Characterizing Sources of Ineffectual Computations in Deep Learning Networks

Miloš Nikolić*, Mostafa Mahmoud*, Yiren Zhao†, Robert Mullins† and Andreas Moshovos*
*The Edward S. Rogers Sr. Department of Electrical and Computer Engineering
University of Toronto, Toronto, Canada
milos.nikolic@mail.utoronto.ca, mostafa.mahmoud@mail.utoronto.ca, moshovos@ece.utoronto.ca
†Department of Computer Science and Technology
University of Cambridge, Cambridge, United Kingdom
yaz21@cam.ac.uk, robert.mullins@cl.cam.ac.uk

***Abstract*—Hardware accelerators for inference with neural networks can take advantage of the properties of data they process. Performance gains and reduced memory bandwidth during inference have been demonstrated by using narrower data types [1] [2] and by exploiting the ability to skip and compress values that are zero [3] [4] [5] [6]. Similarly useful properties have been identified at a lower-level such as varying precision requirements [7] and bit-level sparsity [8] [9]. To date, the analysis of these potential sources of superfluous computation and communication has been constrained to a small number of older Convolutional Neural Networks (CNNs) used for image classification. It is an open question as to whether they exist more broadly. This paper aims to determine whether these properties persist in: (1) more recent and thus more accurate and better performing image classification networks, (2) models for image applications other than classification such as image segmentation and low-level computational imaging, (3) Long-Short-Term-Memory (LSTM) models for non-image applications such as those for natural language processing, and (4) quantized image classification models. We demonstrate that such properties persist and discuss the implications and opportunities for future accelerator designs.**

***Index Terms*—Deep Learning Acceleration, Ineffectual Work, Precision, Sparsity**

I. Introduction

Neural networks typically require billions of operations to perform a forward pass for just one input. Graphics processors are well suited for this high, yet regular, computation demand, as it also exhibits high data-parallelism. However, as Dennard-scaling has ceased [10] and Deep Learning applications proliferate, interest in specialized hardware accelerators for neural networks has been rapidly increasing [11] [12].

Early accelerators were able to obtain significant gains from specialized functional units, exploiting data-flows to maximize data reuse, and employing hardwired control. Follow up work targets additional properties of neural networks that lead to ineffectual computations. Table I shows some of the popular hardware designs that exploit three significant properties: (1) numerical precision requirements, (2) activation and weight sparsity, and (3) bit-level sparsity. Typically these accelerators and the associated optimizations are demonstrated for Convolutional Neural Networks (CNNs) performing image classification.

TABLE I: Hardware accelerators and sources of inefficiencies exploited. Act and Wgt are for activations and weights.

Accelerator	Properties	Target	Granularity
Cnvlutin [6]	Value Sparsity	Act	
Cambricon-X [13]	Value Sparsity	Wgt	
SCNN [5]	Value Sparsity	Wgt+Act	
EIE [14]	Value Sparsity	Wgt+Act	
Eyeriss [12]	Value Sparsity	Act	
Minerva [15]	Value Sparsity	Wgt+Act	
Stripes [1]	Precision	Act	Layer
Dynamic Stripes [16]	Precision	Act	Group
Dpred [7]	Precision	Act	Group
TPU V1 [17]	Precision	Wgt+Act	Layer
BitFusion [2]	Precision	Wgt+Act	Layer
Loom [18]	Precision	Wgt+Act	Layer
Bit Pragmatic [8]	Bit Sparsity	Wgt+Act	
Laconic [19]	Bit Sparsity	Wgt+Act	
Bit Tactical [9]	Precision, Bit & Value Sparsity	Wgt+Act	Group

While a broad range of optimizations based on the properties of weights and activations have led to promising results, Deep Learning is evolving rapidly with innovations in model architecture, techniques and applications. Accordingly, it is uncertain whether these gains will persist for more recent models that are generally more accurate and less overprovisioned than older models. Taking for example image classification models, while AlexNet had less than 10 layers and 3 fairly large fully-connected layers, more recent models have several tens of layers and fewer, much smaller, fully-connected layers. Accordingly, a question relevant to hardware accelerator design is whether the aforementioned three properties still hold and to what degree for more recent models and different input datasets. Beyond image classification there are many other applications and other network types that are equally important and demanding. For example, while image classification models that are widely used in the evaluation of hardware accelerators have been designed to work with the relatively low resolution ImageNet [20] dataset (image resolution is about 230×230), other convolutional neural networks such as those performing computational imaging (e.g., in-painting, de-noising and super-resolution) operate on much higher resolutions. In addition, natural language processing tasks tend to use recurrent or

LSTM models. Accordingly, another relevant question for hardware designers is whether these models exhibit any of these properties.

The goals of this work are: (1) to study whether sparsity, variable precision requirements, and bit-sparsity are exhibited and to what degree for inference of a broader set of neural network models, and (2) to investigate, where possible, how they evolved over time. The latter is made possible when there are models targeting the same application but which were developed at different points in time. As Table II shows, we study more recent CNNs for image classification, CNNs for other applications such as segmentation, object detection, computational imaging operating on higher resolution images, and models other than CNNs for natural language processing. Some models are optimized for computational efficiency, e.g., MobileNet, while others for best-of-class accuracy. e.g., ResNet. We also consider the effects of quantization on a subset of these models.

In summary, we find that the aforementioned properties persist, albeit to a different degree across *all* models studied. This serves as motivation for further developing hardware accelerators that exploit them. In addition, we identify several challenges that such accelerators will face depending on the model or application domain. For example, certain models generally require higher precision than the prevailing 16b or 8b fixed-point representations used in many CNN accelerators. The key observations made are:

- Reduced precision, value sparsity and bit sparsity are exhibited by all models: CNNs, LSTMs and quantized models.
- The precision required varies per layer and dynamically at finer granularity for all models.
- Newer models tend to require higher precision.
- All convolution and fully-connected layers require less than 16b of precision even when this is detected via profiling at the layer granularity. The number of bits needed varies significantly between 2 and 13.
- All but one of the networks can maintain accuracy when the maximum supported precision is 16b. For one LSTM network, 23b are required for one of its LSTM layers.
- In all layers studied, there are few values that greatly skew the precision needed per layer. The precision needed for most values is much shorter.
- For most networks, reducing the maximum precision to 8b does not preserve accuracy necessitating further effort, such as retraining or re-architecting the model to recover the accuracy loss.
- Reducing precision amplifies all sparsity properties. The number of non-zero values and bits is reduced by truncating a suffix of least significant bits, essentially containing noise, without affecting accuracy.
- LSTM layers exhibit much less activation sparsity with only marginal weight sparsity compared to other layers. Both properties are only modestly affected by reducing precision.
- More recent networks offer less potential gains than older ones, however, the potential remains significant.
- Value sparsity for weights and activations varies considerably across networks. Generally, the image classification networks exhibit both. For some computational imaging networks there is not as much sparsity.
- Bit-level sparsity extends beyond image classification networks and is consistently high for all networks and thus remains a potential target for accelerators.
- There are no general trends that can be observed that link the various properties to the relative position of the layer within the network. In some networks, later layers tend to require less precision and exhibit higher sparsity and bit-sparsity, but this trend is reversed or non-existent in other models.
- Activation functions impact sparsity. However, the property may exist even for networks that do not use the ReLU activation function especially after reducing precision.
- The choice of the weight pruning method, where applicable, greatly affects all aforementioned properties.
- Compared to per-layer profiled networks, 8b quantization reduces the potential for activation and dynamic precision. However, bit-sparsity remains abundant.
- Extreme 2b quantized networks exhibit less potential for value- and bit- level activation sparsity. While the dynamic precision is reduced to around 1.5 bits, the gain is relatively smaller compared to per-layer profiled networks. However, value- and bit-sparsity remain high.

II. Network Properties and Benefits

All accelerators exploit the embarrassingly parallel and predictable computation schedule of neural networks, e.g., [11] [17] [12]. In order to further increase performance and energy efficiency, more recent accelerators also exploit properties which have only been demonstrated for image classification CNNs [1] [5] [16] [6] [4] [45] (the exception is forced weight sparsity which is induced via pruning [4]). These properties allow for benefits beyond those gained through parallelization, memory access reuse, or scaling. The properties can be inherent to the general structure, forced during training or through choice of the network architecture. Additionally, these can be profiled statically for a potentially simpler design, or dynamically for greater benefits. A survey of common properties and their effect follows.

A. Value Data-Type/Precision

A common approach in hardware accelerators is to use 16b fixed-point arithmetic and some recent accelerators also support shorter data-types such as 8b, or even 4b, fixed point or 16b floating point. This is contrary to the 32b floating-point arithmetic used in older graphics processors. 16b fixed-point arithmetic has been shown to be more than adequate for inference with CNNs and the conversion of most networks trained using 32b floating-point arithmetic to 16b fixed point is generally considered straightforward. Further reduction in precision has been possible via profiling without, or if desired with controlled, loss of accuracy. Accordingly, the number of bits used for activations or weights may vary per network, per layer, or even per selected group of values. Quantization can

TABLE II: Networks profiled with their application, dataset and number of inputs profiled for precision and all other properties

Network	Task	Dataset	Precision profile	Properties profile	Remarks
CaffeNet [21]	Classification	ImageNet [20]	5000	1000	Baseline
GoogleNet [22]	Classification	ImageNet [20]	5000	1000	Baseline
ResNet18 [23]	Classification	ImageNet [20]	5000	1000	Baseline
SK Caffenet [22]	Classification	ImageNet [20]	5000	1000	Pruned by SkimCaffe
SK GoogleNet [22]	Classification	ImageNet [20]	5000	1000	Pruned by SkimCaffe
SK Resnet [22]	Classification	ImageNet [20]	5000	1000	Pruned by SkimCaffe
ES Alexnet [24]	Classification	ImageNet [20]	5000	1000	Pruned by Eyeriss
ES Googlenet [24]	Classification	ImageNet [20]	5000	1000	Pruned by Eyeriss
SqueezeNet [25]	Classification	ImageNet [20]	5000	1000	Newer and Mobile
MobileNet-V1 [26]	Classification	ImageNet [20]	5000	1000	Newer and Mobile
Q AlexNet [27]	Classification	ImageNet [20]		256	Quantized
Q MobileNet-V1 [27]	Classification	ImageNet [20]		256	Quantized
HWGQ AlexNet [28]	Classification	ImageNet [20]		256	Quantized
HWGQ ResNet18 [28]	Classification	ImageNet [20]		256	Quantized
HWGQ VGG [28]	Classification	ImageNet [20]		256	Quantized
SegNet [29]	Segmentation	CamVid [30]	500	100	Computational Imaging
YOLO V2 [31]	Detection	Pascal VOC [32]	1000	200	Leaky ReLU
Berkeley FCN8 [33]	Segmentation	Pascal VOC [32]	100	50	Computational Imaging
VDSR [34]	Super Resolution	CBSD68 [35]+LIVE1 [36] +Set5 [37]+Set14 [38]	114	10	Computational Imaging
IRCNN 25 [39]	Image De-noising	CBSD68 [35] +Set5 [37]+Set14 [38]	108	50	Computational Imaging
Seq2Seq [40]	Translation	WMT14 FR-EN	5000	500	LSTM
LRCN [41]	Captioning	COCO [42]	5000	1000	LSTM
Bi-dir. LSTM [43]	Captioning	Flickr8k [44]	5000	1000	LSTM

also reduce precision with 8b generally being demonstrated for state-of-the-art image classification networks. In some cases even 4b is sufficient [46]. However, quantization generally requires retraining and possibly re-architecting the network. Because of this additional development effort, quantized models of neural networks are not as broadly available. For the purposes of this work, our approach is to investigate precision reduction at finer levels targeting flexible accelerators with as little of extra work as possible for the network designer. However, since sparsity and bit-sparsity can in principle be prevalent even on heavily quantized models, we also present an analysis of some of the properties for a smaller set of quantized models that was available to us. Finally, there are other floating-point inspired formats that may be used. While providing higher dynamic range, such formats generally incur some overhead compared to fixed-point formats. We leave the analysis of the targeted properties for models using floating-point representations for future work noting that such models are not as widely available as fixed-point ones and that whether and how to exploit some of the properties studied here with such representations has yet to be demonstrated.

Reducing the number of bits used by activations and weights can yield two main benefits:

1) Traffic and Storage Reduction: Using fewer bits per value, especially if applied to large groups like layers, provides a simple and effective compression. Many accelerators store convolution kernels on-chip for reuse. With smaller size of weights, more can fit in on-chip memory reducing expensive off chip accesses. A similar approach can reduce activation footprint and communication needs.

2) Speedup: Accelerators using bit serial computation have shown that with enough parallelism it is possible to achieve speedup inversely proportional to the size of the operands. This can be applied to activations or weights.

There are two secondary considerations related to reducing the data-type size:

3) Granularity: The granularity of precision groups has major effects on the benefits from reduced precision. Finer groups provide larger performance potential since it is expected that few activations and weights will be of high magnitude [7], [16]. However, finer grain precision adaptivity requires more capable hardware, incurs higher metadata overhead (precision needs to be specified per group) and produces more imbalance across groups.

4) Dynamic vs. Static: The weights are known in advance and the best precision can be chosen statically. Conversely, the activations are computed on the fly and ideal precision requirements will vary depending on the inputs. Profiling the networks and setting the precisions statically is simpler, but must support the worst case and thus overemphasizes the few high magnitude activations. Dynamic precision selection according to the input provides better potential gains at the cost of additional hardware support.

B. Value Sparsity

Another property exploited by accelerators is sparsity in weights and activations, which is the fraction of those values that are exactly zero. Whenever at least one of the operands is zero, the multiplication result will be zero and the whole calculation can be skipped. In the simple case multipliers can be turned off to save energy [12] [15], or with some extra hardware these multipliers can be used to execute some other useful operations. Furthermore, if the number of zeros is large enough, there is a substantial opportunity for compression [12] [15] [14]. Specialized accelerators have been

designed to exploit sparsity in weights, activations or both with varying hardware complexities. There are two sources of sparsity:

1) Weights: Weight sparsity is usually induced artificially through pruning. It has been shown that pruning can sometimes reduce the number of non-zeros weights down to 10-20% with small loss of accuracy [47]. Furthermore, pruning can work as a form of regularization, reducing over-fitting and improving the test accuracy [48]. Even though pruning has been most commonly used with CNNs, the same approach can be used for other network structures as well. More recent work on weight pruning has yielded lower levels of sparsity [22] [24].

2) Activations: Activation sparsity is most commonly induced by the choice of the activation function. The most common activation function used in CNNs has been the Rectified Linear Unit (ReLU) function. ReLU zeros out all negative activations and leaves positive ones unchanged. Additionally, small activation values can often be zeroed out without noticeable loss in accuracy. Unfortunately, ReLU is common only in CNNs and some recent works suggest that training speed improves with other activation functions that attenuate but not do zero-out negative values [49].

C. Bit Sparsity

One of the less explored value properties is bit sparsity. The core computation used by neural networks is the multiplication $A \times W$ of an activation A and a weight W. This operation can be decomposed into $\sum_{ij} W \times A_i$ where A_i the ith bit of A. The computations where A_i is zero are ineffectual. With the right choice of representations, image classification CNN models have been shown to have activations where less than 10% of their bits are "1" [8]. Using bit-serial multiplication units with the ability to skip zero bits boosts speedup and energy savings [8]. This approach can also be applied to weights [19].

D. Term Sparsity

Booth encoding can further reduce the number of bits that ought to be processed per activation or weight. In this case, a number is represented as a series of signed powers of two, or *terms*. Provided enough parallelism exists, accelerators have been designed to speedup calculation inversely proportional to the number of terms using term-serial multiplication [8]. Due to space constraints, we do not report measurements on term sparsity. However, we show that all networks exhibit high bit sparsity, and thus high term sparsity; term sparsity is at least as high as bit sparsity.

III. Methodology

The Caffe deep learning framework was used for the analysis of all networks [21]. Table II details the networks, datasets and input counts evaluated. The networks were obtained as out-of-the-box pre-trained models as released by the original author(s) except for the LSTM networks. These networks were trained using the source code and the hyper-parameters provided by the original author(s) as pre-trained models are not provided. All models were originally trained to operate on 32b floating-point values. However, where possible, we first convert the models to use a 16b fixed-point representation. The rest of this section explains how we collected our measurements per studied property.

Value Data-Type/Precision: Static per-layer reduced precisions were determined first and then reduced-precision values were used to study all other properties. The precisions were selected only by profiling without any retraining. We chose to first study profile-based precisions as this would impact some of the other properties. For example, a 16b non-zero value may end up being zero when truncated and represented with fewer bits. Thus, reducing precision is expected to amplify the other various forms of value sparsity.

Static Precision Selection: The per-layer precisions are calculated for each network in 3 steps. In the first step each layer is profiled individually, for weights and activations separately, to maintain the TOP-1 accuracy with the smallest number of bits while all other layers are unrestricted and set to 32b floating point. From this point on, we consider the network with all precisions set, for both weights and activations, according to step 1. When all layers are combined with the precisions selected in step 1, TOP-1 accuracy suffers considerably. In step 2, we gradually increase per-layer precisions to recover accuracy. Specifically, we probe each layer with an extra bit of precision and select the layer that leads to the highest gain in accuracy. We repeat this until we recover the target accuracy. Finally, we probe each layer, remove one bit of precision, and select the layer that leads to the lowest loss of accuracy. We repeat this procedure until no more bits can be removed while maintaining the target accuracy. Although this appraoch does not necessarily arrive at an optimal precision profile, it is a reasonable greedy technique that avoids exhaustive search. In this paper the target accuracy is set to be within 1% (relative) of the original TOP-1 accuracy on a subset of the original dataset. This profiling technique allows us to determine precisions per layer unlike quantization methods that target using a specific precision for all layers.

Dynamic Precision Adjustment: The precision selected via profiling is generally pessimistic. The precision has to accommodate any possible input and all possible values across a layer. Generally, the distribution of values within each layer is not expected to be uniform. Most values tend to be near zero and only a few exhibit a large magnitude. A profile-based precision has to accommodate all and hence overemphasizes the few high magnitude ones [7]. An alternative is to adjust precision on the fly as values are processed. This dynamic precision is calculated as the number of bits required to represent each value. For example, the value $+1$ requires 2 bits, while the value -7 needs 4 bits. In the interest of generality, we measure dynamic precision requirements on an individual value basis. However, a practical design will have to adapt precision at a coarser granularity of a group of n values with $n > 1$ [7], [16]. The decision on which group size is appropriate depends on the specific design.

Value Sparsity: Value sparsity is calculated as the ratio of zero-

valued weights or activations. Some of the small values in 32b floating point will become zero when converted into the selected fixed point format. Sparsity is measured for the 16b baseline and for the profiled static precision. Since pruning generally boosts sparsity, we also include pruned models. However, pruning often requires re-training which represents a considerable time overhead.

Bit Sparsity: Bit sparsity is calculated as the ratio of the number of "0" bits to the total number of bits for the 16b baseline and the profiled static precision.

Property Weighing: Depending on the goal, the aforementioned per-value properties need to be scaled accordingly to reflect their potential contribution. When considering memory footprint, the properties are weighted according to the number of values in each activation or weight matrix. This also serves as a proxy for memory traffic. The actual memory traffic depends on the dataflow and the memory hierarchy used and hence is best studied given a specific hardware design and computation schedule. When considering execution time, the properties are weighted according to the number of runtime calculations involved in each layer.

IV. Results and Discussion

We present measurements for the aforementioned properties in the order described in the previous section. For each property, we first present aggregate results for the convolution, fully-connected, and LSTM layers alone and then the aggregate results for the whole network. Finally, we also show per layer results. In all cases we present two sets of results weighted according to memory footprint or computation usage. Once we cover all aforementioned properties, we measure the potential for work reduction for these properties.

A. Value Data-Type/Precision

We report the number of bits required for convolution, fully-connected, LSTM layers and the full network in Figure 1. The figure reports the precision range for all layers, as well as weighted averages for estimating performance and memory footprint. The figure shows dynamic per-value requirements as well. The ideal performance improvement is estimated as b/n, where b and n are the baseline and required number of bits respectively. The ideal memory footprint per value is estimated to be the calculated average required number of bits.

1) Static Per-Layer Precision: Considering the profile-derived per layer precisions we observe: (1) All but one of the layers require less than 16b. (2) The number of bits when using static precisions vary significantly between 2 and 13 justifying the need for a variable, and potentially fine grain selection of precision bits. (3) Without retraining, an 8b representation is not sufficient to maintain accuracy. (4) Newer networks generally require higher precisions. This observation is highlighted in Figure 3 showing representatives of different years and their precision requirements for weights and activations. As an example, for ResNet18 and MobileNet, representatives from 2015 and 2017, the number of bits required increased by roughly 2 and 4 bits for weights and activations, respectively. (5) On average, reducing precision can reduce the memory footprint of weights and activations to 6.1 and 6.6 bits per value, respectively. (6) Assuming that execution time is proportional to the number of bits, the potential for performance improvement is 2.25× and 2.43×, respectively for weights and activations. (7) Since there appears to be no inherent rule whether activations or weights will require less bits, a more flexible, potentially low overhead, approach which dynamically chooses whether to serially execute weights or activations can further increase the performance gain to 2.56×.

Figure 2 shows the number of bits required to represent weights and activations per layer for a subset of networks. Even though there is a weak downward trend in number of required bits in the latter layers, there appears to be no general rule. This trend is more pronounced in shallower networks like CaffeNet. LRCN shows the opposite, with requirements increasing in the latter layers. Some are constant like ES GoogleNet while others like SK ResNet fluctuate around a constant mean.

Figure 4 considers pruning and compares the precision of CaffeNet, Alexnet and GoogleNet for pruned (SK and ES) and baseline models. CaffeNet is nearly identical to Alexnet in terms of architecture with the only difference being that the order of two of the layers is switched. Pruning has a noticeable effect on precision, however there is no general rule whether pruning increases or reduces the precision needed. Comparing SK and ES networks illustrates that the pruning algorithm can greatly affect precision.

Figure 1 shows that some of the fully-connected layers require noticeably higher precisions compared to the preceding convolution layers (e.g., SK ResNet). However, in general there is no clear trend suggesting that the information capacity needed by each layer varies in network-specific ways that are hard to discern and generalize. When there are multiple fully-connected layers, generally, the last one tends to have higher precision requirements (e.g., LRCN).

One LSTM layer of Seq2Seq requires 23b weights to maintain accuracy. Using 16b values reduces Seq2Seq's accuracy by more than a third. We only evaluated a common fixed point format for all weight matrices in a layer. A finer approach for each matrix, or a floating-point representation might be more suitable for these layers. Future accelerators that wish to exploit precision for Seq2Seq type of networks may opt instead to support extended precisions by implementing composable data representations where a higher precision calculation is decomposed into a number of lower precision ones. Whether and how this can be done is an open question. Bit Fusion supports 16b values using 8b processing elements [2].

All other LSTM layers need considerably fewer bits, comparable to convolution layers. As a result, using variable precision will have the greatest impact, especially if the baseline is changed to 24 or 32 bits in order to support the outlier layer. It is worth noting that the number of bits required for Seq2Seq activations and weights vary wildly. On the other hand, LRCN and Bi-directional LSTM networks appear to have a near constant number of required bits for weights and activations — the precision needed across layers varies by at

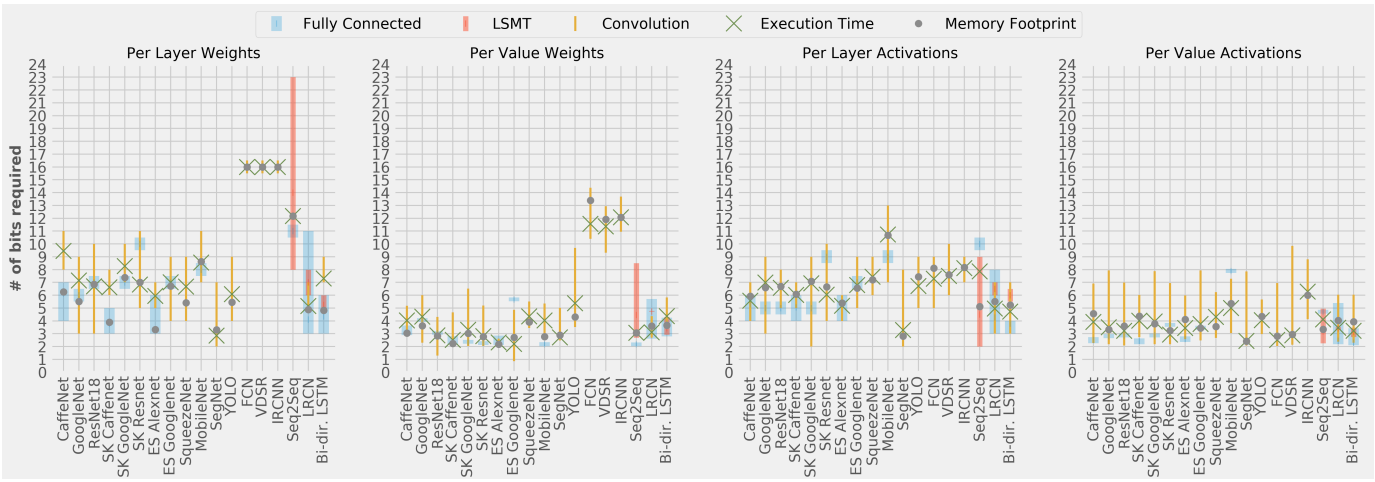

Fig. 1: **Precision:** Number of bits required to represent weights and activations.

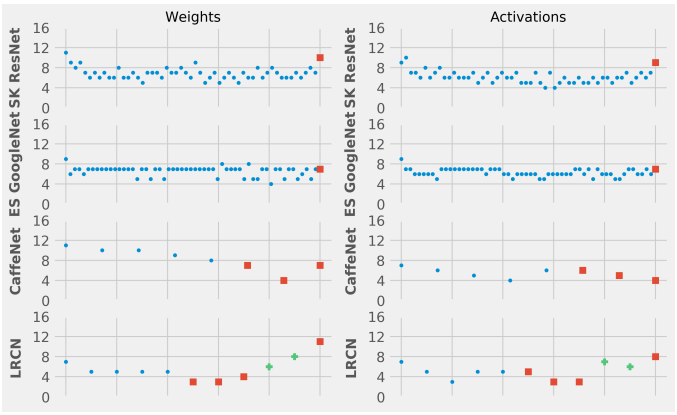

Fig. 2: **Per Layer Precisions of Selected Networks:** Layers are in topological order. ○: Convolution layers, □: fully-connected layers, ✚: LSTM layers.

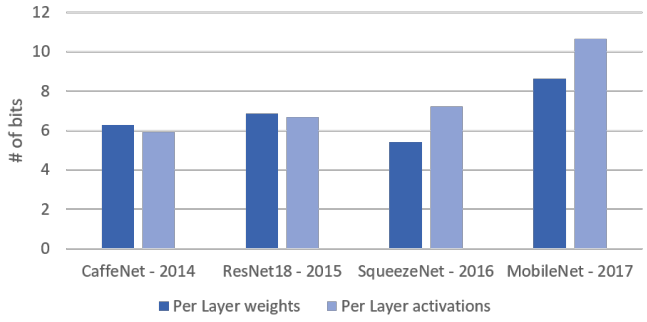

Fig. 3: **Required Precision over time:** Precision requirements of example networks from different years.

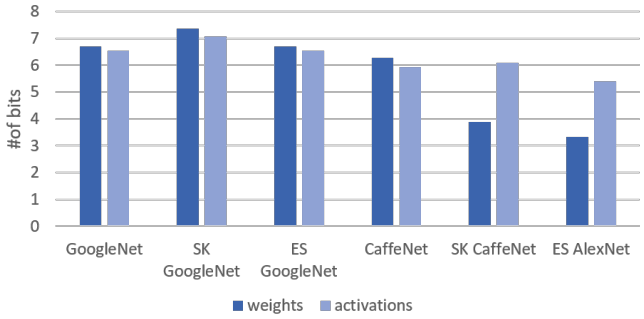

Fig. 4: **Required Precision for different pruning algorithms:** Weighted average number of bits for baseline and pruned versions of GoogleNet and CaffeNet.

most 1b. However, overall, we still observe that the precision requirements vary.

2) Dynamic Per-Value Precision: Average dynamic precision requirements for most networks are between 2 and 9 bits. This confirms that selecting per layer precision using profiling greatly overestimates precision needs and undermines the potential benefits. For example, if the precision can be set on-the-fly bit-serial computation over activation or weights has the potential to improve performance by 4.30× and 3.36×, respectively. If it is possible to choose on the fly whether to process the activations or the weights bit-serially then the performance potential is 5.15×. Storing values with dynamic precision can also potentially reduce the memory footprint to 3.84 and 5.14 bits per activation and weight, respectively. Similarly to static per-layer profiles, a weak trend toward higher precision for newer networks can be seen. Finally, as with static precision, pruning and the algorithm used for pruning also influence dynamic precision requirements.

B. Value Sparsity

As with precision, we present value sparsity measurements per layer type and for the whole network in Figure 5. We

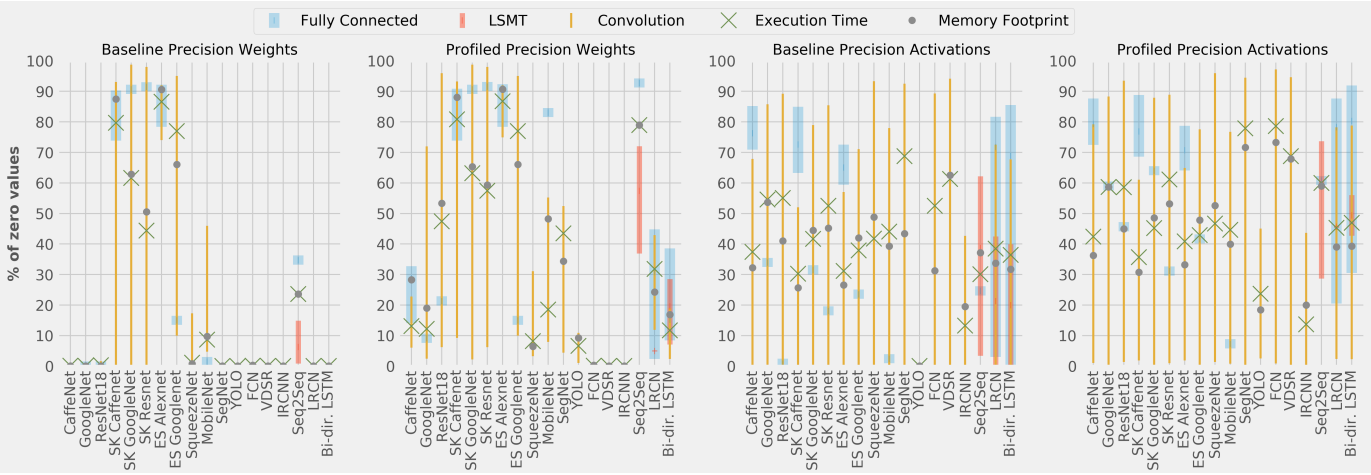

Fig. 5: **Value Sparsity:** Weighted average of number of zero values in convolution layers for activations and weights.

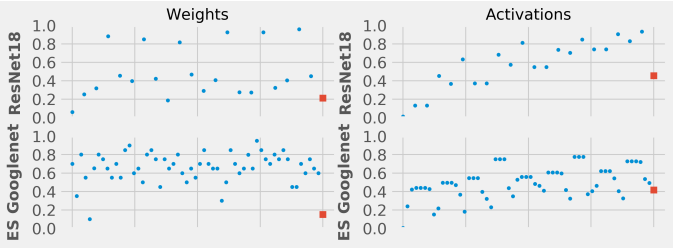

Fig. 6: **Per Layer Value Sparsity of Selected Networks:** Layers are in topological order, ○: Convolution layers, □: fully-connected layers.

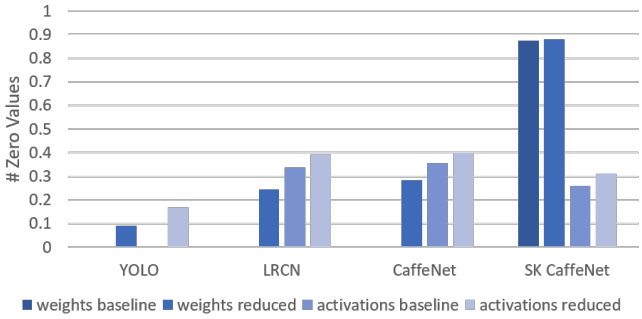

Fig. 7: **Sparsity:** Increase in sparsity after reducing precision.

report sparsity when values are represented using the original 16b precision (full) and after truncating to the profiled per layer precision described above. We estimate the ideal memory footprint as $mem = b \times (1-s)$, and performance potential as $perf = 1/(1-s)$, where b is the baseline precision and s is ratio of zero values.

1) Full 16 bit precision: Most models exhibit significant activation sparsity suggesting that this is not only a property of image classification CNNs. Activation sparsity is very common across all layers proceeded by a ReLU activation function since ReLU zeros out all negative values. The outlier is IRCNN which performs denoising. The activation values for this network closely follow the original pixel values throughout. Activation sparsity is negligible for YOLO which uses the leaky ReLU function which only scales down negative numbers. The potential performance gain of skipping zero activations, excluding YOLO, in full precision networks is 1.75× whereas the average memory footprint can be reduced to 9.81 bits.

Unlike convolution and fully-connected layers LSTMs do not use ReLU. The most common activation functions in LSTMs are sigmoid and hyperbolic tangent functions. However, Figure 5 still shows a noticeable percentage of activations that are zero even when represented in the full 16b precision.

Weight sparsity doesn't appear naturally in most networks. As a result non-pruned networks have negligible potential performance gain of 1.03×, while pruned ones go up to 3.32×. Memory footprint of pruned weights can be reduced to 4.56 bits per value. The choice of the pruning algorithm does impact weight sparsity as well.

Figure 6 shows sparsity per layer for a subset of networks for the 16 bit precision. The figure shows that there is no trend on weight sparsity depending on layer position, while some networks show increased activation sparsity in latter layers.

2) Profiled reduced precision: With the reduced precision, however, many more activations and weights become zero for most networks, even for YOLO. Figure 7 shows how sparsity in networks can by amplified by reducing precision. This effect is caused by zeroing out smaller values below the threshold of the lowest precision available. For weights, this approach works only for non pruned layers, since all small zeros are already zeroed out in pruned versions. Still, sparsity varies considerably even within each network. The performance gains are then increased to 1.26× and 3.71× by reducing precision, for non pruned and pruned networks, respectively. Finally, the average memory footprint can be reduced to 0.75× and 0.26× with reduced precision.

On the activation side performance can be increased by 1.31× and 2.04×, for YOLO and other networks respectively.

C. Bit Sparsity

For this subsection we assume that the values are stored as sign bit + absolute value, and we only consider the absolute value. Figure 8 shows bit sparsity for each layer type and over whole networks. Additionally, Figure 9 shows bit sparsity per layer of each network. Bit sparsity is inherently linked to value sparsity and follows the same trends, while providing significantly more potential uncovering opportunities even when there is insufficient value sparsity.

Activation and weight bit sparsity are reported in Figure 8 for full and reduced precision. Even when values are represented in the full 16b precision, bit sparsity occurs naturally across all networks. It is especially pronounced in high sparsity layers, in weights of pruned networks and in activations of networks with ReLU activation. Bit-sparsity for weight yields a performance potential for 9.48× and 2.98× for pruned and dense networks, respectively. Activation bit-sparsity yields a performance potential of 2.86× and 4.74× for YOLO and the other networks respectively.

Similarly to value sparsity, reduced precision amplifies bit sparsity. With reduced precision, the performance potential is 35.76× for pruned networks and 10.16× for dense networks. With reduced precision there are no outliers anymore and all networks exhibit bit-sparsity well above 60%. The performance potential for activations is 17.86×. The image classification networks exhibit activation bit sparsity that approach or exceed 90% which corroborates past work [8] and suggests that the phenomenon persists even for newer networks. Bit sparsity varies for other networks, but overall remains high.

The results for the fully-connected layers, the LSTM layers, and the whole networks exhibit similar trends. Overall, bit-sparsity appears to be prevalent across all networks and all layers including the LSTMs. Reducing precisions first further boosts bit sparsity, however, in relative terms the improvement is lower compared to the other properties since bit sparsity is naturally high to start with. Furthermore, as expected it is considerably higher than value sparsity and thus has the highest potential among the properties we studied. As we noted earlier, Booth encoding the values would have resulted in higher term sparsity.

D. Effects of Quantization

In this section we consider the interaction of quantization, precision, and value- and bit-sparsity. While quantization has been receiving significant attention with advances being reported regularly, quantized models are usually not as broadly available. In view of these challenges we opted to consider two different quantization schemes, one that targets 8b [27] values which is representative of what is often considered today sufficient for neural models processing images and another that targets extreme quantization to 2b and 1b values [28]. All models studied are CNNs for image classification. Since the input is usually not quantized, in this section we do not study the first convolution layer.

1) 8b Quantization: First, we consider networks that were quantized to 8 bits [27]. Table III shows the average precision requirements, value, and bit sparsity levels for quantized AlexNet and MobileNet ('Q' rows). The table also reports the same characteristics for the profiled, per layer reduced precision 16b versions of these networks ('P' rows).

Even with 8b quantization the precisions required per value are much lower. Compared to the 16b models, the per value precisions with the 8b models are generally longer. In part this is an artifact of our methodology: we were not able to first adjust the precisions per layer through profiling since the quantized models are not presently compatible with our Caffe-based infrastructure. Adjusting precisions per layer removes some of the least significant bits that contain noise. Regardless the results show that sufficient potential for exploiting precision requirements exists even in the 8b models.

Sparsity is present also in the quantized models. Activation sparsity is marginally higher for the quantized models but it is lower for weights in MobileNet. The latter can be explained in part as an artifact of the lack of per layer precisions.

Finally, the 8b networks exhibit similar levels of bit sparsity as the profiled networks even though we were not able to take advantage of a per layer precision selection step first. Table III shows that activation bit sparsity is within 2% of the corresponding profiled models. On the weight side, the 8b quantized AlexNet exhibits 4% more bit sparsity compared to its 16b counterpart. For MobileNet bit sparsity drops by 11% with 8b quantization.

2) Binarized / 2b Quantization: Secondly, we consider networks that were quantized to 1b weights and 2b activations [28]. Since the weights are positive and negative values of the same magnitude, they always require 1b, and exhibit 0% value- and bit- sparsity. Consequently, we present only activation measurements. Note that this low precision scheme does come at a cost of roughly 10% relative loss in accuracy. Table IV shows the average precision requirements, value- and bit- sparsity levels for 1b/2b quantized AlexNet, ResNet18 and VGG, as well as equivalent 16b AlexNet and ResNet18 with profiled per-layer reduced precisions.

The quantized activations show very low dynamic precision requirements, almost at the minimal 1b threshold. Naturally, the relative gain over static precisions is lower compared to the profiled networks since the activations are already quantized down to 2b.

While value- and bit-sparsity are noticeably lower compared to the 16b models — bit-sparsity is rougly 20% less — they are still surprisingly high.

Finally, the three models exhibit very similar levels of each of the three properties. For example, all three exhibit bit-sparsity in the 71% to 74% range.

3) Quantization Summary: This brief analysis confirms that the properties observed in CNNs apply to quantized networks as well with similar results. It also shows that there are diminishing returns as more extreme quanization scheme is selected.

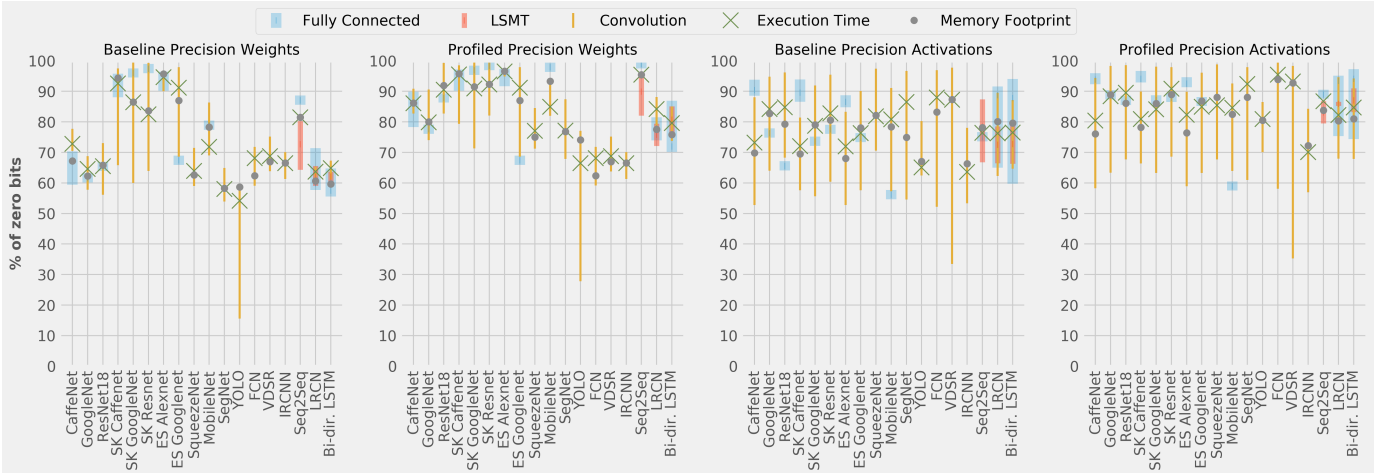

Fig. 8: **Bit Sparsity:** Weighted average of the number of zero bits for activations and weights.

TABLE III: Comparison of averaged properties in quantized and profiled precision networks. **P**: Profiled per layer precision 16b non-quantized models, **Q**: 8b quantized models

Type	Network	Weight Precision per Layer/Value	Activation Precision per Layer/Value	Weight Value Sparsity	Activation Value Sparsity	Weight Bit Sparsity	Activation Bit Sparsity
Q	AlexNet	8.00 / 4.08	8.00 / 3.60	0.28	0.57	0.89	0.88
P		8.30 / 3.54	5.10 / 2.86	0.17	0.68	0.85	0.90
Q	MobileNet-V1	8.00 / 5.43	8.00 / 4.47	0.11	0.37	0.77	0.79
P		8.80 / 3.55	10.5 / 5.45	0.30	0.40	0.88	0.81

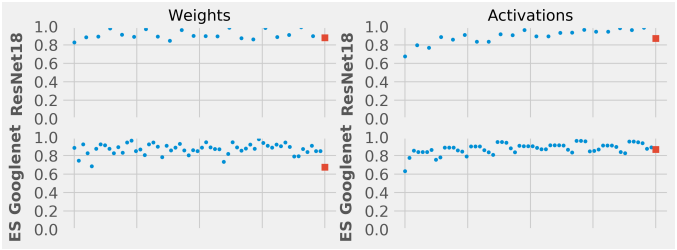

Fig. 9: **Bit sparsity per layer of selected networks:** Layers are in topological order. ○: Convolution layers, □: fully-connected layers.

TABLE IV: Comparison of averaged properties in HWGQ quantized and profiled precision networks. **P**: Profiled per layer precision 16b non-quantized models, **HWGQ**: HWGQ bit quantized models. The HWGQ networks do not achieve the full 32b float accuracy.

Type	Network	Activation Precision per Layer/Value	Activation Value Sparsity	Activation Bit Sparsity
HWGQ	AlexNet	2.00/1.46	0.54	0.72
P		5.10/2.86	0.68	0.90
HWGQ	ResNet18	2.00/1.46	0.52	0.71
P		6.33/2.98	0.6	0.90
HWGQ	VGG	2.00/1.45	0.56	0.74

E. Work Reduction Potential

Finally, we report the work reduction *potential* depending on the type of ineffectual computations targeted in Figure 10. We consider the following work elimination policies: **A:** zero activations, **W:** zero weights, **Ap:** Activation static precision, **Wp:** Weight per layer Precision, **Ad:** dynamic precision for activations, **Wd:** per value weight precisions, **Ab:** bit sparsity for activations, and **Wb:** bit sparsity for weights. We do not report the potential for combinations of these properties due to space limitations. However, the presented measurements suffice to demonstrate that the potential for improvements is strong for all models considered.

We find that all policies demonstrate significant potential for all networks studied, an encouraging result suggesting that hardware designers have several options on which properties to exploit to best deliver the desired performance, area, and energy efficiency characteristics for their target application. While a practical accelerator design is unlikely to achieve the full potential of any of these approaches, understanding their relative potential is useful in informing future acceleration efforts. Overall, as expected, we find that bit sparsity has the greatest potential for work reduction.

V. Limitations

This was a broad survey of neural network properties intended to identify general trends from some of the popular neural networks. Many other network architectures exist, and are yet to be developed. However, this work presents a valuable analysis of important value properties as they stand today. Even

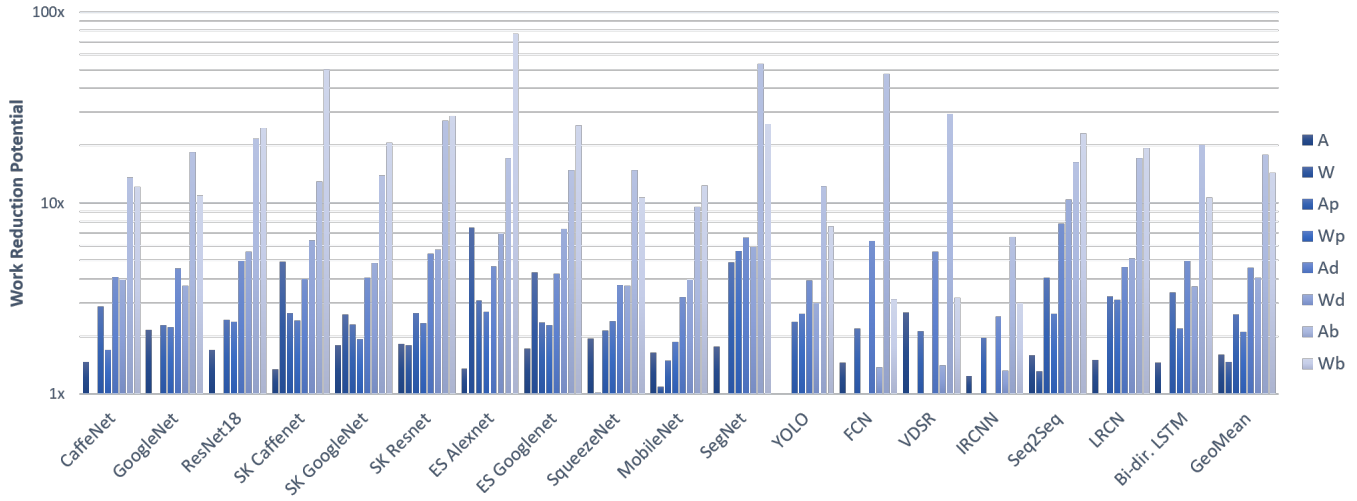

Fig. 10: Per Property Work Reduction Potential. Log scale.

though a fixed point format is commonly used in hardware accelerators, floating point is still commonly used, and there might be other formats that are more suitable in general, or for specific architectures. The potential performance and memory savings assume ideal execution and will most likely not be obtainable in practice. Some of the properties can be amplified by techniques not explored here. For instance, retraining networks with reduced precisions can usually regain the lost accuracy and further reduce the required precisions. Retraining is thus, most likely to amplify the evaluated opportunities. Another technique that may influence the properties studied is quantization. We showed that studied properties persist in a subset of 8b and low precision quantized networks. However further investigation is needed to confirm this for different applications, especially LSTMs. Extreme quantization such as using binary or ternary activations and/or weights often requires re-architecting the network to match baseline accuracy. For this reason quantized models are not as readily available. Despite the aforementioned limitations this is an important study that can inform future acceleration design efforts as it demonstrated that the studied properties persisted over time and apply on a broader set of neural network models.

VI. Conclusion

We have investigated whether certain value related properties that have previously have been demonstrated for image classification CNNs persist in a broader class of networks. We have shown that indeed they persist for newer image classification networks, quantized networks and for several other types of networks such as LSTMs, and for a broader set of applications. The results of this study can inform future acceleration studies as they demonstrate that the studied properties remain valuable for a broader set of networks.

We have shown that the required precisions vary considerably. This result suggests that accelerators that support fine-grain selection of precision may be desirable. Moreover, we have shown that the precisions needed are on the average much smaller than those needed when considering whole layers. Accordingly, this motivates designs which can take advantage of precision variability at a much finer granularity than the layer and those that can detect precisions on the fly. However, for one LSTM layer using 16b of precision is insufficient. The straightforward solution would be to adjust the baseline precision to the necessary 24b. However, it may be possible to achieve a better area, performance and energy-efficiency trade-off if future designs target composable units or arithmetic where the rarely needed high precision computations are decomposed into several lower precision ones.

We have observed that value sparsity does not naturally appear in all applications and models. Reducing precision first helps but in some cases activation and/or weight sparsity remain marginal. Pruning may be able to boost weight sparsity but this does not guarantee a boost in activation sparsity.

Bit sparsity appears to be an inherent property of all the models studied here. Note that bit sparsity does not imply a small value range. This suggests that exploring accelerators that exploit bit sparsity is worthwhile for most applications.

Overall, the results of this study are positive in that they show that precision and sparsity properties of neural networks seem to persist and remain available for exploitation by future accelerator designs.

Acknowledgments

This work was supported in part by the NSERC COHESA Research Network, an NSERC/DND Discovery Supplement, an NSERC Discovery Grant, an NSERC Strategic Partnership Grant, and by the Samsung Advanced Institute of Technology.

References

[1] P. Judd, J. Albericio, T. Hetherington, T. Aamodt, and A. Moshovos, "Stripes: Bit-serial Deep Neural Network Computing ," in *Proceedings of the 49th Annual IEEE/ACM International Symposium on Microarchitecture*, ser. MICRO-49, 2016.

[2] H. Sharma, J. Park, N. Suda, L. Lai, B. Chau, J. K. Kim, V. Chandra, and H. Esmaeilzadeh, "Bit fusion: Bit-level dynamically composable architecture for accelerating deep neural networks," *CoRR*, vol. abs/1712.01507, 2017. [Online]. Available: http://arxiv.org/abs/1712.01507

[3] Chen, Yu-Hsin and Krishna, Tushar and Emer, Joel and Sze, Vivienne, "Eyeriss: An Energy-Efficient Reconfigurable Accelerator for Deep Convolutional Neural Networks," in *IEEE International Solid-State Circuits Conference, ISSCC 2016, Digest of Technical Papers*, 2016, pp. 262–263.

[4] S. Zhang, Z. Du, L. Zhang, H. Lan, S. Liu, L. Li, Q. Guo, T. Chen, and Y. Chen, "Cambricon-X: An accelerator for sparse neural networks," in *49th Annual IEEE/ACM International Symposium on Microarchitecture, MICRO 2016, Taipei, Taiwan, October 15-19, 2016*, 2016, pp. 1–12. [Online]. Available: https://doi.org/10.1109/MICRO.2016.7783723

[5] A. Parashar, M. Rhu, A. Mukkara, A. Puglielli, R. Venkatesan, B. Khailany, J. Emer, S. W. Keckler, and W. J. Dally, "SCNN: An accelerator for compressed-sparse convolutional neural networks," in *Proceedings of the 44th Annual International Symposium on Computer Architecture*, ser. ISCA '17. New York, NY, USA: ACM, 2017, pp. 27–40. [Online]. Available: http://doi.acm.org/10.1145/3079856.3080254

[6] J. Albericio, P. Judd, T. Hetherington, T. Aamodt, N. Enright Jerger, and A. Moshovos, "Cnvlutin: Ineffectual-neuron-free deep neural network computing," in *2016 IEEE/ACM International Conference on Computer Architecture (ISCA)*, 2016.

[7] A. D. Lascorz, S. Sharify, P. Judd, K. Siu, M. Nikolic, and A. Moshovos, "Dpred: Making typical activation values matter in deep learning computing," *CoRR*, vol. abs/1804.06732, 2017. [Online]. Available: http://arxiv.org/abs/1804.06732

[8] J. Albericio, A. Delmás, P. Judd, S. Sharify, G. O'Leary, R. Genov, and A. Moshovos, "Bit-pragmatic deep neural network computing," in *Proceedings of the 50th Annual IEEE/ACM International Symposium on Microarchitecture*, ser. MICRO-50 '17, 2017, pp. 382–394.

[9] A. Delmas, P. Judd, D. M. Stuart, Z. Poulos, M. Mahmoud, S. Sharify, M. Nikolic, and A. Moshovos, "Bit-tactical: Exploiting ineffectual computations in convolutional neural networks: Which, why, and how," *CoRR*, vol. abs/1803.03688, 2018. [Online]. Available: http://arxiv.org/abs/1803.03688

[10] H. Esmaeilzadeh, E. Blem, R. St. Amant, K. Sankaralingam, and D. Burger, "Dark silicon and the end of multicore scaling," in *Proceedings of the 38th Annual International Symposium on Computer Architecture*, ser. ISCA '11. New York, NY, USA: ACM, 2011, pp. 365–376.

[11] Y. Chen, T. Luo, S. Liu, S. Zhang, L. He, J. Wang, L. Li, T. Chen, Z. Xu, N. Sun, and O. Temam, "DaDianNao: A machine-learning supercomputer," in *Microarchitecture (MICRO), 2014 47th Annual IEEE/ACM International Symposium on*, Dec 2014, pp. 609–622.

[12] Y.-H. Chen, J. Emer, and V. Sze, "Eyeriss: A spatial architecture for energy-efficient dataflow for convolutional neural networks," in *Proceedings of the 43rd International Symposium on Computer Architecture*, ser. ISCA '16, 2016, pp. 367–379.

[13] S. Liu, Z. Du, J. Tao, D. Han, T. Luo, Y. Xie, Y. Chen, and T. Chen, "Cambricon: An instruction set architecture for neural networks," in *2016 IEEE/ACM International Conference on Computer Architecture (ISCA)*, 2016.

[14] S. Han, X. Liu, H. Mao, J. Pu, A. Pedram, M. A. Horowitz, and W. J. Dally, "EIE: Efficient inference engine on compressed deep neural network," in *Proceedings of the 43rd International Symposium on Computer Architecture*, ser. ISCA '16. Piscataway, NJ, USA: IEEE Press, 2016, pp. 243–254. [Online]. Available: https://doi.org/10.1109/ISCA.2016.30

[15] B. Reagen, P. Whatmough, R. Adolf, S. Rama, H. Lee, S. K. Lee, J. M. Hernández-Lobato, G.-Y. Wei, and D. Brooks, "Minerva: Enabling low-power, highly-accurate deep neural network accelerators," in *Proceedings of the 43rd International Symposium on Computer Architecture*. IEEE Press, 2016, pp. 267–278.

[16] A. Delmas, P. Judd, S. Sharify, and A. Moshovos, "Dynamic stripes: Exploiting the dynamic precision requirements of activation values in neural networks," *CoRR*, vol. abs/1706.00504, 2017. [Online]. Available: http://arxiv.org/abs/1706.00504

[17] N. P. Jouppi, C. Young, N. Patil, D. Patterson, G. Agrawal, R. Bajwa, S. Bates, S. Bhatia, N. Boden, A. Borchers, R. Boyle, P.-l. Cantin, C. Chao, C. Clark, J. Coriell, M. Daley, M. Dau, J. Dean, B. Gelb, T. V. Ghaemmaghami, R. Gottipati, W. Gulland, R. Hagmann, C. R. Ho, D. Hogberg, J. Hu, R. Hundt, D. Hurt, J. Ibarz, A. Jaffey, A. Jaworski, A. Kaplan, H. Khaitan, D. Killebrew, A. Koch, N. Kumar, S. Lacy, J. Laudon, J. Law, D. Le, C. Leary, Z. Liu, K. Lucke, A. Lundin, G. MacKean, A. Maggiore, M. Mahony, K. Miller, R. Nagarajan, R. Narayanaswami, R. Ni, K. Nix, T. Norrie, M. Omernick, N. Penukonda, A. Phelps, J. Ross, M. Ross, A. Salek, E. Samadiani, C. Severn, G. Sizikov, M. Snelham, J. Souter, D. Steinberg, A. Swing, M. Tan, G. Thorson, B. Tian, H. Toma, E. Tuttle, V. Vasudevan, R. Walter, W. Wang, E. Wilcox, and D. H. Yoon, "In-datacenter performance analysis of a tensor processing unit," in *Proceedings of the 44th Annual International Symposium on Computer Architecture*, ser. ISCA '17, 2017, pp. 1–12.

[18] S. Sharify, A. D. Lascorz, P. Judd, and A. Moshovos, "Loom: Exploiting weight and activation precisions to accelerate convolutional neural networks," *CoRR*, vol. abs/1706.07853, 2017. [Online]. Available: http://arxiv.org/abs/1706.07853

[19] S. Sharify, M. Mahmoud, A. D. Lascorz, M. Nikolic, and A. Moshovos, "Laconic deep learning computing," *CoRR*, vol. abs/1805.04513, 2018. [Online]. Available: http://arxiv.org/abs/1805.04513

[20] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei, "ImageNet Large Scale Visual Recognition Challenge," *arXiv:1409.0575 [cs]*, Sep. 2014, arXiv: 1409.0575.

[21] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell, "Caffe: Convolutional architecture for fast feature embedding," *arXiv preprint arXiv:1408.5093*, 2014.

[22] J. Park, S. Li, W. Wen, P. T. P. Tang, H. Li, Y. Chen, and P. Dubey, "Faster CNNs with Direct Sparse Convolutions and Guided Pruning," in *5th International Conference on Learning Representations (ICLR)*, 2017.

[23] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," *CoRR*, vol. abs/1512.03385, 2015.

[24] Yang, Tien-Ju and Chen, Yu-Hsin and Sze, Vivienne, "Designing Energy-Efficient Convolutional Neural Networks using Energy-Aware Pruning," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[25] F. N. Iandola, M. W. Moskewicz, K. Ashraf, S. Han, W. J. Dally, and K. Keutzer, "SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and <1mb model size," *CoRR*, vol. abs/1602.07360, 2016. [Online]. Available: http://arxiv.org/abs/1602.07360

[26] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "MobileNets: Efficient convolutional neural networks for mobile vision applications," *CoRR*, vol. abs/1704.04861, 2017. [Online]. Available: http://arxiv.org/abs/1704.04861

[27] Y. Zhao, X. Gao, R. Mullins, and C. Xu, "Mayo: A framework for auto-generating hardware friendly deep neural networks," 2018.

[28] Z. Cai, X. He, J. Sun, and N. Vasconcelos, "Deep learning with low precision by half-wave gaussian quantization," *CoRR*, vol. abs/1702.00953, 2017. [Online]. Available: http://arxiv.org/abs/1702.00953

[29] V. Badrinarayanan, A. Kendall, and R. Cipolla, "SegNet: A deep convolutional encoder-decoder architecture for image segmentation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2017.

[30] G. J. Brostow, J. Fauqueur, and R. Cipolla, "Semantic object classes in video: A high-definition ground truth database," *Pattern Recognition Letters*, vol. xx, no. x, pp. xx–xx, 2008.

[31] J. Redmon and A. Farhadi, "YOLO9000: better, faster, stronger," *CoRR*, vol. abs/1612.08242, 2016. [Online]. Available: http://arxiv.org/abs/1612.08242

[32] M. Everingham, S. M. A. Eslami, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman, "The pascal visual object classes challenge: A retrospective," *International Journal of Computer Vision*, vol. 111, no. 1, pp. 98–136, Jan. 2015.

[33] E. Shelhamer, J. Long, and T. Darrell, "Fully convolutional networks for semantic segmentation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 4, pp. 640–651, April 2017.

[34] D. Li and Z. Wang, "Video superresolution via motion compensation and deep residual learning," *IEEE Transactions on Computational Imaging*, vol. 3, no. 4, pp. 749–762, Dec 2017.

[35] D. Martin, C. Fowlkes, D. Tal, and J. Malik, "A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics," in *Proceedings Eighth IEEE International Conference on Computer Vision. ICCV 2001*, vol. 2, 2001, pp. 416–423 vol.2.

[36] C. Dong, Y. Deng, C. Change Loy, and X. Tang, "Compression artifacts reduction by a deep convolutional network," in *Proceedings of the IEEE International Conference on Computer Vision*, 2015, pp. 576–584.

[37] M. Bevilacqua, A. Roumy, C. Guillemot, and M. L. Alberi-Morel, "Low-complexity single-image super-resolution based on nonnegative neighbor embedding," 2012.

[38] R. Zeyde, M. Elad, and M. Protter, "On single image scale-up using sparse-representations," in *Curves and Surfaces*, J.-D. Boissonnat, P. Chenin, A. Cohen, C. Gout, T. Lyche, M.-L. Mazure, and L. Schumaker, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 711–730.

[39] K. Zhang, W. Zuo, S. Gu, and L. Zhang, "Learning Deep CNN Denoiser Prior for Image Restoration," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 3929–3938.

[40] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," in *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2*, ser. NIPS'14. Cambridge, MA, USA: MIT Press, 2014, pp. 3104–3112. [Online]. Available: http://dl.acm.org/citation.cfm?id=2969033.2969173

[41] J. Donahue, L. A. Hendricks, S. Guadarrama, M. Rohrbach, S. Venugopalan, K. Saenko, and T. Darrell, "Long-term recurrent convolutional networks for visual recognition and description," in *CVPR*, 2015.

[42] T. Lin, M. Maire, S. J. Belongie, L. D. Bourdev, R. B. Girshick, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft COCO: common objects in context," *CoRR*, vol. abs/1405.0312, 2014. [Online]. Available: http://arxiv.org/abs/1405.0312

[43] C. Wang, H. Yang, C. Bartz, and C. Meinel, "Image captioning with deep bidirectional lstms," in *Proceedings of the 2016 ACM on Multimedia Conference*. ACM, 2016, pp. 988–997.

[44] C. Rashtchian, P. Young, M. Hodosh, and J. Hockenmaier, "Collecting image annotations using amazon's mechanical turk," in *Proceedings of the NAACL HLT 2010 Workshop on Creating Speech and Language Data with Amazon's Mechanical Turk*, ser. CSLDAMT '10. Stroudsburg, PA, USA: Association for Computational Linguistics, 2010, pp. 139–147. [Online]. Available: http://dl.acm.org/citation.cfm?id=1866696.1866717

[45] P. Judd, J. Albericio, T. Hetherington, T. M. Aamodt, N. E. Jerger, and A. Moshovos, "Proteus: Exploiting numerical precision variability in deep neural networks," in *Proceedings of the 2016 International Conference on Supercomputing*, ser. ICS '16. New York, NY, USA: ACM, 2016, pp. 23:1–23:12. [Online]. Available: http://doi.acm.org/10.1145/2925426.2926294

[46] J. Choi, Z. Wang, S. Venkataramani, P. I. Chuang, V. Srinivasan, and K. Gopalakrishnan, "PACT: parameterized clipping activation for quantized neural networks," *CoRR*, vol. abs/1805.06085, 2018. [Online]. Available: http://arxiv.org/abs/1805.06085

[47] M. Zhu and S. Gupta, "To prune, or not to prune: exploring the efficacy of pruning for model compression," *ArXiv e-prints*, Oct. 2017.

[48] Y. L. Cun, J. S. Denker, and S. A. Solla, "Advances in neural information processing systems 2," D. S. Touretzky, Ed. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1990, ch. Optimal Brain Damage, pp. 598–605. [Online]. Available: http://dl.acm.org/citation.cfm?id=109230.109298

[49] D. Clevert, T. Unterthiner, and S. Hochreiter, "Fast and accurate deep network learning by exponential linear units (elus)," *CoRR*, vol. abs/1511.07289, 2015. [Online]. Available: http://arxiv.org/abs/1511.07289