The extended version of this paper published in IEEE Transaction on Computers (DOI: 10.1109/TC.2020.3015531)

Data-type Aware Arithmetic Intensity for Deep Neural Networks

Nandan Kumar Jha*, Sparsh Mittal*, Sasikanth Avancha†
*Indian Institute of Technology Hyderabad, †Parallel Computing Lab, Intel Corporation
Email: {cs17mtech11010, sparsh}@iith.ac.in, sasikanth.avancha@intel.com

***Abstract*—In recent years, researchers have focused on reducing the model size and number of computations (measured as "multiply-accumulate" or MAC operations) of DNNs. The energy consumption of a DNN depends on both the number of MAC operations and the energy efficiency of each MAC operation. The former can be estimated at design time, however, the latter depends on the intricate data reuse patterns and underlying hardware architecture and hence, estimating it at design time is challenging. In this work, we show that the naive approach to estimate the data reuse, viz. arithmetic intensity, does not always correctly estimate the degree of data reuse in DNNs since it gives equal importance to all the data types. We propose a novel model, termed "data type aware weighted arithmetic intensity" (DI) which accounts for unequal importance of different data types in DNNs. We evaluate our model on 25 state-of-the-art DNNs on two GPUs and show that our model accurately models data-reuse for all possible data reuse patterns, for different types of convolution and different types of layers. We show that our model is a better indicator of energy efficiency of DNNs. We also show its generality using the central limit theorem.**

***Index Terms*—Deep neural networks (DNNs), energy-efficiency, arithmetic intensity, roofline model.**

I. Introduction

DNNs are now being used in a wide range of cognitive applications. After the success of AlexNet [1], the research in DNNs has focused on achieving higher accuracy even at the cost of DNN size and computational complexity. This has led to over-parameterized DNNs, e.g., VGG-16 [2], Inception-v4 [3], ResNet152-v2 [4] etc. By contrast, recent networks such as SqueezeNet [5], MobileNet-V1 [6], and MobileNet-V2 [7] focus on making the DNN compact by reducing the number of parameters and/or MACs. However, reducing the number of MACs does not necessarily make DNNs energy efficient because energy is dominated by data movement rather than computation [8]. The data movement primarily depends on the degree of data reuse present in the workloads.

To enable deployment of DNN models in a wide range of applications such as autonomous driving, drones, etc., the energy consumption of DNN inference must be within a prescribed envelope. Hence, DNNs need to be carefully examined based on the number of computations (i.e., MACs) as well as the energy efficiency of MAC operations. Unfortunately, the latter metric has been largely overlooked in DNN design because a study of energy efficiency requires precise knowledge of the degree of data reuse and parallelism present in the DNNs, and how the underlying hardware platform exploits this parallelism. Further, the implications of reducing the number of parameters and MACs on the energy efficiency of DNN is not well-understood.

Traditionally, arithmetic intensity [9] is used to model the degree of data reuse and also used in "roofline model" [10] for predicting whether a workload is compute-bound or memory-bound. Therefore, it is a representative of the *degree of data reuse* and *bandwidth pressure*. Lower arithmetic intensity implies a lower degree of data reuse and high bandwidth pressure and vice versa. The arithmetic intensity considers the memory footprint and the number of arithmetic operations and shows the degree of data reuse available in an algorithm. In other words, arithmetic intensity shows how efficiently arithmetic operation can reuse the data fetched from different levels in the memory hierarchy. However, the memory footprint does not reflect the actual number of off-chip accesses, which largely depends on the architecture of memory hierarchy and how well the underlying platform exploits the data reuse available in the algorithm. Arithmetic intensity can be used to represent power/energy efficiency only when all the data types have the same access behavior. For example, [11] and [12] use arithmetic intensity to model the power/energy efficiency.

DNNs have different types of data such as filter weights, input and output activations, partial sums, etc. which have different reuse pattern [13] and hence, reuse importance. Also, the layers in DNNs have distinct computation and reuse patterns with different bandwidth requirements. For example, convolution (Conv) layers have a high degree of reuse, and they are compute-bound, whereas fully connected (FC) layers have low-reuse and a high number of parameters and hence, they are memory bound [14]. Furthermore, due to the different DNN topologies such as branching, skip connections, dense connections, etc., the number of concurrent activation varies during the runtime [15] which in turn affects the reuse behavior of DNNs. Given these factors, *it is interesting to investigate whether arithmetic intensity can be used as a representative of energy efficiency in DNN models*, or, is there a need for a more accurate metric.

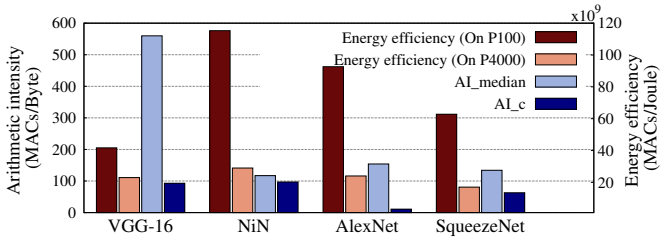


Fig. 1: AI_{median} and AI_c are not good indicators of data-reuse

In Figure 1, cumulative arithmetic intensity (AI_c) and median value of layer-wise arithmetic intensity (AI_{median}) of DNNs

are shown (these metrics are explained in §III). VGG-16 [2] and NiN [16] have almost equal AI_c but NiN has significantly higher energy efficiency (measured on both GPU P100 and P4000) than VGG-16, even when VGG-16 has quite higher AI_{median}. Also, AlexNet has quite low AI_c and AI_{median} than VGG-16 but has very high energy efficiency than VGG-16. It shows that both layer-wise arithmetic intensity and cumulative arithmetic intensity are not good indicators of energy efficiency of DNNs and hence, *there is a need for better model/metric to estimate the data reuse in DNNs to understand their energy efficiency.*

Our main **contributions** can be summarized as follows.

1) We first explore the intricacies of data reuse patterns in DNNs (§II) and perform a detailed analysis of layer-wise data reuse patterns in DNNs (§III). We include all the possible variations of data reuse patterns arising from (A) different types of convolutions such as standard, group, pointwise, depthwise, etc., (B) different types of layers such as Conv, FC and others, and (C) different design heuristics such as feed-forward/skip connections. Through our comprehensive experiment and analysis, we show that *data reuse estimated by arithmetic intensity is not tightly-coupled with the energy efficiency of MAC operations in DNNs*.
2) We experimentally measure the energy consumption and energy efficiency of DNNs on two GPUs, P100 and P4000, and show that *activation reuse has a higher impact on energy efficiency than weight reuse*.
3) We propose a novel model to capture the dynamics of data reuse patterns in DNNs (§IV). Our proposed metric (DI) more accurately quantifies the intrinsic relationship between data reuse and energy efficiency of MAC operations (§IV).
4) We validate our model on 25 state-of-the-art DNNs, including both highly-accurate DNNs and compact DNNs, which have between 221 to 15,470 million weights and between 0.54 to 138 million MACs (§VI). We also use "central limit theorem" to prove the generality of our proposed model (§VII).

II. Background and Motivation

Table I lists the symbols used. For simplicity and ease of comparison, we assume that a) height and width of filter are same, b) height and width of output feature map (ofmap) are same, and c) spatial size of input feature map (ifmap) and ofmap are equal.

TABLE I: Symbols (Fmap = feature map, Ops = operations)

Quantity (symbol)	Unit	Quantity (symbol)	Unit
Energy per pixel (EPP)	Joule	Energy efficiency	MACs/Joule
# Weights (W)	Millions	Throughput	MACs/sec
# MACs (M_c)	Millions	# Activations (A)	Millions
Fmap height (S_o)	-	Fmap width (S_o)	-
Filter height (S_k)	-	Filter width (S_k)	-
# filter-channels (M)	-	# filters (N)	-
Group Size (G)	-	# Groups ($g = \frac{M}{G}$)	-
Compute to memory ratio (CMR)	OPs/sec/Byte	Disparity factor (d_f)	-
Cumulative arithmetic intensity (AI_c)	MACs/Byte		-

A. Data reuse patterns in DNNs

Types of convolution: Broadly, there are four types of convolutions. They are discussed below and their properties are summarized in Table II. Here, weight (learnable filter coefficients) and activation (ifmaps and ofmaps) reuse are estimated as M_c/W and M_c/A, respectively. We use arithmetic intensity defined as $\frac{M_c}{W+A}$, as a metric to evaluate the bandwidth requirement of MACs [9].

1. Standard (spatial) convolution: In standard convolution, filtering (i.e. feature extraction) and combining (i.e. feature aggregation) are performed together. The total number of filter weights and activations (combined ifmaps and ofmaps) involved in standard convolution is $M \times N \times S_k^2$ and $(M + N) \times S_o^2$, respectively. The total number of MAC operations in standard convolution is $M \times N \times S_k^2 \times S_o^2$. Because of combined feature extraction and aggregation, standard convolution incurs high computational complexity.

2. Pointwise convolution: In pointwise convolution, the smaller receptive size of filter reduces the number of MACs as well as number of filter weights involved, which are $M \times N \times S_o^2$ and $M \times N$, respectively. Number of activations is same as that in standard convolution. Pointwise convolution has been used in NiN [16], in inception module [3], [17], [18] and in SqueezeNet [5].

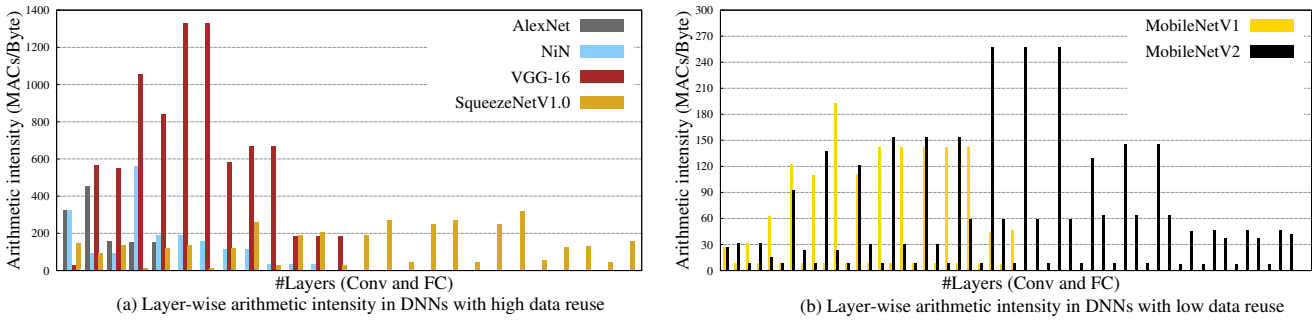
3. Group convolution: In group convolution, each group of 2-D filters are convolved with G number of input feature maps. Compared to standard convolution, in group convolution, the number of MACs and number of filter weights are reduced by a factor of G. The number of activations remains same as that in standard convolution. Group convolution has been used in AlexNet [1] ($g = 2$), ResNext (g = 32) [19] and 1.0-G-SqNxt-23 (g = 2) [20].

4. Depthwise convolution: Depthwise convolution performs only feature extraction where, one filter convolves with only one input feature map, i.e., one channel of input. Compared to standard convolution, it reduces the number of MACs and number of weights by a factor of N (Table II). Total number of activations involved in depthwise convolution is $2 \times M \times S_o^2$. Depthwise convolution has been used in MobileNet-V1 [6], MobileNet-V2 [7], and XceptionNet [21]. Note that, depthwise convolution is an extreme case of group convolution where G = 1, i.e. $g = N = M$.

TABLE II: Characteristics of different types of convolution

Convolution	Arithmetic intensity	$\frac{M_c}{W}$	$\frac{M_c}{A}$
Standard	$\frac{M \times N \times S_k^2 \times S_o^2}{M \times N \times S_k^2 + (M+N) \times S_o^2}$	S_o^2	$\left(\frac{M \times N}{M+N}\right) \times S_k^2$
Pointwise	$\frac{M \times N \times S_o^2}{M \times N + (M+N) \times S_o^2}$	S_o^2	$\left(\frac{M \times N}{M+N}\right)$
Group	$\frac{M \times N \times S_k^2 \times S_o^2}{M \times N \times S_k^2 + g \times (M+N) \times S_o^2}$	S_o^2	$\left(\frac{M \times N}{M+N}\right) \times \frac{S_k^2}{g}$
Depthwise	$\frac{M \times S_k^2 \times S_o^2}{M \times S_k^2 + (M+M) \times S_o^2}$	S_o^2	$\left(\frac{M}{M+M}\right) \times S_k^2$

Others layers: The non-Conv layers such as pooling, ReLU, BatchNorm etc. have negligible number of learnable parameters and there is no MAC operation involved, although there are other operations such as element-wise addition, comparison, division etc. These layers have low arithmetic intensity and high bandwidth requirement [14]. FC layers have very high number of parameters (weights), and fewer activations, which makes FC layers memory bound. In other words, both the arithmetic intensity and weight reuse in FC layer is approximately equal to 1 (as $M_c \approx W$ and $A \ll W$).



(a) Layer-wise arithmetic intensity in DNNs with high data reuse

(b) Layer-wise arithmetic intensity in DNNs with low data reuse

Fig. 2: Layer-wise (Conv and FC) arithmetic intensity in (a) DNNs with high data reuse and (b) DNNs with low data reuse.

B. Motivation

Different types of convolutions have been applied to accomplish different design goals and to achieve a trade-off between performance and computation/bandwidth overhead. Apart from this, various design heuristics have been used, such as residual connections [4], [22] to facilitate backpropagation in deeper networks, dense connections [23] to enable feature reuse, etc. These design heuristics lead to different computational complexity and degrees of data reuse. Even in standard convolution, degree of data reuse depends on multiple factors such as filter's dimensions, convolution stride, dimensions of ifmaps, etc. For the comparison of data reuse and computational complexity in different types of convolution, we assume specific values of variables and show the values of metrics normalized to that for standard convolution in Table III.

TABLE III: Metrics values normalized to standard convolution (assuming $N = M = 256$, feature map size $(S_o \times S_o) = 28 \times 28$, filter size $(S_k \times S_k) = 3 \times 3$ and group size $(g) = 4$)

Convolution	Arithmetic intensity	M_c	$\frac{M_c}{W}$	$\frac{M_c}{A}$
Standard	1.00	1.000	1	1.000
Pointwise	0.24	0.111	1	0.111
Group	0.45	0.250	1	0.250
Depthwise	0.01	0.004	1	0.004

Evidently, the relative number of MACs decreases from standard convolution to depthwise convolution but the arithmetic intensity also reduces which would lead to high bandwidth requirement. Since energy is dominated by data movement rather than computation [8], *a decrease in number of MACs can be dwarfed by the increase in memory accesses and hence, the overall energy consumption can be increased*. These observations motivate us to investigate a model which can better incorporate the dynamics of data reuse in DNNs and is a better indicator of the energy efficiency of MACs in DNNs.

C. Experimental setup and metrics

We perform our experiments using Caffe [24] on two GPUs, viz., Tesla P100 [25] and Quadro P4000 [26] which have significantly different compute and memory resources, as shown in Table IV. Former is a data-center scale GPU while latter is a desktop GPU.

TABLE IV: Parameters of GPU used in our experiments

GPU	# core	L2 size	Peak bandwidth	Peak Throughput	CMR
P4000	1792	2 MB	243 GB/s	5.2 TFLOPS	21.4 FLOPs/Byte
P100	3584	4 MB	549 GB/s	9.3 TFLOPS	16.94 FLOPs/Byte

We use input batch size of four for better utilization of GPU compute resources. The inference time is averaged over 50 iterations and for power measurements, we use `nvidia-smi` utility. "Energy per pixel (EPP)" (Eq. 1) measures the energy consumed in processing of one input pixel over entire DNN, whereas "energy efficiency" (Eq. 2) shows the energy consumed in one MAC operation. We have used EPP to remove the bias due to differences in input image size used by different DNNs. For example, InceptionV3, InceptionV4, and XceptionNet work with input image size 299×299 while most of DNNs work with input size 224×224.

$$\text{EPP} = \frac{(\text{Average power}) \times (\text{inference time})}{\#\text{Pixels in input frame}} \quad (1)$$

$$\text{Energy Efficiency} = \frac{(\text{batch size}) \times (\#\text{ MACs})}{(\text{Average power}) \times (\text{inference time})} \quad (2)$$

III. Naive Approaches

We now discuss two possible approaches for estimating the data reuse in DNNs and also show their limitations. These approaches are (1) layer-wise arithmetic intensity (§III-A) and (2) cumulative arithmetic intensity (AI_c) (§III-B).

A. Layer-wise arithmetic intensity

Figure 2 shows arithmetic intensity of a layer (Conv and FC) defined as the ratio of "number of MACs performed in that layer" to "the sum of total number of weights and activations in that layer". We divide DNNs in two categories based on their degree of data reuse: those with higher data reuse, e.g., AlexNet, VGG-16, NiN, and SqueezeNetV1.0 (Figure 2(a)) and those with lower data reuse, e.g., MobileNet-V1 and MobileNet-V2 (Figure 2(b)). From Figure 2(a), we observe that arithmetic intensity of nearly all layers of VGG-16 and NiN are higher and lower (respectively) than that of other DNNs. Similarly, from Figure 2(b), the arithmetic intensity of layers in MobileNet-V2 is either comparable or higher than that of the layers in MobileNet-V1.

Table V shows the median and variance of the arithmetic intensity of layers in these DNNs. The median value of layer-wise arithmetic intensity in VGG-16 is significantly higher than the median value of layer-wise arithmetic intensity in other DNNs with higher data reuse. Also, NiN has the lowest median among the DNNs with higher data reuse. Intuitively, the energy efficiency of VGG-16 should be higher compared to other DNNs with high data reuse and also the energy efficiency of NiN should be lowest in the same group. Surprisingly, on

both P4000 and P100 GPUs, *the energy efficiency of NiN is highest, and that of the VGG-16 is lowest among the DNNs with higher data reuse*.

TABLE V: Median and variance of layer-wise arithmetic intensity in DNNs (SqNet=SqueezeNetV1.0)

Metric	AlexNet	VGG-16	NiN	SqNet	MobileNet-V1	MobileNet-V2
Median	154	560	117	134	18	32
Variance	2.69E+04	2.09E+05	2.24E+04	8.30E+03	3.70E+03	4.52E+03

Similarly, MobileNet-V2 has lower energy efficiency than MobileNet-V1, which is counter-intuitive. By observing the variance of the layer-wise arithmetic intensity of DNNs in Table V, one may argue that since the variance values of VGG-16 and MobileNet-V2 are higher than the other DNNs in their respective groups, hence their energy efficiency is lowest, despite having higher median. This argument is flawed because it is unable to explain why NiN has higher energy efficiency than SqueezeNetV1.0 even when NiN has significantly higher variance than SqueezeNetV1.0. Above discussion proves that *"layer-wise arithmetic intensity" is not a good indicator of energy efficiency of DNNs* with both high and low data reuse. Besides this, finding layer-wise arithmetic intensity in DNNs such as Inception-V3, Inception-V4, Inception-resnet-v2, etc. is tedious work because these networks have thousands of Conv layers and many layers have several branches leading to irregular computation and reuse patterns.

B. Cumulative arithmetic intensity

For each DNN, a single metric, viz., arithmetic intensity ($AI_c = \frac{M_c}{W+A}$) is defined as the ratio of "total number of MAC operations performed by network in one forward pass" to "the sum of total number of weights and activations that network has". We have plotted the roofline models with AI_c for 25 DNNs (Table VIII) on two GPUs P4000 (Figure 3(a)) and P100 (Figure 3(b)). We also measure the energy efficiency of 25 DNNs on P4000 and P100 and plot in Figures 3(e) and 3(f), respectively.

In the roofline models (with AI_c) on both GPUs (Figure 3(a) and (b)), MobileNet-V1, DenseNet and XceptionNet are predicted as compute-bound whereas AlexNet is predicted as bandwidth-bound. It is well known that due to the costlier off-chip accesses, bandwidth-bound operations are energy inefficient compared to compute-bound operations. Despite this, AlexNet has substantially higher energy efficiency compared to that of the MobileNet-V1, DenseNet, and XceptionNet (as measured on both P4000 and P100 GPUs). From Figure 3(e) and 3(f), it is evident that MobileNet-V1 has very low energy efficiency while AlexNet has very high energy efficiency. Also, XceptionNet and DenseNet are predicted as compute-bound, but their energy efficiency is lower than that of the AlexNet. In summary, *data reuse predicted by AI_c is not correlated with the energy efficiency of DNNs*.

To understand the reason behind limitations of AI_c, we study the architecture of XceptionNet and DenseNet. We found that DenseNet has many feed-forward/skip connections which increases the number of concurrent activations [15] and decreases the effective data reuse. Similarly, XceptionNet uses depthwise separable convolution (DWSconv) which has very low data reuse (Table III). It also has skip connections which increase the concurrent activation data and further reduce the data reuse. AI_c $\left(\frac{M_c/W}{1+A/W} = \frac{M_c/A}{1+W/A}\right)$ gives equal importance to weight reuse (M_c/W) and activation reuse (M_c/A) and hence, it is unable to capture the runtime change in data reuse.

IV. Proposed Model

In this section, we first discuss the need of giving unequal importance to reuse of different data types, specifically weights and activations (§IV-A). Then, we propose our model, which more accurately incorporates the dynamics of data reuse in DNNs (§IV-B). We highlight the effectiveness of our model (§IV-C) and compare it with AI_c to get more insights and explain why AI_c fails to predict the nature (memory-bound/compute-bound) of some DNNs (§IV-D).

A. Reuse of different data types has unequal importance

As we know, the number of weights (W) in a DNN does not change at runtime. However, the number of concurrent activations can change at run time and grows in proportion to the number of feed-forward connections (as discussed in §III-B). DNNs such as DenseNet have a relatively higher number of skip connections, which leads to a substantial increase in concurrent activations. Further, as shown in Table VIII, the ratio of the total number of activations to total number of weights, i.e., $\frac{A}{W}$ varies significantly across different DNNs, ranging from 0.03 in AlexNet to 32.80 in 1.0-G-SqNxt. This imbalance between W and A creates an imbalance between $\frac{M_c}{W}$ and $\frac{M_c}{A}$, for example, compact DNNs such as MobileNet and SqueezeNext have very low $\frac{M_c}{A}$ (Table VIII). To account for the imbalance between the weight and activation reuse and also to model the runtime change in effective data reuse, our model decouples the weight and activation reuse.

Does the decoupling of weight and activation reuse inherits the properties of arithmetic intensity? As shown in Table II, the arithmetic intensity of commonly used convolution types are different. On decoupling the data reuse in terms of weight and activation reuse, we find that in all convolution types, the weight reuse is the same (i.e., S_o^2) whereas activation reuse is different. *This is quite interesting because it shows that the variation in arithmetic intensity for different convolution types are governed by the variation in activation reuse*. Since lower arithmetic intensity leads to higher bandwidth pressure, lower activation reuse leads to a higher number of memory accesses, which can make DNN energy inefficient. Based on these insights, activation reuse should be given more importance than weight reuse while modeling the importance of weight and activation reuse. As shown in Table III, the activation reuse decreases from standard convolution to depthwise convolution in the same order in which the relative value of arithmetic intensity is decreasing (but with different magnitudes). This confirms that *decoupling the data reuse in weight and activation reuse inherits the properties of arithmetic intensity.*

B. Decoupling the weight and activation reuse

We now decouple weight, and activation reuse from the formulation of AI_c and establish a relation between arithmetic

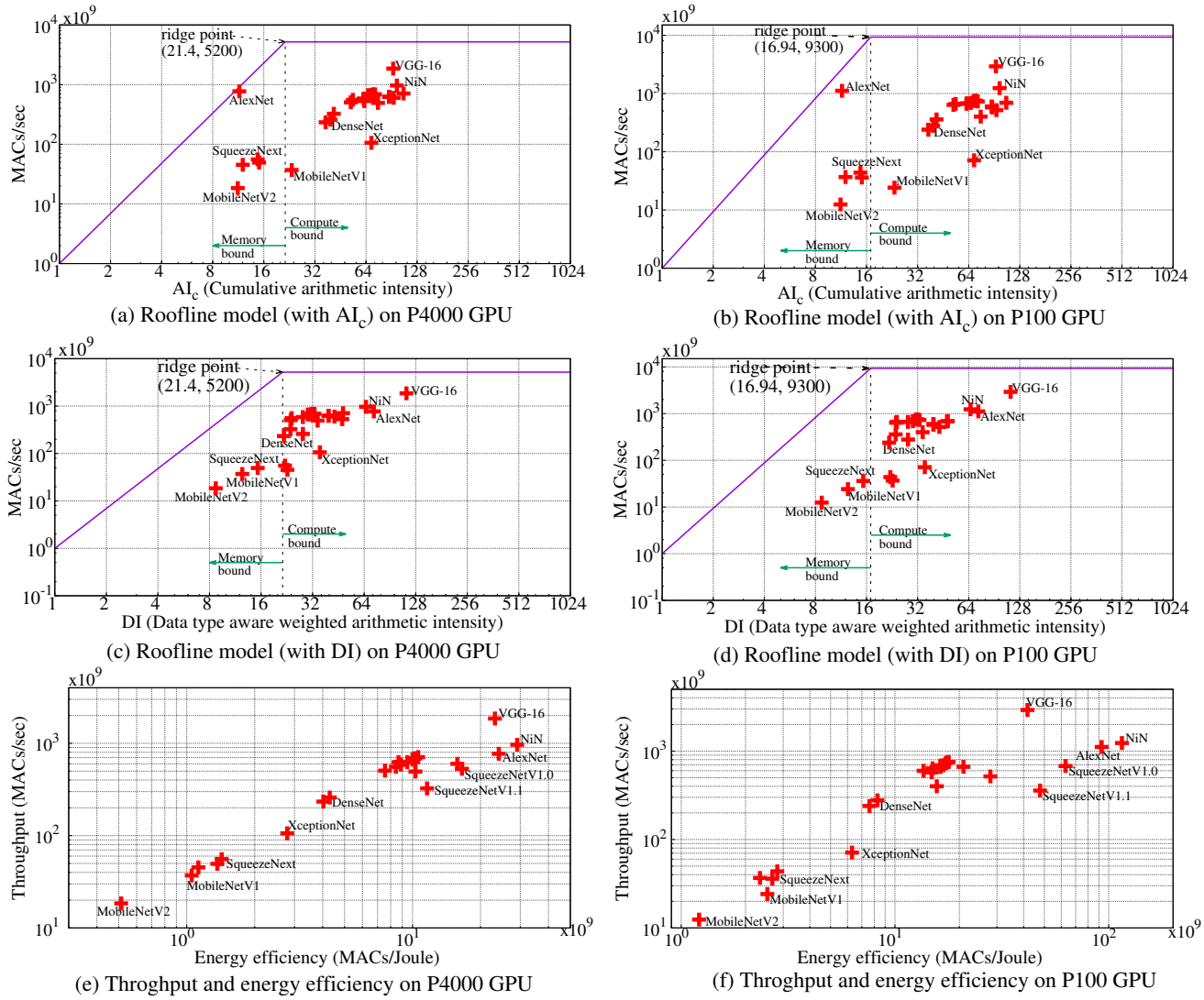


Fig. 3: Roofline model with (a) AI_c and (c) DI, and (e) energy efficiency on P4000. (b), (d), (f): these results on P100.

intensity (AI_c) and weight and activation reuse. Since arithmetic mean is never less than harmonic mean, we have

$$\frac{W+A}{2} \geqslant \frac{2\times W\times A}{W+A} \implies \frac{2\times M_c}{W+A} \leqslant \frac{M_c\times(W+A)}{2\times W\times A}$$

$$\implies \frac{M_c}{W+A} \leqslant \frac{1}{4}\times\left[\frac{M_c}{A}+\frac{M_c}{W}\right]$$

$$\implies AI_c \leqslant \frac{1}{4}\times[\text{ActivationReuse}+\text{WeightReuse}] \quad (3)$$

To give unequal importance, we introduce a *data reuse coefficient* (α) in Eq. 3, where $0 \leq \alpha \leq 1$. Different values of α would give different weightage to both types of reuse. After introducing α in Eq. 3, we refer the resultant metric as "data type aware weighted arithmetic intensity" (DI).

$$DI = \frac{1}{4}\times[\alpha\times\text{ActivationReuse}+(1-\alpha)\times\text{WeightReuse}] \quad (4)$$

To find the value of α such that DI would become a more accurate indicator of energy efficiency, we find the Pearson correlation coefficient (r) between DI and energy efficiency for α values between 0 to 1 with a step size of 0.05. We experimentally measure the energy efficiency of 25 state-of-the-art DNNs. As we know, a value of r close to +1/-1 indicates stronger positive/negative (respectively) correlation and a value close to 0 indicates no correlation. Figure 4 shows the results, and we observe that, on both GPUs, with increasing value of α, the correlation continues to increase till it reaches a plateau at $\alpha \approx 0.80$. The increase in r with higher value of α is consistent on both GPUs, which have significantly different compute power and CMR (Table IV). Thus, these results are not platform-dependent rather platform-agnostic. Hence, we take $\alpha = 0.80$ and substituting this value in Eq. 4 gives the final expression for DI. We do not take value of α greater than 0.80 because correlation starts saturating at $\alpha = 0.80$ and $\alpha > 0.80$ renders the population correlation coefficient (ρ) outside the confidence window (as explained in §VII).

In Table VI, we compare our model (DI with α = 0.80) with AI_c using correlation coefficient. Evidently, compared to AI_c, DI has a much stronger correlation with energy efficiency as measured for 25 DNNs on both P4000 and P100 GPU. Hence, DI is a better indicator of energy efficiency of DNNs.

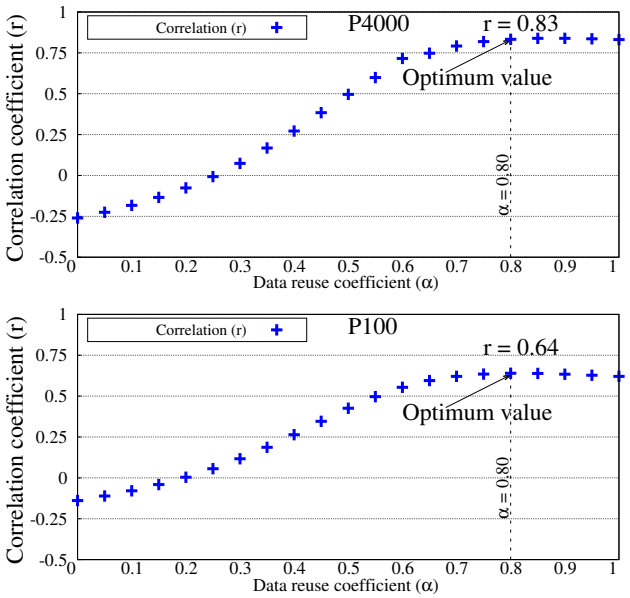


Fig. 4: Correlation (for different values of α) between DI and energy efficiency on P4000 and P100 GPU.

From Table II, we concluded that *its the activation reuse which causes variation in arithmetic intensity, not the weight reuse*. In our model, we obtained $\alpha = 0.80$, which indicates that activation reuse has a much higher impact on DI than weight reuse. Evidently, our model confirms the intuition (§IV-A) and highlights the importance of activation reuse.

TABLE VI: Correlation of AI_c and DI with energy efficiency of 25 DNNs measured on P4000 and P100 GPUs

GPU	X	Y	r = **correlation** (X,Y)
P4000	AI_c	Energy efficiency	0.52
	DI (**Ours**)	Energy efficiency	**0.83**
P100	AI_c	Energy efficiency	0.23
	DI(**Ours**)	Energy efficiency	**0.64**

C. Salient features of our model

We first discuss how modeling of activation and weight reuse captures all types of reuses across different layers in DNNs. Based on the computation and data reuse patterns, layers in DNNs can be broadly categorized as Conv, non-Conv, and FC layers. In Conv layers, there is feature map reuse, filter reuse, and filter weight reuse [13], whereas in FC layers, except filter-weight reuse, all these types of reuses are possible. In other words, all the possible data reuse in both Conv and FC layer can be expressed as weight and activation reuse. However, in non-Conv layers, only ifmap and ofmap activations are processed, and their reuse can be described as activation reuse. As shown in Table II, the degree of data reuse in different types of convolutions is different due to the activation reuse, whereas weight reuse remains constant across all types of convolutions. This implies that, energy efficiency of MAC operations across different type of convolutions can be expressed in terms of activation reuse. Since, non-Conv layers do not have learn-able parameters and process only feature maps, their energy metrics are governed by activation reuse. However, in FC layers, $M_c \approx W$ and $A \ll W$, thus, arithmetic intensity and weight reuse are approximately equal to 1, and hence, the energy efficiency of MACs in FC layers are governed by weight reuse.

In summary, except FC layer, the energy efficiency of all the layers can be expressed in terms of activation reuse, and the higher value of α implies higher importance of activation reuse. This shows that FC layers have lower impact on the energy efficiency of DNNs. In fact, deeper networks such as Inception-V4, Inception-ResNet-V2 have hundreds of Conv and non-Conv layers, but very few FC layers and some networks such as NiN have no FC layers at all.

We now discuss how our model addresses the shortcomings of AI_c. As discussed in §III-B, AI_c predicts AlexNet as memory-bound and MobileNet-V1 as compute-bound (shown in roofline models in Figure 3(a) and 3(b)) but the energy-efficiency of AlexNet is quite high and that of the MobileNet-V1 is quite low (Figure 3(e) and 3(f)). This is counter-intuitive because operations (MACs) of memory-bound workload are energy-inefficient due to the higher number of memory-accesses. The reason for these irregularities are better explained by our model. As shown in Table VIII, AlexNet has very low $\frac{M_c}{W}$ but significantly high $\frac{M_c}{A}$, whereas MobileNet-V1 has high $\frac{M_c}{W}$ but significantly low $\frac{M_c}{A}$. By virtue of giving higher importance to $\frac{M_c}{A}$, our model is able to accurately predict AlexNet as compute-bound and MobileNet-V1 as memory-bound (Figure 3 (c) and (d))

D. When and why does AI_c fails?

We define relative disparity (d_f) between AI_c and DI as follows.

$$d_f = \left(\frac{AI_c - DI}{AI_c}\right) \times 100 = 75 - 6.25 \times \left[\frac{A}{W} + 3 \times \frac{W}{A}\right] \quad (5)$$

Above Equation (Eq. 5) shows that, $\frac{A}{W}$ has less impact on relative disparity (d_f) than $\frac{W}{A}$. Since, $\frac{A}{W}$ is same as weight reuse to activation reuse ratio, Eq. 5 implies that weight reuse has less impact on (d_f). Table VIII shows weight reuse ($\frac{M_c}{W}$), activation reuse ($\frac{M_c}{A}$), AI_c and DI, activation to parameter ratio ($\frac{A}{W}$) and d_f value for 25 state-of-the-art DNNs. For gaining more insights, we study three cases which are shown in Table VII.

TABLE VII: AI_c, DI and d_f for 3 cases

	Case 1: $A \ll W$	Case 2: $A \approx W$	Case 3: $A \gg W$
AI_c	$\approx M_c/W$	$\approx 0.5 \times M_c/A$	$\approx M_c/A$
DI	$\approx 0.2 \times M_c/A$	$\approx 0.25 \times M_c/A$	$\approx 0.06 \times M_c/W$
d_f	$\approx 75 - 18.75 \times \frac{W}{A}$	≈ 50	$\approx 75 - 6.25 \times \frac{A}{W}$

As shown in Table VII, in **case 1**, activation reuse ($\frac{M_c}{A}$) dominates total data reuse ($\frac{M_c}{W}$ + $\frac{M_c}{A}$), but AI_c is nearly equal to weight reuse ($\frac{M_c}{W}$). This leads to huge disparity between AI_c and DI. For example, AlexNet has 30× higher activation reuse than weight reuse and hence, its relative disparity is highest among all the 25 DNNs (Table VIII). In **case 3**, weight reuse dominates the total data reuse, however, disparity is noticeable but not as large as in case 1 because weight reuse has less impact compared to activation reuse (refer Eq. 4). In **case 2**, the d_f is lower and AI_c would be able captures the dynamics of data reuse in DNNs. In summary, when either $A \approx W$, for example in variants of InceptionNet, ResNet and ResNext (Table VIII), or when A is significantly higher than W, for

example MobileNet-V2 and variants of SqueezeNext (Table VIII), AI_c would be able to capture the data reuse patterns in DNNs. *However, AI_c fails to estimate the data reuse when the $\frac{A}{W}$ ratio is very low (e.g., AlexNet) and also, when $\frac{A}{W}$ ratio is moderately high (e.g., MobileNet-V1).*

TABLE VIII: Reuse, AI_c, $\frac{A}{W}$, DI, and d_f for various DNNs

Model Name	$\frac{M_c}{W}$	$\frac{M_c}{A}$	AI_c	DI	$\frac{A}{W}$	d_f
AlexNet [1]	12	362	11	73	0.03	**-564**
VGG-16 [2]	112	537	93	113	0.21	-22
NiN [16]	146	291	97	66	0.50	32
GoogLeNet [17]	227	158	93	43	1.44	54
Inception-V2 [18]	196	122	75	34	1.61	55
Inception-V3 [18]	240	138	88	40	1.74	55
Inception-V4 [3]	287	169	106	48	1.70	55
ResNet-50 [22]	151	83	54	24	1.83	56
ResNet-101 [22]	170	108	66	30	1.58	55
ResNet-152 [22]	188	113	70	32	1.66	54
ResNet101-V2 [4]	176	112	68	31	1.56	54
ResNet152-V2 [4]	192	116	72	33	1.65	54
Inception-ResNet-V2 [3]	236	139	87	40	1.70	54
ResNext50-32x4d [19]	191	72	52	24	2.67	54
ResNext101-32x4d [19]	206	89	62	28	2.31	55
DenseNet-121 [23]	386	44	40	28	8.77	30
DenseNet-169 [23]	263	43	37	22	6.10	41
SqueezeNet-V1.0 [5]	678	69	63	48	9.84	24
SqueezeNet-V1.1 [5]	282	48	41	24	5.81	41
1.0-SqNxt-23 [20]	381	15	15	22	24.84	-47
1.0-SqNxt-23v5 [20]	242	16	15	15	15.12	0
1.0-G-SqNxt-23 [20]	406	12	12	23	32.80	-92
MobileNet-V1 [6]	136	28	23	12	4.80	48
MobileNet-V2 [7]	125	12	11	9	10.10	18
XceptionNet [21]	367	85	69	35	4.32	49

V. Applicability of proposed model

Our model estimates the data reuse available in DNNs with the assumption that underlying platforms have sufficient compute/memory resources to exploit the available data reuse in DNNs. However, different hardware platforms are optimized for contrasting design goals and have dissimilar memory-hierarchy with non-identical number of layers and capacity. We now discuss whether our model is applicable to hardware platforms such as CPU, FPGA, etc., or do we need to re-calibrate the value of α on them?

GPU: Both P100 and P4000 GPUs have substantially different memory and compute capability, which is also manifested by different CMR values (Table IV), and hence the memory-access pattern and their cost would vary significantly for both the GPU. In Figure 5(a), it is shown that EPP values of DNNs are higher on P4000 GPU compared to that on P100 GPU. Our extensive experiments validate the proposed model, and hence does not require re-calibration of α for both the GPUs, even though they have different compute and memory resources. This indicates applicability of our model to different GPUs regardless of their memory-hierarchy and CMR values.

CPU: CPUs have different memory hierarchy and memory management techniques to optimize the single thread performance. Specifically, the hardware managed cache hierarchy with other optimization techniques such as cache miss management and cache coherence protocol largely vary among different CPU architectures. Also, unlike the GPUs, CPUs have higher off-chip memory, which can accommodate a larger model with larger batch size. Furthermore, unlike GPUs, which have a large number of cuda-cores (Table IV), CPUs have limited parallelism in the compute unit. Thus, to include these dynamics in our proposed model, α may need to be re-calibrated. However, decoupling weight and activation reuse would still be useful.

FPGA: Unlike CPUs and GPUs, FPGAs have limited on-chip/off-chip memory resource along with limited compute power, which may be insufficient to exploit the available data reuse in large models. Moreover, these platforms are optimized for some specific application, and hence their memory architecture and memory management techniques do not vary as much as in general purpose computing engines. We believe that, for such platforms, the energy efficiency of MACs may be different compared to that of in general purpose hardware platforms, however, the correlation between DI and energy efficiency would remain intact. Hence, we do not need to re-calibrate the α.

Effect of memory hierarchy: We discuss two extreme cases. The processor has 1) negligible on-chip memory and all data-items (weights and activations) are stored off-chip and 2) sufficiently large on-chip memory and no off-chip accesses are required.

Case 1: Since all operands have to be fetched from off-chip memory, and there is no data reuse possible. The energy-efficiency depends on the memory-footprint, and hence, a DNN with larger memory-footprint would consume more energy as compared to a DNN with smaller memory-footprint. For example, despite having very low data reuse, compact DNNs such as MobileNet-V1 and MobileNet-V2 would be more energy efficient than AlexNet which has better data reuse (Table VIII). Note that the fragmented memory-accesses in DWConv may affect the row-buffer hit counts in DRAM, however, the energy savings through row-buffer hits would be negligible as compared to the cost of off-chip accesses [27]. Thus, the impact of data reuse in DNNs will be very small on energy-efficiency.

Case 2: DNNs with high data reuse will have more accesses to the on-chip memory closer to compute engine and DNNs with low data reuse will have more accesses to the on-chip memory farther from the compute engine. Since, the data movements from smaller and closer (to compute engine) memory consumes less energy than that from larger and farther memory [8], [28], the energy efficiency of MACs in DNNs with higher data reuse will be higher than that in DNNs with low data reuse. In effect, the relative difference between the energy efficiency of MACs with higher and lower data reuse would remain unchanged from that discussed in our model. Hence, the correlation between α and energy efficiency would not be altered, and our model would be useful in this case.

There are other corner cases for which the α may need to be re-calibrated. For example, when on-chip memory is very limited but able to accommodate the compact DNNs such as MobileNets, then despite of having poor data reuse behavior of DWConv, MobileNets will get benefits from cheaper on-chip accesses.

VI. The implication of design heuristics on energy efficiency of DNNs

Figure 5(a) shows EPP and MAC operations, whereas Figure 5(b) shows energy efficiency of DNNs. EPP and energy

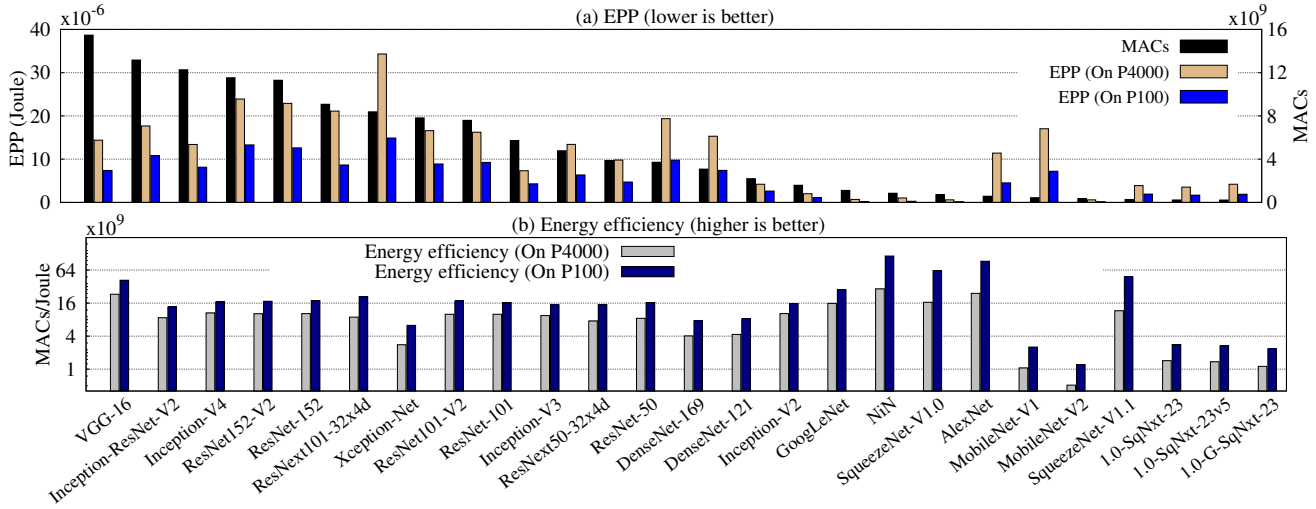


Fig. 5: (a) EPP and MACs (b) energy efficiency of 25 DNNs measured on P4000 and P100 GPUs (SqNxt=SqueezeNext)

efficiency values are measured experimentally on P4000 and P100 GPUs. As shown in Figure 5(a), EPP of XceptionNet, MobileNet-V1, and MobileNet-V2 is much higher than the EPP of their neighboring DNNs with higher MACs since energy efficiency of these DNNs are much lower (Figure 5(b)). These DNNs employed DWConv to reduce the M_c, however, due to very low data reuse in DWConv it increases the bandwidth pressure and reduces the energy efficiency. Similarly, the feed-forward connections in exacerbate the low activation reusability in XceptionNet and MobileNet-V2. Thus, the former has highest EPP and the latter has lowest energy-efficiency among 25 DNNs (Figure 5)

VII. Generality of Our Model

For generality test, we use confidence intervals for the population correlation coefficient (ρ), which measure the linear correlation between two variables over entire population. Whereas sample correlation coefficient (r) is a measure of the correlation between the two variables over a sample (ϕ) taken randomly from the population. We perform the following steps to compute the confidence intervals for given r and ϕ.

Step 1: "Central limit theorem" can be applied when the data follow normal distribution or the sample size is large. In our experiment, ϕ is 25 which is large enough (refer Eq. 6) to apply "central limit theorem". To get normal distribution, we transform r using "Fisher's Z transform" as $Z_r = \frac{1}{2} \times \log_e\left(\frac{1+r}{1-r}\right)$ [29]. Then, we calculate standard error (S_e) which is approximated as $\frac{1}{\sqrt{\phi-3}}$ [29].

Step 2: For 95% confidence, the upper limit (U_r) and lower limit (L_r) of confidence intervals are $U_r = Z_r + (1.96 \times S_e)$ and $L_r = Z_r - (1.96 \times S_e)$ respectively. Similarly, for 99% confidence, $U_r = Z_r + (2.58 \times S_e)$ and $L_r = Z_r - (2.58 \times S_e)$ respectively [29]. Note that these confidence intervals are corresponding to Z_r.

Step 3: We compute inverse Fisher transform [29] to calculate the upper (U) and lower (L) limit of confidence intervals corresponding to r, which are $U = \frac{e^{2\times U_r}-1}{e^{2\times U_r}+1}$ and $L = \frac{e^{2\times L_r}-1}{e^{2\times L_r}+1}$.

Observations: Table IX shows the confidence interval for both 95% and 99% confidence. For better approximation of

TABLE IX: Confidence intervals for ρ.

GPU	Metric	95% confidence L	95% confidence U	95% confidence Δ	99% confidence L	99% confidence U	99% confidence Δ
P4000	AI_c	0.16	0.76	0.60	0.03	0.81	0.78
	DI (**Ours**)	0.65	0.92	**0.27**	0.57	0.94	**0.37**
P100	AI_c	-0.18	0.57	0.75	-0.30	0.66	0.96
	DI (**Ours**)	0.33	0.83	**0.50**	0.21	0.86	**0.65**

ρ using r, window size (Δ) for confidence interval, should be as narrow as possible. Smaller window size implies lesser deviation in ρ and ensures that same correlation would hold in other population also.

Observation 1: For AI_c on P4000 GPU, at 99% confidence, L=0.03 and Δ=0.78. On P100 GPU, these values are L= -0.30 and Δ=0.96. This shows that, AI_c can have very poor correlation with energy efficiency in some cases, e.g., for AlexNet, relative disparity (d_f) is very high (Table VIII). By comparison, with DI at 99% confidence, L=0.57 and Δ=0.37 on P4000 GPU, whereas L=0.21 and Δ=0.65 on P100 GPU. Thus, it can be said with 99% confidence that ρ lies between 0.57 to 0.94 and between 0.21 to 0.86 on P4000 and P100 GPU, respectively. Clearly, *our model exhibits better positive correlation with energy efficiency in any population of DNNs on both GPUs.*

Observation 2: For DI, the upper limit (U) at 95% and 99% confidence are well above 0.80. Hence, in the best case, i.e., when ρ approaches U, DI shows a very strong positive correlation with energy efficiency. Evidently, our model is generic and applies to a large range of DNNs. Hence, it can be used as a DNN design heuristic.

Minimum ϕ required for generality test: Unlike Δ, window size corresponding to Z_r, i.e. $\Delta_r = U_r - L_r$, is independent of r. We find minimum sample size (ϕ) such that $\Delta_r \leqslant 1$ at 95% confidence.

$$\Delta_r \leqslant 1 \implies 2 \times 1.96 \times \frac{1}{\sqrt{\phi-3}} \leqslant 1 \implies \phi \geqslant 18.37 \quad (6)$$

Thus, the minimum number of DNNs required for generality test is 19. We have taken ϕ as 25 for which Δ_r = 0.836.

VIII. Conclusion and Future Work

We propose a novel model which decouples the weight and activation reuse and accounts for their unequal importance. We

show that our model is applicable to a diverse set of DNNs and is a better representative of energy efficiency in DNNs. In future work, we will evaluate our model on CPUs, FPGAs, and other accelerators to show its applicability over a large range of hardware platforms.

REFERENCES

[1] A. Krizhevsky *et al.*, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems 25*, 2012.

[2] K. Simonyan *et al.*, "Very deep convolutional networks for large-scale image recognition," *CoRR*, vol. abs/1409.1556, 2014.

[3] C. Szegedy *et al.*, "Inception-v4, inception-resnet and the impact of residual connections on learning," in *AAAI*, 2017.

[4] K. He *et al.*, "Identity mappings in deep residual networks," in *ECCV*, 2016.

[5] F. N. Iandola *et al.*, "Squeezenet: Alexnet-level accuracy with 50x fewer parameters and $<$ 1mb model size," *CoRR*, vol. abs/1602.07360, 2016.

[6] A. G. Howard *et al.*, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," *CoRR*, vol. abs/1704.04861, 2017.

[7] M. Sandler *et al.*, "Mobilenetv2: Inverted residuals and linear bottlenecks," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, June 2018.

[8] A. Pedram *et al.*, "Dark memory and accelerator-rich system optimization in the dark silicon era," *IEEE Design Test*, vol. 34, no. 2, pp. 39–50, April 2017.

[9] M. Harris, "Mapping computational concepts to gpus," in *SIGGRAPH*, 2005.

[10] S. Williams *et al.*, "Roofline: An insightful visual performance model for multicore architectures," *Commun. ACM*, vol. 52, no. 4, Apr. 2009.

[11] J. W. Choi *et al.*, "A roofline model of energy," in *IPDPS*, 2013.

[12] M. Ghane *et al.*, "Power and energy-efficiency roofline model for gpus," *CoRR*, vol. abs/1809.09206, 2018. [Online]. Available: http://arxiv.org/abs/1809.09206

[13] T.-J. Yang *et al.*, "A method to estimate the energy consumption of deep neural networks," *ACSSC*, pp. 1916–1920, 2017.

[14] D. Jung *et al.*, "Restructuring batch normalization to accelerate CNN training," *SysML*, vol. abs/1807.01702, 2019.

[15] L. Lai *et al.*, "Not all ops are created equal!" *SysML*, vol. abs/1801.04326, 2018.

[16] M. Lin *et al.*, "Network in network," *CoRR*, vol. abs/1312.4400, 2013.

[17] C. Szegedy *et al.*, "Going deeper with convolutions," in *CVPR*, June 2015.

[18] C. Szegedy *et al.*, "Rethinking the inception architecture for computer vision," in *CVPR*, June 2016.

[19] S. Xie *et al.*, "Aggregated residual transformations for deep neural networks," *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[20] A. Gholami *et al.*, "Squeezenext: Hardware-aware neural network design," in *CVPR Workshops*, 2018.

[21] F. Chollet, "Xception: Deep learning with depthwise separable convolutions," in *CVPR*, July 2017.

[22] K. He *et al.*, "Deep residual learning for image recognition," in *CVPR*, 2016.

[23] G. Huang *et al.*, "Densely connected convolutional networks," in *CVPR*, July 2017.

[24] Y. Jia *et al.*, "Caffe: Convolutional architecture for fast feature embedding," in *Proceedings of the 22Nd ACM International Conference on Multimedia*, 2014.

[25] http://www.nvidia.in/object/tesla-p100-in.html, 2012.

[26] "P4000 data sheet," 2018. [Online]. Available: https://bit.ly/2r1vBFn

[27] A. N. Udipi *et al.*, "Rethinking dram design and organization for energy-constrained multi-cores," in *Proceedings of the 37th Annual International Symposium on Computer Architecture*, ser. ISCA '10. New York, NY, USA: ACM, 2010, pp. 175–186. [Online]. Available: http://doi.acm.org/10.1145/1815961.1815983

[28] "On-chip energy consumption." [Online]. Available: https://bit.ly/2UaFlON

[29] B. V *et al.*, "Statistics review 7: Correlation and regressions," *Critical Care*, 2003.