation Dropdown

Haitham S. Hasan
Business Information Technology Department, Business Informatics College, University of Information Technology and Communications, Baghdad, Iraq
Email: Haitham@uoitc.edu.iq

Abstract

The difficulty of automatically modifying and updating operations within Deep Learning (DL) frameworks can slow down the performance of Deep Neural Network processing (DNNs). This research presents a novel approach to software optimization by leveraging dynamically collected profile data. A unique online auto-tuning system for DNNs was developed to enhance both the training and inference phases. Python Distributed Training of Neural Networks (PyDTNN) is a lightweight toolkit designed for distributed DNN training and estimation. It is utilized to evaluate the VGG19 model on two distinct multi-core architecture options. In testing, our auto-tuning system performs comparably, if not better, than a static selection strategy. The performance of each variation of PyDTNN that employs static selection remains consistently high throughout execution. Conversely, the auto-tuned version initially performs at a set level and progressively improves as more feasible choices become available. While both variations yield similar results in training, the selection strategy outperforms all other inference options by autonomously determining the best strategy for each layer in VGG19. The new online implementation selection tool assists in choosing the best performance option from numerous alternatives while the program is running. Its key features include constructing layered judgments and thoroughly examining 35 possibilities. Our advanced systems represent the optimal choice for monitoring sustainable environmental systems with maximum effectiveness, efficiency, and timeliness.

Keywords: DNNs; auto-tuning; implementation selector; Artificial intelligence; and sustainable development

1. Introduction

The development of new algorithmic approaches and the availability of large quantities of computing power have led to an increase in the application of artificial intelligence, particularly deep neural networks (DNNs), in recent years. Consequently, organizations like Google and Facebook have developed deep learning frameworks such as TensorFlow and PyTorch, tailored to their specific requirements. Optimizing and customizing deep learning frameworks for specialized systems is crucial for reducing training and inference costs. Therefore, selecting the best methods for DNN processing can significantly reduce the manual effort required to develop optimized versions. Moreover, artificial intelligence also contributes positively to environmental sustainability by addressing issues such as low-carbon transportation, sustainable agriculture, biodiversity conservation, protection of water and energy resources, management of mineral raw materials, and waste and pollution management.

The outcomes of convolutional layer training depend on the algorithmic version and complexity provided. Runtime may be impacted by other operations. It is recommended to adjust DNN layers based on profile data. Additionally, an online implementation picker for deep learning frameworks selects the best implementation at runtime.

We provide an online auto-tuner that leverages past execution data to determine the best technique for each issue size. This tool is accessible to all our users. We will explore its features and how to utilize them, including nested choices, which enable decisions based on implementation options that depend on other alternatives, and grouped options, which facilitate the selection

Doi: https://doi.org/10.54216/FPA.160115

Received: February 27, 2024 Revised: March 28, 2024 Accepted: May 05, 2024

of the best-performing routines when used together. Convolutional neural networks (CNNs) employ specific modules such as the Winograd method and forward-backward passes [6]. Variants of Winograd's convolution algorithm on the same filter size have been integrated into our PyDTNN distributed DNN training and inference framework, which is small and lightweight. The VGG19 CNN model is trained and evaluated on nodes equipped with Intel® Xeon® Scalable Processors from the 5th generation. This paper also includes a per-layer evaluation of performance and throughput increases during training operations.

In a study conducted by S. Pree [7], it was demonstrated that convolutional layers can be trained through forward and backward passes. Today's software libraries utilize kernels to process data types based on the hardware being used. When a user selects a processor from a shortlist, the most appropriate computational kernel is typically chosen automatically. M. Sowjanya [11] has written about an online technique that considers multiple options before deciding on the best one based on the execution time of 18 actual calculations. As mentioned earlier, the procedure and settings must be iteratively repeated before utilizing DNN for training. B. Zheng and L. Zheng [3,4] have developed a runtime selector for various convolution implementations, which is only available in the latest version of CuDNN. The variable used can result in a wide range of outcomes.

The remainder of the essay is organized as follows: Section 2 discusses Online Implementation Selector. Section 3 compares the integration in PYDTNN to native versions. Section 4 discusses the dataset and materials used to evaluate and build the proposed model. Section 5 presents a detailed analysis of performance differences. Section 6 concludes the article by summarizing the key findings and offering final remarks.

2. An Online Implementation Selector

After a predetermined number of rounds, this section describes a Python-based online auto-tuner capable of running various algorithms and ultimately selecting the optimal solution for each issue, such as Type 2 diabetes.

2.1 Algorithm Description

The algorithm dynamically selects the best approach based on prior performance data for different issue sizes. To achieve this, the constructor receives a list of pairs, where each pair indicates which function should be called if a specific option is selected. Even non-compliant functions can be utilized by wrapping them to ensure they have the same input parameters and are called in the same order. For transposing tensor dimensions, Numpy's native transpose function is employed, unlike the two Python solutions that use various loop orders. The auto-Python tuner class includes constructors and auxiliary member methods that can be utilized after the object has been constructed. The call method is responsible for determining the optimal execution time and selecting the best option if required. This process continues until a predefined number of rounds have been completed, and the best approach is selected based on average performance. The nested and auto-grouping tuner features differentiate it from its competitors. To ensure that the auto-tuner works properly, all parameters must be independent.

2.2 Optimization algorithms

To optimize performance, re-use prior function call results. In convolution layers, use the row transform for forward and backward propagation. There is no need to rewrite row translation in the reverse direction since the temp variable is used in the forward operation. Apply the same strategy consistently for speed. For the auto-tuner to function properly, all parameters must be independent. By knowing the outcome of a prior function call, alternatives can perform optimizations. The row transform is a tool that shows how information moves forward and backward in a convolution layer. To make this work, we use a temporary variable. This saves time and effort because we don't have to rewrite the row translation in the opposite direction. This strategy is used throughout to ensure speed and efficiency. To effectively manage the dependencies in a set of algorithms that must be executed simultaneously, we need to implement a robust and well-designed strategy. Each group is monitored, and the extent of the problem is evaluated in the internal system. After a set number of rounds, the highest-performing group is executed. Additionally, the set of this auto-feature tuner has nested settings that can be extended. A decision tree is used to make real-time decisions based on a range of criteria. The traceback Python module creates a decision tree using the stack and current function calls in the code. Stack frames are used to determine if the current object was previously called from another instance.

2.3 The Feature

When optimizing a node, the autotuner can discover the most efficient branching in the tree. Before determining the execution time of its alternatives, each node must first identify the choices made by each of its offspring. This means that evaluations of parents are put on hold until all their children make decisions. Additionally, it provides metrics and decision trees that users can analyze after the event to gain insights into the best-performing solutions for various scenarios.

Doi: https://doi.org/10.54216/FPA.160115

Received: February 27, 2024 Revised: March 28, 2024 Accepted: May 05, 2024

3. Integration in PyDTNN

PyDTNN's computations extensively utilize data parallelism (DP) [12]. DP relies on message-passing libraries for effective communication, while its fast, multi-threaded kernels depend on multi-threaded libraries. Alternative implementations of PyDTNN in Python include utilizing the auto-tuner to determine which procedures use which code. Integration into the network was achieved in PyDTNN. The convolutional 2D layers of PyDTNN already feature algorithms for forward and backward passes with minimal filtering. In this scenario, the grouping function is employed to optimize forward-backward options. Variations on the Winograd technique are utilized for this purpose, resulting in nested selection since Winograd's technique was previously provided as an algorithm option. While Numpy transpose may perform well with small tensors, it may require assistance with larger arrays due to being serialized and memory-bound [17],[18],[19]. PyDTNN offers two concurrent Python transposition implementations, both utilizing OpenMP. In these variants, access to dimensions is provided in the opposite order of the conventional method [20]. Despite TensorFlow and PyTorch being more mature and feature rich, PyDTNN, despite its lack of maturity and complete features, provides a more accessible and configurable solution for quick DNN model training and inference.

4. Materials and methods

4.1. Data set

The data set used in the study consists of white and dark-skinned people, from which the facial image is recognized. Examples of 60 color images appear in the dataset and we summarize them with two images as in Figure (1).

Figure 1: A black-skinned and white-skinned men

The dataset consists of images and their corresponding category information. There are two basic categories in which data is named: "white" and "brown." These categories were used to train and test the model.

4.2. Deep learning, CNN, and transfer learning

Deep learning employs neural networks comprising multiple layers to extract a hierarchy of features from raw input data. This process refers to the learning that occurs within these networks. Unlike classical machine learning, which relies on manual feature extraction, deep learning autonomously learns complex feature hierarchies from data, particularly in image analysis tasks. Convolutional Neural Networks (CNNs) are a new and popular machine learning technique [4]. Convolutional Neural Networks (CNNs) are widely used. It is a deep learning technique used for classification.

4.3 Model CNN convolution and pooling

It consists of pooling, activation, and classification layers. The feature extractor applies filters over the input, enabling the detection of specific features. To enhance study development, data reduction occurs in the data layer by selecting the largest, smallest, or average data from previous layers. The activation function enhances the nonlinearity of input reaching the neuron in the subsequent layer, ensuring stable results. The choice of appropriate activation depends on the type of operation performed within the neural network. The rectified linear unit (ReLU) has gained widespread usage in CNN architectures recently. The entire link layer is pivotal in transitioning to the prediction phase.

4.4 Prediction layer output:

The values are transferred to the loss function and returned according to the loss between the expected value and the actual value. Based on the loss, the weights are updated to achieve the best result. Transfer learning uses the information of a network that has been pre-trained with large amounts of data to solve a similar problem. It is transferred to another form created to solve

Doi: https://doi.org/10.54216/FPA.160115

Received: February 27, 2024 Revised: March 28, 2024 Accepted: May 05, 2024

it. The pre-trained model can be used to successfully classify small amounts of data, especially if transferred information from a large dataset is available. Another advantage of transfer learning is that it reduces the processing load. The calculated weights are transferred to the new model and only the classifiers in the last part of the new model are trained.

4.5. MobileNetV2 architecture:

MobileNetV2 is a convolutional neural network specifically developed to achieve high performance on mobile devices in the field of Network Engineering. In deep neural networks, neurons that are not highly activated during training can be ineffective. In such cases, adjustments are made to the remaining values within the network structure.

MobileNetV2 consists of blocks that feed into subsequent layers. There are two types of blocks in MobileNetV2. The first type is the residual block, which accounts for the deviation in a single step (referred to as a "roadblock"). The other type is a block designed to reduce the value through a two-step offset. Both types of blocks comprise three fundamental layers. The initial layer is a convolutional layer with a 1x1 filter size and ReLU activation. The second layer consists of a deeper convolutional block with a 3x3 filter and ReLU activation. Finally, the last layer is another convolutional layer, providing a 1x1 output with linear activation (see Figure 2).

4.6. MobileNetV3 architecture:

M-Net [3] is a CNN developed for segmentation, particularly for biomedical images. Clouds are also a key component onboard satellites [5]. Its architecture comprises an encoder and a decoder. The encoder's role is to capture context, while the decoder focuses on hash-mask calculation. For efficiency reasons, we replaced the original encoder with MobileNet-v3-small [21], ensuring seamless integration with the new encryption. The decoder has been simplified to include 5 DE convolution blocks followed by a convolutional layer. The sigmoid activation facilitates mask generation, ensuring it matches the input image's dimensions. The first 4 of the 5 decoder blocks consist of a transposition convolution layer, batch normalization layer, and dropout layer, followed by Relu activation. The outputs of these four blocks relate to the outputs of the encoder. Thus, our UNetMobileNetV3 comprises 4,693,553 parameters, including 1,529,968 for the MobileNetv3-small encoder and 3,163,585 for our set-top box.

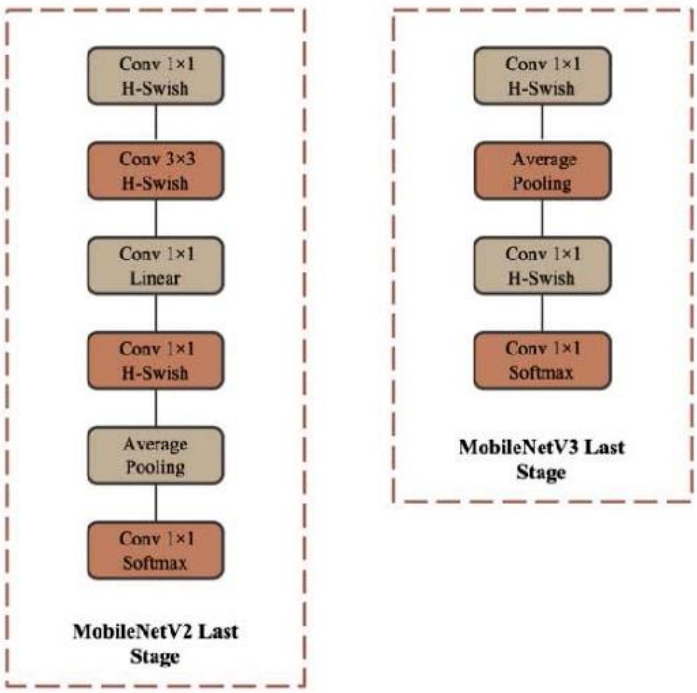

Figure 2: MobileNetV2 and MobileNetV3 Architectures

5. Results and Discussion

Doi: https://doi.org/10.54216/FPA.160115
Received: February 27, 2024 Revised: March 28, 2024 Accepted: May 05, 2024

In this study, 60 photographs of 60 people were captured. The resulting dataset underwent classification using the MobileNetV2 architecture. This model is renowned for its effectiveness on mobile devices, even with constrained resources. It was selected due to its availability and potential applicability to future mobile phone systems. MobileNetV2, based on the ImageNet architecture, has been trained on over 14 million images spanning 1000 categories. The weight values obtained from training on our dataset were then transferred to the developed model. Following this, the classification process was completed by incorporating an independent classification layer.

After transferring the model, mean pooling was applied, followed by the output matrix. These were then normalized and fed into the neural network responsible for classification. The neural network has a density of 64 neurons. Following this layer, there is a 40% dropout. Finally, the dataset consists of 3 neurons, matching the number of classes. It consists of an output layer. ReLU exists in both the first dense layer and the output layer of the neural network. The Softmax activation function, which produces probability values, was utilized. The model can be trained with a total of 72,233 parameters. Since the dataset contains images of various sizes, each image is resized to 600 x 1066 pixels. The dataset was divided into 80% training and 20% testing samples. To reduce the possibility of overfitting during model training, data augmentation techniques were employed to enhance the model's success.

The model underwent 50 epochs, and the success of the classification process was evaluated using various metrics. Among these metrics, accuracy is the most used measure of success in classification. It is defined as the ratio of correctly classified samples to the total number of data points, as shown in Equation 1. In this study, the classification accuracy achieved after 50 epochs was 97.55%. The accuracy and loss variation graphs of the model are depicted in Figure 3.

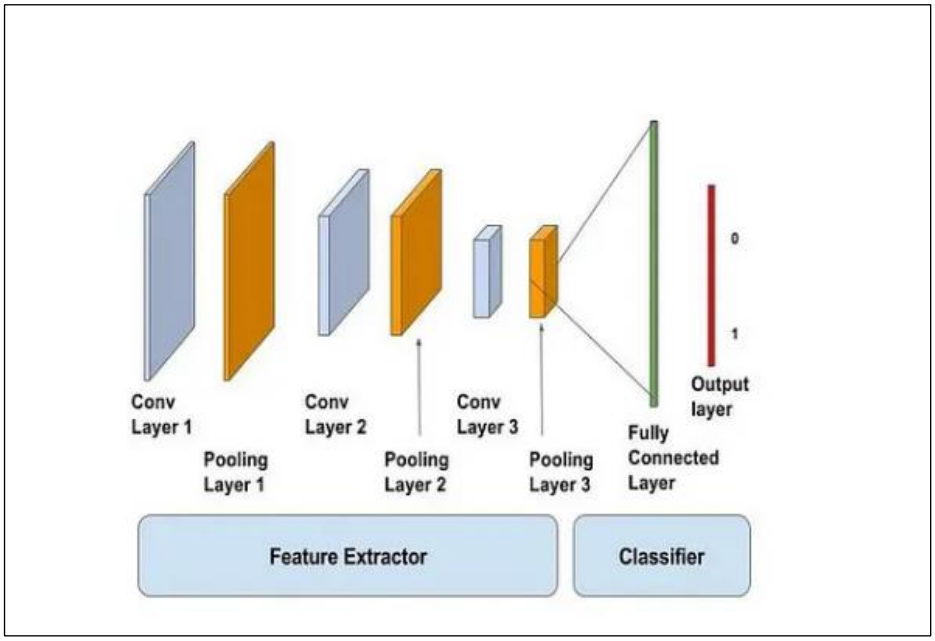

Figure 3: Model accuracy and error graphs

Classification accuracy, on the other hand, provides an overview of the model's performance, it is beneficial to deeply investigate misclassifications and assess them using additional performance measures. For this purpose, confusion matrices are used (Figure 4).

Doi: https://doi.org/10.54216/FPA.160115

Received: February 27, 2024 Revised: March 28, 2024 Accepted: May 05, 2024

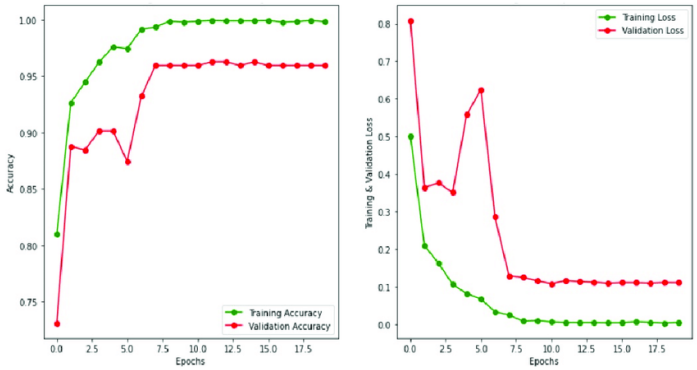

Figure 4: Model accuracy and error graphs

If the positively labeled data in a dataset is correctly classified as positive by the model, it will be labeled as true positive (TP). Conversely, if positively labeled data is misclassified as negative, it will result in a false negative (FN) error. Similarly, when negatively labeled data in a dataset is correctly classified as negative, it is labeled as true negative (TN). Conversely, if negatively labeled data is incorrectly classified as positive, it is described as a false positive (FP). Different performance metrics are calculated based on the parameters obtained from the complexity matrix. One of these metrics, sensitivity (also known as recall), is the ratio of true positives to the sum of true positives and false negatives, as shown in Equation 1.
Different performance = T+/ (T+) +(F-)……. Equation 1

Sensitivity refers to the ability of the model to distinguish between true positives. The classification done in this study
The sensitivity value obtained for this procedure is 96%.
Another performance metric, accuracy, is the number of true positives, true positives, and false positives.
It is the ratio of the total numbers (Equation 2).

Sensitivity = T+/ (T+) +(F+)……. Equation 2

Accuracy refers to the ability of the model to eliminate false positives. In this study, the classification resulted in an accuracy value of 97%. The F1 score, representing the harmonic mean of sensitivity and accuracy, provides insight into the balance between these two metrics. It is calculated using the formula presented in Equation 3. Score= 2* Different performance* Sensitivity/ Different performance* Sensitivity. The F1 score value obtained because of the classification process conducted in this study is 94.33%.

5.1 Inference Results for Machine Learning Model

The results of various convolutional methods and an alternative have been shown in Figure 3 for configurations with one to sixteen threads. It has been observed that the most effective inference method differs depending on the dataset, the number of threads, and the node design, which contradicts the observations made during training. Winograd's algorithm works best for VGG19 and 16 threads on Altec, while ConvGEMM outperforms Winograd on Volta but beats Winograd on Altec when VGG19 and 16 threads are used. In addition, ConvGEMM not only outperforms all other algorithms in every case, but it also outperforms the best approach. As a result, not all VGG19 layers employ the same algorithm. Figure 4 displays the average time for each VGG19 convolutional layer with 16 threads forward approaches, and the best option is shown.

Doi: https://doi.org/10.54216/FPA.160115

Received: February 27, 2024 Revised: March 28, 2024 Accepted: May 05, 2024

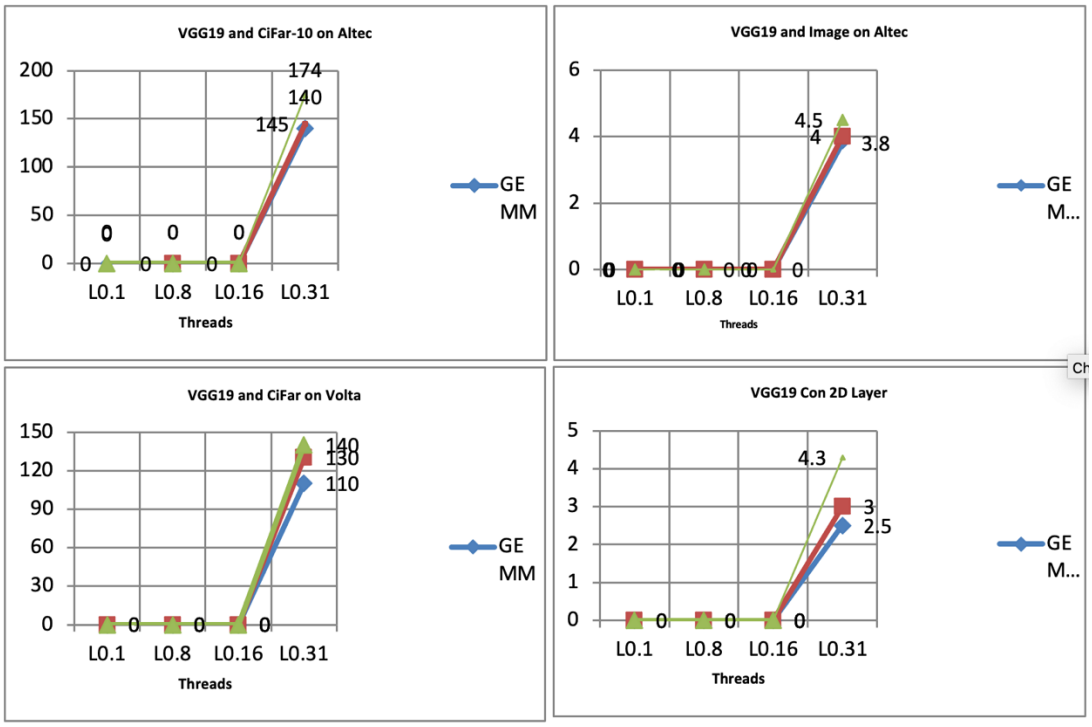

Figure 5: Many forward-backward options for training on CIFAR-10 or ImageNet with VGG19, as well as the same options for Altec and Volta (top and bottom, respectively).

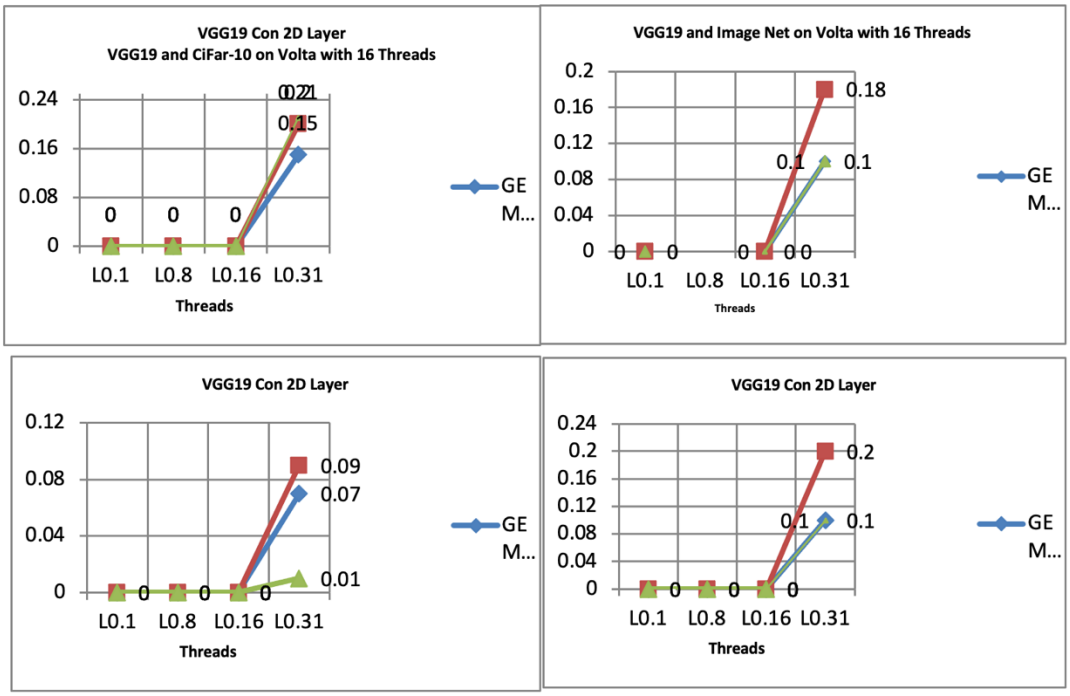

Figure 6: Both Altec (top) and Volta (bottom) were utilized to train the VGG19 convolution layer, and 16 threads were utilized on average to train on CIFAR-10 and ImageNet

Doi: https://doi.org/10.54216/FPA.160115
Received: February 27, 2024 Revised: March 28, 2024 Accepted: May 05, 2024

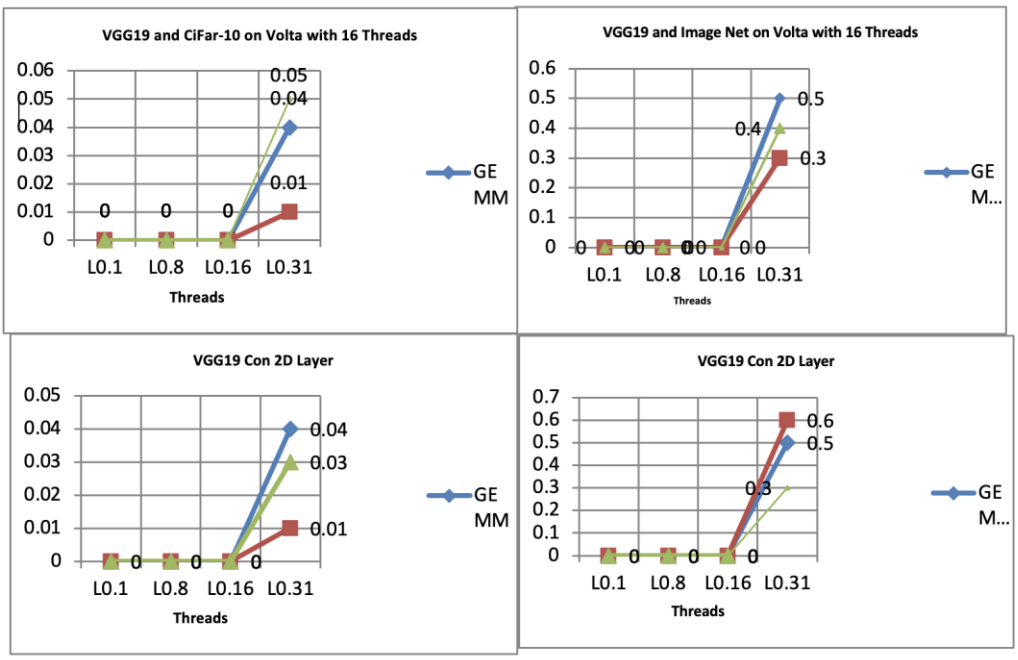

Figure 7: The outcomes of Volta-based VGG19 inference on CIFAR-10 and ImageNet, with several forward-backward options and the same characteristics as before (on the left and right, respectively).

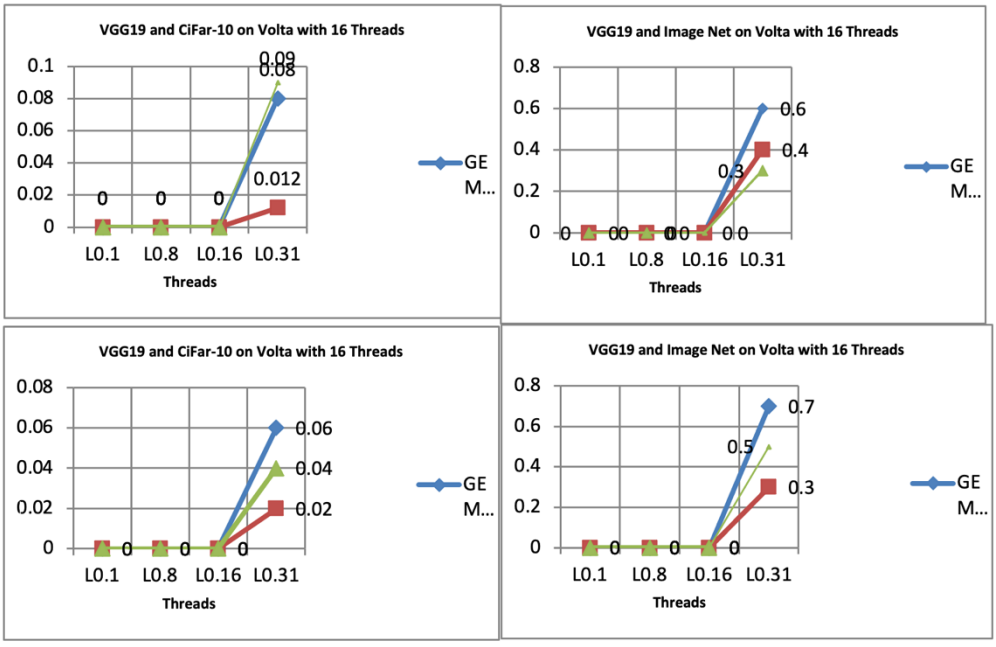

Figure 8: To test the forward-backward choices explored by the VGG19 convolutional layers, Altec and Volta with 16 threads were utilized to assume about CIFAR-10 and ImageNet.

6. Conclusions

Facial recognition technology is of great importance, especially in security fields. The use of computer-assisted automatic image analysis systems is also a radical solution in this study. It was revealed that using layered filtering technology supported by deep neural networks would achieve an accuracy of 97.88%. The MobilNetV2 and MobilNetV3 models used to develop the classification model also provide high performance in mobile systems. It also has an architecture that can provide high

Doi: https://doi.org/10.54216/FPA.160115

Received: February 27, 2024 Revised: March 28, 2024 Accepted: May 05, 2024

accuracy in face recognition, and in this sense, the study was conducted on mobile control systems that could be successfully operated and converted into a portable solution. In our latest work, we introduce "BestOf", which is an online implementation selector that can choose the best-performing alternative among different options at runtime. BestOf offers essential features that enable the evaluation of multiple alternatives and the ability to make nested decisions. Extensive experiments conducted by our team using the VGG19 model have proven that our auto-tuner yields notable enhancements in both training and inference times when processing convolutional layers with various algorithms. Additionally, we have discovered that BestOf can easily identify the most suitable option. It's imperative to note that the efficiency and accuracy of the chosen algorithm remain uncompromised, irrespective of the target architecture. This guarantees optimal performance that meets your requirements. This eliminates the need for manual assessment of the available choices. Therefore, we can confidently state that the BestOf auto-tuner offers significant benefits that far surpass any negligible overhead costs and add lines of code to the primary application. Our newly developed online tool helps in selecting the best implementation method that can produce the highest performance outcomes from a wide range of alternatives while running the program. This tool can construct layered judgments and can thoroughly examine 35 different categories of options, making it a valuable resource for decision-making.

References

[1] V. Sze, Y.-H. Chen, T.-J. Yang, and J. S. Emer, "Efficient processing of deep neural networks: A tutorial and survey," Proc. IEEE, vol. 105, no. 12, pp. 2295–2329, 2017.

[2] S. Pouyanfar et al., "A survey on deep learning: Algorithms, techniques, and applications," ACM Comput. Surv., vol. 51, no. 5, pp. 92:1–92:36, Sep. 2018.

[3] B.Zheng, Z. Jiang, C. Hao , H. Shen, J. Fromm, Y. Liu, Y. Wang, L. Ceze, T. Chen, G., "DietCode: Automatic Optimization for Dynamic Tensor Programs Part of Proceedings of Machine Learning and Systems 4 (MLSys 2022)

[4] L. Zheng, C. Jia, M. Sun, Z. Wu, C. H. Yu, A. Haj-Ali, Y. Wang, J. Yang, D. Zhuo, 19 K. Sen et al., "Ansor: Generating high-performance tensor programs for deep learning," in{OSDI}1420)th {, 2020, pp. 863–879.USENIX} Symposium on Operating Systems Design and Implementation

[5] S. Barrachina, A. Castell´o, M. Catal´an, M. F. Dolz, and J. I. Mestre, "Pydtnn: A(22 user-friendly and extensible framework for distributed deep learning," The Journal of 23 Supercomputing, pp. 1–17, 2021.

[6] 6. P. S. Juan, A. Castell´o, M. F. Dolz, P. Alonso-Jord´a, and E. S. Quintana-Ort´ı, "High performance and portable convolution operators for multicore processors," inIEEE International Symposium on Computer Architecture and High Performance32nd Computing, SBAC-PAD 2020, Porto, Portugal, September 9-11, 2020. IEEE, 2020, pp. 91–98. [Online]. Available: https://doi.org/10.1109/SBAC-PAD49847.2020.00023

[7] S. Preet, MK Sharma, J Mathur "Analytical model of semi-transparent photovoltaic double-skin façade system (STPV-DSF) for natural and forced ventilation modes ,Journal of Ventilation, 2023, https://doi.org/10.1080/14733315.2021.1971873

[8] R. C. Whaley and J. J. Dongarra, "Automatically tuned linear algebra software," in 31 Proceedings of the 1998 ACM/IEEE Conference on Supercomputing, ser. SC '98. USA: 32 IEEE Computer Society, 1998, p. 1–27.

[9] J. Fern´andez, A. S. Cuadrado, D. del Rio Astorga, M. F. Dolz, and J. Daniel Garc´ıa, "Probabilistic-based selection of alternate implementations for heterogeneous plat-forms," in Algorithms and Architectures for Parallel Processing, S. Ibrahim, K.-K. R. 35 Choo, Z. Yan, and W. Pedrycz, Eds. Cham: Springer International Publishing, 2017, pp. 749–758.

[10] J. Planas, R. M. Badia, E. Ayguad´e, and J. Labarta, "Self-adaptive ompss tasks in heterogeneous environments," inand Distributed Processing, 2013, pp. 138–149.2013 IEEE 27th International Symposium on Parallel 39 11. M. Jorda`, P. Valero-Lara, and A. J. Pen˜a, "Performance evaluation of cudnn convolution algorithms on nvidia volta gpus," IEEE Access, vol. 7, pp. 70461–70473, 2019. 41 12. T. Ben-Nun and T. Hoefler, "Demystifying parallel and distributed deep learning: An 42 in-depth concurrency analysis," CoRR, vol. abs/1802.09941, 2018. [Online]. Available:

[11] M Sowjanya, L Devi, "Mounting-based knowledge transfer network Model Using Aspect-based sentiment analysis. (2023) https://doi.org/10.21203/rs.3.rs-2970874/v1.This work is licensed under a CC BY 4.0 License

[12] Winograd, Arithmetic Complexity of Computations Society for Industrial and Applied Mathematics, 1980.

[13] A. Castell´o, M. F. Dolz, and E. S. Quintana-Ort´ı, "Towards portable realizations of 46 winograd-based convolution with vector intrinsics and openmp," in 30th EUROMICRO 47 Workshop on Parallel, Distributed and Networked Processing PDP 2022appear., 2022, p. To

[14] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," 2015.

[15] 16. H Hasan., Abdul Kareem S (2013) Fingerprint image enhancement and recognition algorithms: a survey. Neural Comput Appl 23:606–1608, https://doi.org/10.1007/s00521-012-1113-0

[16] S. Haitham Hasan Mais A Al-Sharqi (2021) Hand vein recognition with rotation feature matching based on fuzzy algorithm International Journal of Nonlinear Analysis and Applications (IJNAA), doi: 10.22075/IJNAA.2021.5536

[17] Urvashi Gupta , Rohit Sharma, Multi-sensor Data Fusion based Medical Data Classification Model using Gorilla Troops Optimization with Deep Learning, Fusion: Practice and Applications, Doi: https://doi.org/10.54216/FPA.150101, Vol. 15

Received: February 27, 2024 Revised: March 28, 2024 Accepted: May 05, 2024

Issue. 1 PP. 08-09, (2024)
[18] Dilobar Isomjonovna Ruzieva, The Fusion of Digital Technologies in Small Business for Ensuring the Socio-Economic Development: Panel Data Analysis, Fusion: Practice and Applications, Doi: https://doi.org/10.54216/FPA.150106, Vol. 15 Issue. 1 PP. 66-67, (2024)
[19] Priyanka Dhaka , Ruchi Sehrawat, Adaptive Ensembled Fusion Based Deep CNN-Bilstm Model For Heart Disease Prediction In IoT, Fusion: Practice and Applications, Doi: https://doi.org/10.54216/FPA.140104, Vol. 14 Issue. 1 PP. 40-41, (2024)>
[20] Anita Madona M. , Paneer Arokiaraj S. Effectual Augmentation of Glaucoma Prediction in Retinal Fundus Images using Hybrid Level Fusion of Image Pre-Processing Techniques, Fusion: Practice and Applications, Doi: https://doi.org/10.54216/FPA.140108,Vol. 14 Issue. 1 PP. 93-104, (2024)>

Received: February 27, 2024 Revised: March 28, 2024 Accepted: May 05, 2024