Al-Rafidain Engineering Journal (AREJ) Vol.27, No.2, September 2022,pp.198-205

U-Net Cost Analysis Using Roofline Model

Ula T. Salim ula.tariq@uomosul.edu.iq

Shefa A. Dawwd shefa.dawwd@uomosul.edu.iq

Fakhrulddin H. Ali fhazaa@uomosul.edu.iq

Computer Engineering Department, Collage of Engineering, University of Mosul, Mosul, Iraq

Received: 10/5/2022 Accepted: 4/6/2022

ABSTRACT

One of the most important challenges facing U-Net architecture performance is the method design of its components and how to choose the suitable hardware computing device to deal with the training labelled datasets. Convolution is the most process that requires computations and memory costs, which is needs to minimize. Thus, one of the suitable selection is to change the type of the convolution. Other suggested solutions are to reduce the size of image, number of bits, and, stride value, in addition to number of filters, and image batches. Therefore, in this paper the roofline model will used as performance guide in analyzing the FLOPs and the memory bandwidth boundaries of a U-Net model with different configurations. The cost has been assessed with compared to the limitation of three computing devices, GPU230MX, GPU940MX and GPU2060rtx super. 128 × 128 image dataset has been used during the U-Net cost-performance evaluation process. Based on the analysis, the evaluation results show that the solution that achieves a balance between memory and computations is to implement a U-Net model in parallel using RTX2060 super card with the configurations of batch size is 16, image size of 128×128, number of bits is 32, shared memory management.

Keywords:

Roofline model; U-Net; FLOPs; Memory bandwidth; Arithmetic intensity.

===

1. INTRODUCTION

The superiority of using deep learning models to achieve good and accurate estimation in one application is related to the size of the data and depends on the type of algorithm/model. However, many factors limit the performance of the algorithm implementation, thus implementation details must be taken into consideration, especially when the hardware used has limited resources. The algorithm is related with how to implement it and optimizing its components with least complexity. The implementation means running an algorithm with the suitable high performance computing (HPC). The floating-point throughput for the HPC, as well as the memory bandwidth of the DRAM chip, which determines the performance cost and depends on the manufacturing specification. However, it may not give a visualization of what the user can achieve.

U-Net architecture[1] as a convolutional neural network (CNN) is one example that has been used in many applications with different configurations[2]. The basic structure of U-Net is composed of several types of layers with different operations, but still the convolution operation is the engine operation. However, the complex U-Net based CNN design and large amount of datasets with its labels will enhance the accuracy but it is demands memory storage space, while the reverse will be happen with small model size. So implementing U-Net variants on general-purpose processors (CPUs) may be inappropriate due to performance bottlenecks resulting from their lack of parallelism. Today, many hardware options give the users flexibility when chosen for training and testing. These options include GPUs [3][4], TPUs[5][6], FPGAs [7][8], heterogeneous[9] and servers[10].

There are another solutions but it that depend on user's efforts and proposals. Different frameworks based parallelism strategies are introduced to tackle memory limitation such as TensorFlow Large Model Support (TFLMS) and Mesh-TensorFlow[5][11].

Other attempts are based on reducing the number of bits and quantizing the model size. A helpful solution is changing the type of convolution such as employing depthwise separable convolution as an alternative to the traditional convolution [12]. Another suggestion is performing the convolution process with stride value more than 1 [13].

Roofline model is one useful tool that is used in several works and studies and applied on different architectures[14][15][16][17][18] [19].

In this paper, a performance cost of U-Net architecture modeled with different parameters using a helpful roofline model. A Roofline model is employed as an efficient analyzing approach to alleviating the errors before developing the network models. It is can result an appropriate balance between different network configurations and selecting the suitable hardware environment.

In addition to this introduction, the rest sections of this paper is divided as follows: section 2 explains the Roofline models, section 3 describes the details of the U-Net structure, section 4 analyzes the U-Net cost with different configurations. Finally, section 5 summarizes the paper with the most conclusions and presents suitable future work.

2. ROOFLINE MODEL

Roofline model is a theoretical pre-programing step used to avoid errors that may appears during runtime execution such as out of memory (OOM). It is an abstract architectural model, which determines the performance throughput edges using each of the peak performance and the peak bandwidth. This model is usually used visually as a logarithmic plot of peak performance versus arithmetic intensity (AI) as illustrates in figure (1)[14].

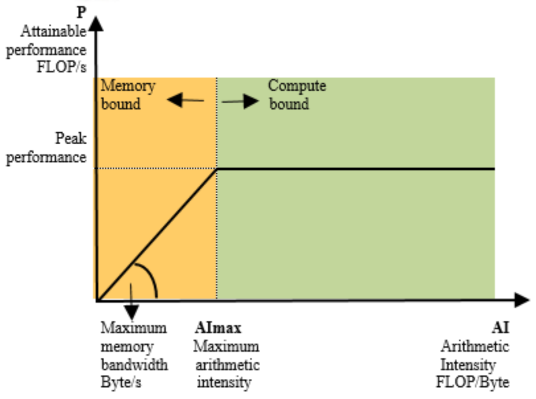

Fig. 1 Basic representation of roofline model

The Peak performance is the number of arithmetic operations a computer performs per second on a given algorithm and expressed as an floating point operations per second (FLOPs/s). The flops/s is a useful measurement because the higher it is, the faster the algorithm can execute more data. The FLOPs/s is often related with the type of hardware, algorithm, and implementation. On the other hand, the Peak Bandwidth represents the fastest the processor can load data. It is measured in bytes/second.

Arithmetic intensity (AI) is a measure of the number of operations performed per byte loaded or stored from memory.

$$\mathrm{AI} = \frac{\mathrm{FLOPs}}{\text{Read bytes} + \text{Write bytes}} \qquad (1)$$

For any platform, the AI referred to as AImax and its calculation depends on its limitations.

Roofline curves help you better understand how one application works on a given architecture.

Therefore, the maximum theoretical performance of an algorithm/model using roofline is determined according to equation (2)

$$\mathrm{P} = \min(\text{peak performance}, \mathrm{AI} \times \text{peak bandwidth}) \qquad (2)$$

When the AI of an application is greater than AImax, then the maximum theoretical performance P that an application can achieve is limited by computational throughput(FLOPs/s) , on the contrary, the maximum performance P is a memory bound and the P will be equal to the multiplication of AI by peak bandwidth. The best use of the platform resources can be achieved at ridge point when the P is equal to AImax.

3. U-NET MODEL

In this section, typical forward pass of U-Net model is implemented. The U-Net model consists of encoder and decoder, which are connected via short- long connections constructing a U-shaped as shown in figure (2).

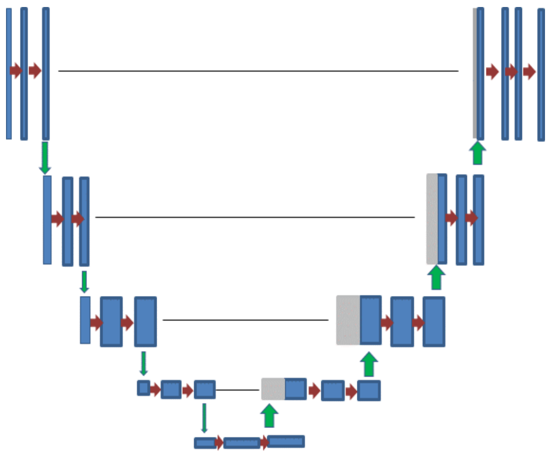

Fig. 2 2D U-Net architecture model

Both of the encoder and decoder have repeated levels of blocks, where each block includes two alternative 3x3 same convolutions and activation function of a type of a rectified linear unit (ReLU). In each level of the encoder, the spatial data is decreased by two using 2x2 max-pool, while the number of filters are doubled by two, the reverse happens with the decoder, where the spatial data increases using nearest algorithm and the number of filters halves by two. The final layer applied a 1x1 convolution layer and a sigmoid activation function on each one of 64-component feature vector through mapping to a four classes.

4. PREDICTION RESULTS

This section predicts the computational performance and memory bottlenecks of the U-Net model. U-Net is configured with 32-filters as starting point and works with an image size of 128x128 to predict four class.

The cost is measured and analyzed in compared to the three graphic processors integrated on three types of machines specified as in table 1.

Table 1: Compute devices specifications

Computing name	GeForce 230MX	GeForce 940MX	GeForce RTX 2060 super
Architecture	Pascal	Maxwell	Turing
CUDA Driver Version	11.1	10.2	11.2
CUDA Capability	6.1	5.0	7.5
# Multiprocessors	2	3	34
# CUDA Cores	256	384	2176
Clock frequency	1.531	1.242GHz	1.68GHz
Memory bandwidth	56.08 GB/s	14.4 GB/s	448 GB/s
Single Precision FLOP/s	784 GFLOS	953.472 GFLOPS	7.181 TFLOPS
Arithmetic intensity	13.98	66.2	16

The FLOPs and total memory access among different layer types is analyzed as shown in figures (3) and (4). One of the observations that needs to be taken into account that the convolution layer takes the largest FLOPs and memory complexities compared with the other operations. This due to addition and multiplication with large number of parameters. On the other hand, log scale is used to reduce the large disparity between the convolution and the cost of the rest computational layers.

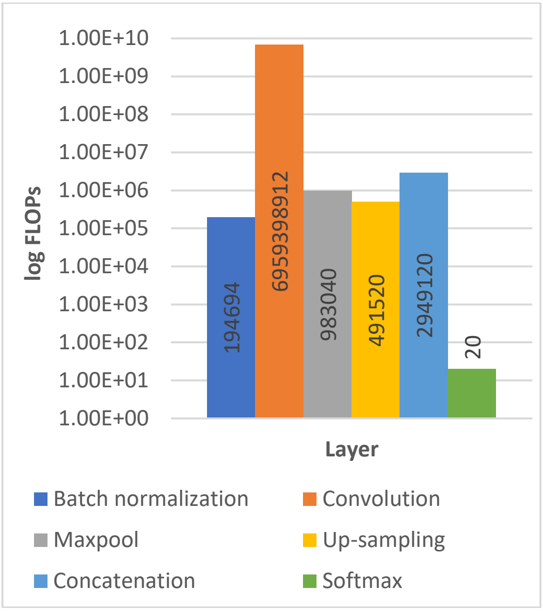

Fig. 3. The FLOPs of U-Net

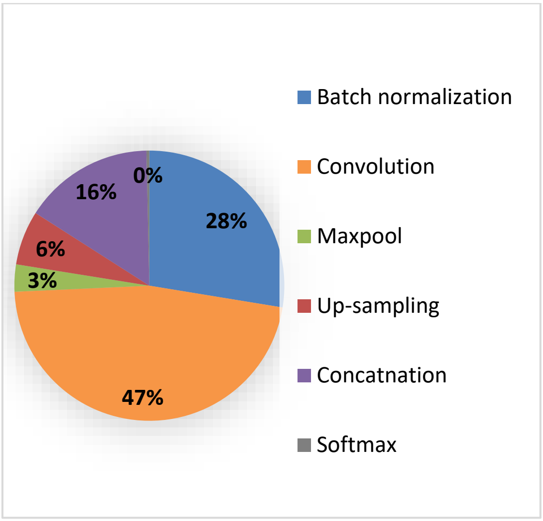

Fig. 4. The number of memory access in U-Net

Figure (5) shows AI per layer compared with the used graphic cards. All smaller AI will be close to the memory bound while the reverse makes the layers will be computed bound. The smaller layer values will be neglected when estimation the computation of designing models. The observed issue is found at the convolution layer with input-output(128×128×32).

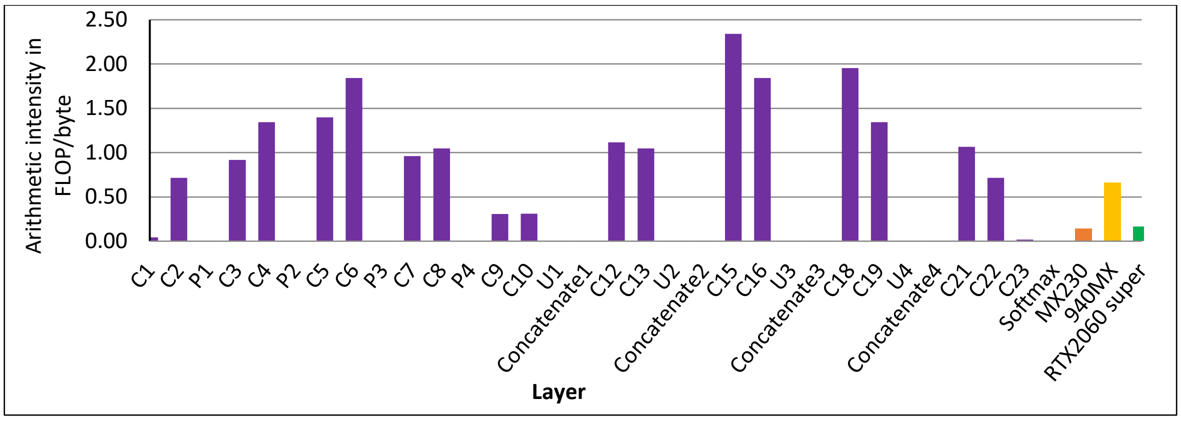

Fig. 5. The arithmetic intensity for each layer of the U-Net

As evident in the roofline model the arithmetic intensity is a significant factor to be consider. However, the complex U-Net will enhance the accuracy but it is demands memory storage space, while the reverse will be happen with small model size. Therefore, for an image size of 128×128 and the starting number of filters is 32, some estimations will be carry out with some tuning and solutions to tackle the computation and memory limitation as follows:

4.1. Impact of convolution type

Convolution is the essence linear operator in the convolutional layer which extracts the important features of input data channels by repeated sliding the learnable filter of stride number over an input data then applying element-wise multiplication-accumulation outcome with corresponding window of input data and generate neurons that construct an output feature maps as illustrated in figure (6).

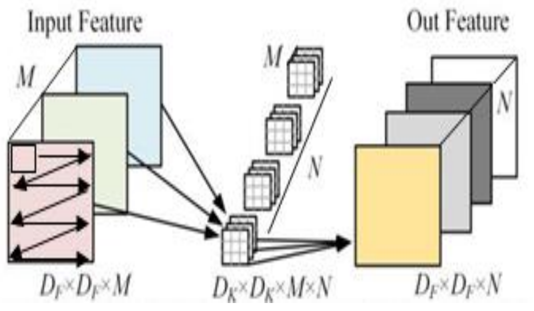
Input Feature
Out Feature
M
M
N
N
$D_F{\times}D_F{\times}M$
$D_K{\times}D_K{\times}M{\times}N$
$D_F{\times}D_F{\times}N$

Fig. 6 Standard convolution (SC) application in convolutional layer

To note the performance(AI) per one layer with change convolution type, the most important types of convolutions[20] listed in table (2).

Table 2: The arithmetic intensity of various structures of convolution

Convolution	Arithmetic intensity
Standard	$\dfrac{BMND_K{}^2D_F{}^2}{BD_F{}^2(M+N)+D_K{}^2MN}$
Pointwise	$\dfrac{BMND_F{}^2}{BD_F{}^2(M+N)+MN}$
Depthwise	$\dfrac{BMD_K{}^2D_F{}^2}{2BMD_F{}^2+D_K{}^2M}$
Separable depthwise	$\dfrac{BMND_F{}^2}{BD_F{}^2(M+N)+MN}+\dfrac{BMD_K{}^2D_F{}^2}{2BMD_F{}^2+D_K{}^2M}$

Figure(7) shows the behavior of the arithmetic intensity resulted for an input layer(128×128×32) and output layer(128×128×32) with a batch size is 1. Evidently, the (AI) is increases as got close to pointwise convolution. Since, the performance will be dominated by memory bandwidth rather than throughput computation according to a table (2).

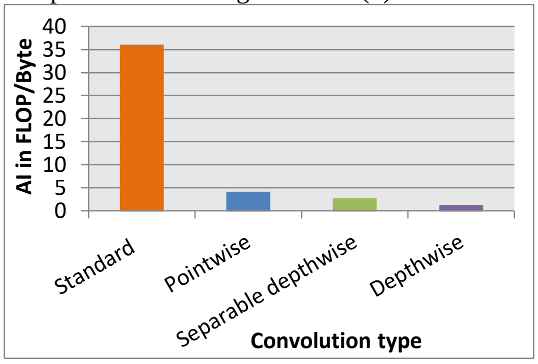

Fig. 7 Impact of convolution type on the performance(arithmetic intensity)

4.2. Impact of stride number

Figure (8) illustrates the accumulative arithmetic intensity when adjusting the step size(stride) for the second convolution layer by 2 and named the new network lightweight U-Net(LWU-Net). It is clear that the stride summarized the data and as a result led to an increase in memory efficiency.

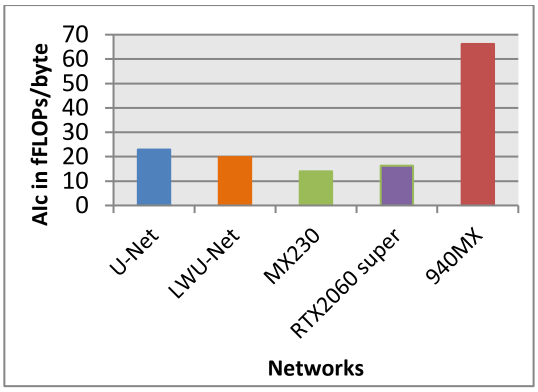

Fig. 8 Impact of adjusting stride number on the performance(accumulative arithmetic intensity).

The Roofline model mapping suggested networks only on the NVIDIA's graphic cards of MX230, 940MX and RTX2060 SUPER and generating three results.

The first result in figure (9) indicates that the networks are more bounded by the computing performance of the MX230 GPU processor architecture.

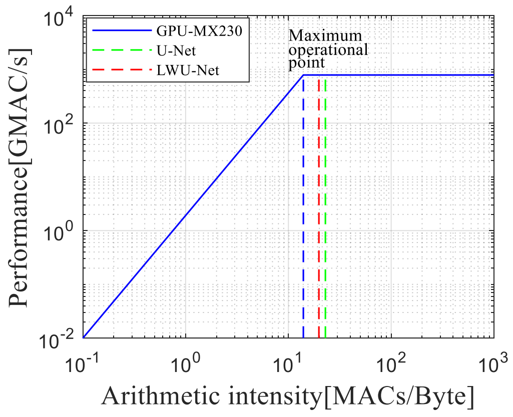

Fig. 9 Roofline model for GPU MX230 super and accumulative arithmetic intensity for the suggested models with batch size 1.

The second results in the figure(10), show that the networks lies within the memory bound region. This indicates that memory bandwidth is the crucial bottleneck for the proposed learning applications running on the 940MX processor.

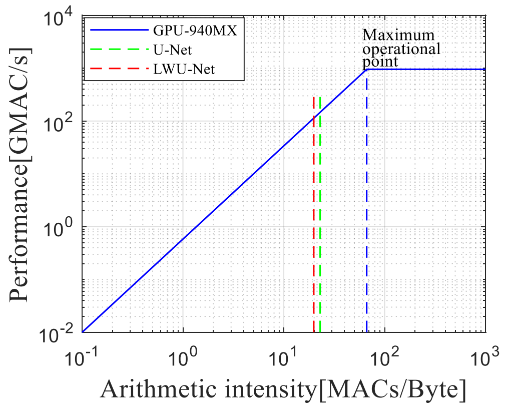

Fig. 10 Roofline model for GPU 940MX super and accumulative arithmetic intensity for the suggested models with batch size 1.

The third result in the figure(11) show that the networks are computed bound.

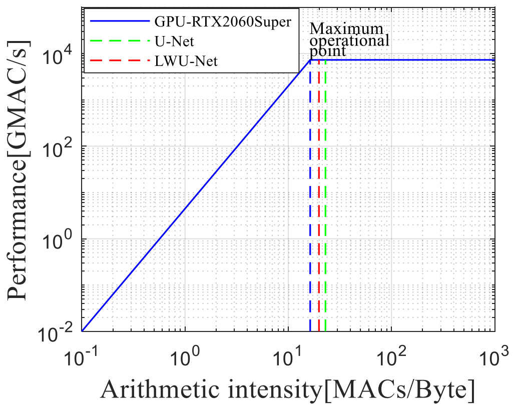

Fig. 11 Roofline model for GPU RTX2060 super and accumulative arithmetic intensity for the suggested models with batch size 1.

Therefore, the 940MX can't be used. It is better to choose the machine that uses the RTX2060 super processor, as it is superior to the MX230 processor in terms of specifications and performance, as shown in the table (1)

4.3. Impact of batch size

Figure(12) shows the roofline model based on the accumulative arithmetic intensity when adjusting the batch size from 1 to 16. It can be seen that the larger the batch size, the higher the accumulative arithmetic intensity by a small percentage, thus avoiding the memory limitation problem. Another note, that the effect of changing the network structure on the accumulative arithmetic intensity is balanced as a result of changing both the number of computations as well as the number of times of memory access. In this state, only the 940MX GPU will be the best choice for implementing all the networks at all the batch sizes, but at the cost of taking a longer time. Thus, to reduce the consuming time, it is better to implement networks in parallel and with a 16-batches size.

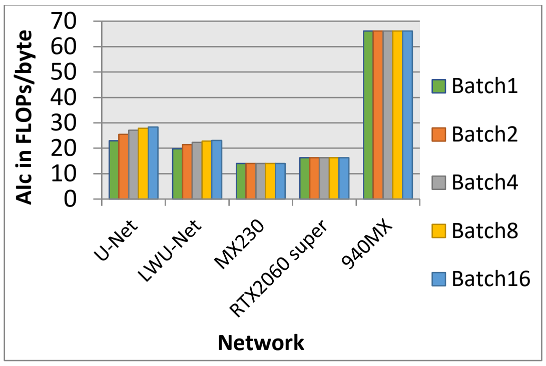

Fig. 12 Impact of adjusting batch size in the 2D Convolution kernel on the performance (accumulative arithmetic intensity).

4.4. Impact of number of bits

Figure(13) shows the roofline model based on the accumulative arithmetic intensity when changing the number of bits from 8 to 64 by a factor of 2.

It is noticed that the higher the bits, the lower the accumulative arithmetic intensity and therefore it will approach the memory limits and therefore not all processors will be able to deal with the data bit size of 64. So, it is better to use the number of bits as 16 so that all networks can be implemented with all networks but at the expense of accuracy. In order to increase the accuracy and in a suitable time, a 32-bits can be used with the networks.

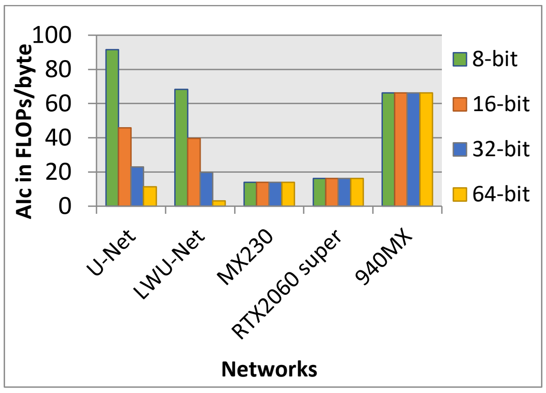

Fig. 13 Impact of adjusting number of bits on the performance(accumulative arithmetic intensity)

4.5. Impact of image size

Figure(14) shows the roofline model based on the accumulative arithmetic intensity when increasing the image size from 128×128 to 192×192 . It is noticed that the higher the image size, the lower the accumulative arithmetic intensity and therefore it will approach the memory limits and therefore not all processors will be able to deal with the image size of 192p. So, it is better to use an image of size 128p but the accuracy of all the networks may be affected depending on the network structure. To implement the networks with fast time, a RTX2060 super is the best processor due to containing on 2176 cores.

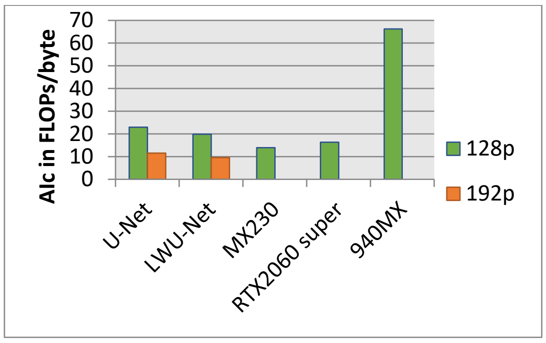

Fig. 14 Impact of adjusting image size on the performance(accumulative arithmetic intensity)

4.6. Impact of CUDA memory management

Understanding the memory hierarchical properties provided by CUDA architecture helps in optimizing GPU kernels and thus alleviating the problem of limited memory resources. As shown in figure(15) CUDA architecture contains different classes of memories that are global, constant, shared, registers and local. Global memory is the simplest type of memory available in a GPU. This is the memory that the host usually accesses when transferring data to the device. Global memory provides the maximum storage size on a GPU approximately a few gigabytes, but it has the disadvantage of slow read and write operations (situated off-chip away from streaming multiprocessors).

Constant memory is the fastest type of memory. It is called "constant" because writting to it is done only by host code. Therefore, it is useful when the kernel needs to access read-only data. The size of this type of memory is much smaller than the size of the global memory, only 64K [1]. In spite of it is yet off the chip, but so they can be accessed much faster than global memory due to they are cached on the chip.

A small portion of shared memory(48-163 KB) is allocated to each thread block that can be read and written by that block alone. Because the shared memory is found on the chip, it is much faster to access, which makes it useful for storing intermediate values in the kernel or data that needs to be accessed repeatly.

The GPU architecture as well contains registers and local memory that are unique to each thread. Registers provide storage for variables or arrays defined by threads in the kernel, and are the fastest to access, but in a limited amount. Registers provide storage for variables or arrays defined by threads in the kernel, and are the fastest to access, but in a limited amount. The CUDA compiler specifies what data is put into registers and what data is passed to local memory. It is called local because of its domain, not its location. Local memory is off-chip, but it's local to each thread because it can only be accessed by specific threads, but it takes longer to get to it[21].

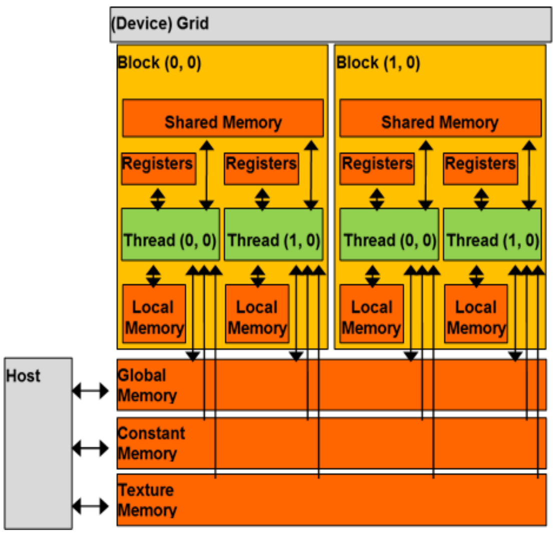

Fig. 15 Memory Hierarchy in the CUDA

The type of memory is affect on the implementation[22], Figure (16) compares between Global, constant and shared as the most memory types for an input layer(128×128×32) and output layer(128×128×32) with a batch size is 1. The results shows that the arithmetic intensity increases when moving towards memory, as well as improving memory management and moving away from the problem of limited memory.

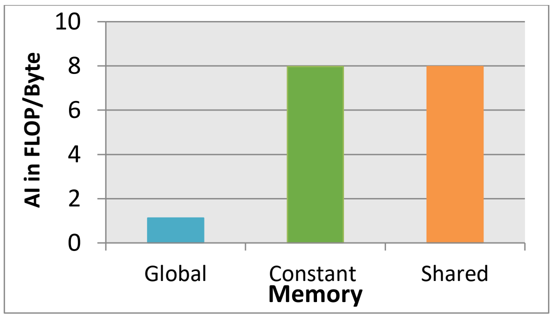

Fig. 16 Impact of memory type on the performance(arithmetic intensity)

5. CONCLUSION

Accurate U-Net performance is a reflection of the complexity design and the training data size with labels. Thus it is necessary to use compatible platform for the implementing work, consequently datasets distribution and batch size are controlling by the performance. In this paper various structures of convolutional layer are estimated. Also, roofline model was exploited as throughput performance model to reduce the time that consumes by the developer in running U-Net architecture. The prediction results shown that RTX2060 super platform is more appropriate selection among the used platforms to achieves well balancing between flops and memory bandwidth. For future work, the roofline model will be extended to take into account another modeling parameters and variables and their effects e.g. on implementation time.

REFERENCES

[1] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics), vol. 9351, pp. 234–241, May 2015, doi: 10.1007/978-3-319-24574-4_28.

[2] U. T. Salim, F. H. Ali, and S. A. Dawwd, "U-Net Convolutional Networks Performance Based on Software-Hardware Cooperation Parameters : A Review," International Journal of Computing and Digital System, vol. 11, no. 1, 2022.

[3] Y. Oyama, N. Maruyama, N. Dryden, E. McCarthy, P.Harrington, , J. Balewski, S. Matsuoka, P. Nugent, and B. Van Essen,"The Case for Strong Scaling in Deep Learning: Training Large 3D CNNs with Hybrid Parallelism," IEEE Trans. Parallel Distrib. Syst., vol. 32, no. 7, pp. 1641–1652, 2021, doi: 10.1109/TPDS.2020.3047974.

[4] D. Pati, C. Favart, P. Bahl, V. Soni, Y. C. Tsai, , M. Potter, J. Guan, X. Dong, and V. R. Saripalli, "Impact of Inference Accelerators on hardware selection," pp. 1–5, 2019, [Online]. Available: http://arxiv.org/abs/1910.03060.

[5] L. Hou, Y. Cheng, N. Shazeer, N. Parmar, Y. Li, P. Korfiatis, T. M. Drucker, D. J. Blezek, and X. Song, "High Resolution Medical Image Analysis with Spatial Partitioning," pp. 15–19.

[6] J. Civit-masot, F. Luna-perejón, S. Vicente-díaz, J. María, R. Corral, and A. Civit, "TPU Cloud-Based Generalized U-Net for Eye Fundus Image Segmentation," vol. 7, pp. 142379–142387, 2019, doi: 10.1109/ACCESS.2019.2944692.

[7] S. Liu and W. Luk, "Towards an efficient accelerator for DNN-based remote sensing image segmentation on FPGAs," Proc. - 29th Int. Conf. Field-Programmable Log. Appl. FPL 2019, pp. 187–193, 2019, doi: 10.1109/FPL.2019.00037.

[8] S. Liu, H. Fan, X. Niu, H. Ng, and Y. Chu, "Optimizing CNN-based Segmentation with Deeply Customized Convolutional and Deconvolutional Architectures on FPGA," vol. 1, no. 1, 2018.

[9] B. K. Joardar, N. K. Jayakodi, J. R. Doppa, H. Li, P. P. Pande, and K. Chakrabarty, “GRAMARCH: A GPU-ReRAM based heterogeneous architecture for neural image segmentation,” in 2020 Design, Automation & Test in Europe Conference & Exhibition (DATE), 2020, pp. 228–233.

[10] D. Ojika, B. Patel, G. A. Reina, T. Boyer, C. Martin, and P. Shah, “Addressing the Memory Bottleneck in AI Model Training,” pp. 3–5, 2020, [Online]. Available: http://arxiv.org/abs/2003.08732.

[11] H. Imai, S. Matzek, T. D. Le, and Y. Negishi, “Fast and Accurate 3D Medical Image Segmentation with Data-swapping Method,” pp. 1–13.

[12] B. Niepceron, A. Nait-sidi-moh, F. Grassia, B. Niepceron, A. Nait-sidi-moh, and F. Grassia, “Moving Medical Image Analysis to GPU Embedded Systems : Application to Brain Tumor Segmentation Moving Medical Image Analysis to GPU Embedded Systems : Application to Brain Tumor Segmentation,” 2020, doi: 10.1080/08839514.2020.1787678.

[13] N. Beheshti, “Squeeze U-Net : A Memory and Energy Efficient Image Segmentation Network.”

[14] S. Williams, A. Waterman, and D. Patterson, “Roofline: An insightful visual performance model for multicore architectures,” Commun. ACM, vol. 52, no. 4, pp. 65–76, 2009, doi: 10.1145/1498765.1498785.

[15] B. Da Silva, A. Braeken, E. H. D’Hollander, and A. Touhafi, “Performance and resource modeling for FPGAs using high-level synthesis tools,” Adv. Parallel Comput., vol. 25, pp. 523–531, 2014, doi: 10.3233/978-1-61499-381-0-523.

[16] A. Ilic, F. Pratas, and L. Sousa, “Beyond the roofline: Cache-aware power and energy-efficiency modeling for multi-cores,” IEEE Trans. Comput., vol. 66, no. 1, pp. 52–58, 2017, doi: 10.1109/TC.2016.2582151.

[17] J. Kwack, T. Applencourt, C. Bertoni, Y. Ghadar, H. Zheng, C. Knight, and S. Parker, “Roofline-based performance efficiency of hpc benchmarks and applications on current generation of processor architectures,” in 2019 Cray User Group Meeting, 2019, vol. 5.

[18] M. Hill and V. Janapa Reddi, “Gables: A roofline model for mobile SoCs,” Proc. - 25th IEEE Int. Symp. High Perform. Comput. Archit. HPCA 2019, pp. 317–330, 2019, doi: 10.1109/HPCA.2019.00047.

[19] C. Yang and L. Berkeley, “Hierarchical Roofline Analysis on GPUs,” 2020.

[20] N. K. Jha and S. Mittal, “Modeling Data Reuse in Deep Neural Networks by Taking Data-Types into Cognizance,” IEEE Trans. Comput., vol. 70, no. 9, pp. 1526–1538, 2021, doi: 10.1109/TC.2020.3015531.

[21] NVIDIA Corporation, CUDA C Programming Guide - Version 4.2. 2012.

[22] B. Van Werkhoven, J. Maassen, H. E. Bal, and F. J. Seinstra, “Optimizing convolution operations on GPUs using adaptive tiling,” Futur. Gener. Comput. Syst., vol. 30, no. 1, pp. 14–26, 2014, doi: 10.1016/j.future.2013.09.003.

تحليل كلفة U-Net باستخدام نموذجRoofline

علا طارق سالم
ula.tariq@uomosul.edu.iq

شفاء عبد الرحمن داؤد
shefa.dawwd@uomosul.edu.iq

فخر الدين حامد علي
fhazaa@ uomosul.edu.iq

جامعة الموصل - كلية الهندسة - قسم هندسة الحاسوب

الملخص

من أهم التحديات التي تواجه أداء هندسة U-Net هي طريقة تصميم مكوناتها وكيفية اختيار جهاز الحوسبة المناسب للتعامل مع مجموعات البيانات التدريبية المسمى. الالتفاف هو أكثر العمليات التي تتطلب الحسابات وتكاليف الذاكرة ، والتي يجب تقليلها. وبالتالي ، فإن أحد الاختيارات المناسبة هو تغيير نوع الالتواء. تتمثل الحلول الأخرى المقترحة في تقليل حجم الصورة وعدد البتات وقيمة الخطوة بالإضافة إلى عدد المرشحات ودُفعات الصور. لذلك ، في هذا البحث ، سيتم استخدام نموذج Roofline كدليل أداء في تحليل FLOPs وحدود عرض النطاق الترددي للذاكرة لنموذج U-Net بتكوينات مختلفة. تم تقييم التكلفة بالمقارنة مع محدودية ثلاثة أجهزة حوسبة ، GPU230MX و GPU940MX و GPU2060rtx super. تم استخدام مجموعة بيانات صورة 128 × 128 أثناء عملية تقييم أداء التكلفة في U-Net. بناءً على التحليل ، تظهر نتائج التقييم أن الحل الذي يحقق التوازن بين الذاكرة والحسابات هو تنفيذ نموذج U-Net بالتوازي باستخدام بطاقة سوبر RTX2060 مع تكوينات حجم الدفعة 16 ، حجم الصورة 128 × 128 ، عدد البتات هو 32 ، إدارة الذاكرة المشتركة.

الكلمات الداله :

نموذج Roofline، U-Net *، عدد العمليات الفاصلة العائمة ، عرض النطاق الترددي للذاكرة ، الكثافة الحسابية.*