SuperSlash: A Unified Design Space Exploration and Model Compression Methodology for Design of Deep Learning Accelerators with Reduced Off-Chip Memory Access Volume

Hazoor Ahmad, Tabasher Arif, Muhammad Abdullah Hanif, Rehan Hafiz, Muhammad Shafique

***Abstract*—Deploying Deep Learning (DL) models on resource-constrained embedded devices is a challenging task. The limited on-chip memory on such devices results in increased off-chip memory access volume, thus limiting the size of DL models that can be efficiently realized in such systems. Design Space Exploration (DSE) under memory constraint, or to achieve minimal off-chip memory access volume, has recently received much attention. Unfortunately, DSE alone cannot reduce the amount of off-chip memory accesses beyond a certain point due to the fixed model size. Model compression via pruning can be employed to reduce the size of the model and the associated off-chip memory accesses. However, in this paper, we demonstrate that pruned models with even the same accuracy and model size may require a different number of off-chip memory accesses depending upon the pruning strategy adopted. Thus mainstream pruning techniques may not be closely tied to the design goals, and thereby hard to be integrated with existing DSE techniques. To overcome this problem, we propose SuperSlash, a unified solution for DSE and Model Compression. SuperSlash estimates off-chip memory access volume overhead of each layer of a deep learning model by exploring multiple design candidates. In particular, it evaluates multiple data reuse strategies for each layer, along with the possibility of layer fusion. Layer fusion aims at reducing the off-chip memory access volume by avoiding the intermediate off-chip storage of a layer's output and directly using it for processing of the subsequent layer. SuperSlash then guides the pruning process via a ranking function, which ranks each layer according to its explored off-chip memory access cost. We demonstrate that SuperSlash not only offers an extensive design space coverage but also provides lower off-chip memory access volume (up to 57.71%, 25.83%, 47.73%, 29.02% reduction for VGG16, ResNet56, ResNet110, and MobileNetV1 respectively) as compared to the state-of-art.**

***Index Terms*—Design space exploration, DSE, off-chip memory access volume, deep neural network, accelerators, model compression, pruning, optimization.**

I. Introduction

OVER the past decade, Deep Neural Networks (DNNs) have achieved remarkable success by providing state-of-art accuracy in several challenging computer vision tasks such as image classification [1], object recognition [2], and autonomous driving [3]. A convolutional layer within a deep Convolutional Neural Network (CNN) requires the convolution of an Input Feature Map (IFM) with learned filter parameters/weights (WHTs) to compute the corresponding Output Feature Map (OFM). Since DNNs comprise millions of parameters, they require a large amount of on-chip storage. Thus, their deployment on embedded/edge devices with limited on-chip memory [4] [5], is a challenging task [6]. This is because, for each of IFM, OFM, and WHTs, the data needs to be shuttled between off-chip and on-chip memory in the form of small *tiles* of data chunks. Unfortunately, the energy cost of this off-chip memory (typically a DRAM) accesses can be orders of magnitude higher than that of the arithmetic operations (up to 80% for [7] and up to 90% for DianNao [8]) further limiting the size of models that can be realized efficiently on such resource-constrained devices. Provided a DNN model and hardware constraints, the off-chip memory access volume depends upon the *tile size* and the *data-reuse strategy*, i.e., the particular order in which the IFM, OFM, and WHTs are accessed [7]. This led to a growing interest in the Design Space Exploration (DSE) techniques [7], [9]–[11] to not only select a suitable *tile size* for each of the IFM, OFM and WHTs, but also to select a corresponding *data-reuse strategy* aimed at reducing the off-chip memory accesses [12]. In case the off-chip memory access volume is still higher than the design goal, DSE cannot reduce it further below a theoretical limit that is set due to the fixed size of the DNN model being deployed. *Parameter pruning* can than be employed to reduce the model size and consequently reduce the off-chip memory access volume. Pruning has successfully been explored in the past to perform over 70% model compression for just 1% loss in accuracy for a few networks [13]–[18]. However, there is a caveat. Most of these compression techniques prune the filter parameters without taking into account the DSE design goals and hence may not provide the desired results. In the following, we provide a supporting motivational analysis with the key insights.

This article was presented in the International Conference on Hardware/Software Codesign and System Synthesis 2020 and appears as part of the ESWEEK-TCAD special issue.

H. Ahmad, T. Arif and R. Hafiz are with Information Technology University, Lahore 54700, Pakistan. (e-mail: phdee17004@itu.edu.pk, msee16012@itu.edu.pk and rehan.hafiz@itu.edu.pk)

M. A. Hanif is with Institute of Computer Engineering, Vienna University of Technology (TU Wien), 1040 Vienna, Austria. (e-mail: muhammad.hanif@tuwein.ac.at)

M. Shafique is with Division of Engineering, New York University Abu Dhabi (NYU AD), Abu Dhabi, United Arab Emirates, and Institute of Computer Engineering, Technische Universität Wien (TU Wien), Vienna, Austria. (e-mail: muhammad.shafique@nyu.edu and muhammad.shafique@tuwein.ac.at)

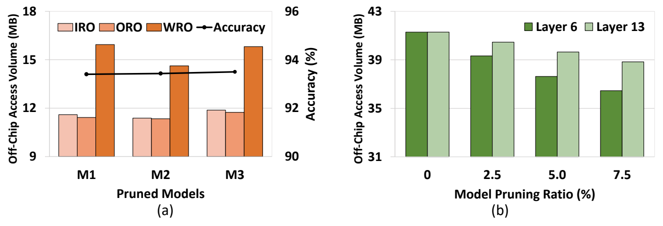

Fig. 1. Motivational analysis highlighting that off-chip memory access volume varies even when (a) the same number of parameters are pruned from VGG16 network via 3 different pruning schemes (M1, M2, M3), and when (b) the same number of parameters are pruned from different layers (Layer 6 and Layer 13) of the VGG16 network.

Motivational Analysis and Key Challenges: We consider three pruned models (M1, M2, and M3) for a VGG16 network [19] trained on the CIFAR-10 dataset [20] with a baseline accuracy of 93.4%. The three models were obtained by applying three different pruning techniques with each one of them aimed at achieving 64% pruning ratio for a minimal loss in accuracy. The decision to keep 64% pruning ratio for this motivational analysis was made to obtain reproducible results since this was the cumulative pruning ratio achieved after applying the layer-wise pruning ratio as reported by the state-of-the-art in [15]. The corresponding model is represented by M1 in Figure 1(a). Model M2 was obtained using the technique being proposed in this paper see (details in Section 3), while Model M3 was generated by equally distributing the pruning budget across

Fig. 2. The best design points from DSE of one pruned model (Red Cross in (a) for M1) may not relate with the best design points from another model (Black TriangleN in (b) for M2) and vice versa. The best design points of M2 and M1 from their respective DSE are shown in (a) and (b) using the black triangleN and red cross , respectively. Access volume for M2 and M1 on the best points of M1 and M2 are shown in (a) and (b) using black cross and red triangleN, respectively.

all the layers. In Figure 1(a), we plot the off-chip memory access volume obtained by performing DSE over all possible tile sizes for all the three data reuse strategies, i.e., Input Reuse Oriented (IRO), Weight Reuse Oriented (WRO), and Output Reuse Oriented (ORO), as deﬁned in [12] (and described later in Section-3), considering a 32KB on-chip buffer. There are two key insights.

1) Even for the same model, different design reuse strategies result in a different number of off-chip memory accesses.
2) Even for the same data reuse strategy, the three pruned models require a different number of off-chip memory accesses.

To investigate the reason, we obtained six pruned models variants of VGG16 by removing the least signiﬁcant 375K, 750K and 1,125K parameters from either Layer 6 or Layer 13. Removal of 375K, 750K, and 1,125K number of parameters from a total of 15M parameters of VGG16 corresponds to a cumulative pruning ratio of 2.5%, 5%, and 7.5%, respectively. In Figure 1(b) we plot the minimum off-chip memory access volume for each of these models considering DSE across all the three data reuse strategies (i.e., IRO, WRO and ORO). It can be observed that, as compared to the Layer 13, pruning the same amount of parameters from Layer 6 provides greater beneﬁt in terms of reduced off-chip memory access volume. Thus, even with the same amount of parameters removed from different layers can result in a different effect on the off-chip memory access volume. Consequently, provided a pruning ratio goal, distribution of the pruning budget across the network layers, while aiming for low off-chip memory access volume, is a challenging task

Since different pruning techniques may remove a different number of parameters from the network layers even for the same cumulative pruning ratio, we evaluate if DSE carried out for one pruned model (M1) is indeed applicable to another (M2). Figure 2(a) and (b) plot valid design points for DSE carried out on M1 (Red Crosses) and M2 (Black Triangles) models, respectively. These points were explored by considering the WRO data reuse strategy via state-of-the-art SmartShuttle DSE technique [12] for multiple on-chip buffer sizes. In Figure 2(a) black cross represents the off-chip memory access volume for model M2 when conﬁguration (tile sizes) of the best design point (lowest number of off-chip memory accesses) of M1 DSE is utilized. We then compared the off-chip memory access volume of this point to that of the best design point from M2 DSE, represented by the black triangle in Figure 2(a). Similarly, in Figure 2(b) red triangle and red cross point to the off-chip memory access volume for model M1 considering the best design point considering M2 DSE and M1 DSE, respectively. It can be observed from Figure 2 that the best design points for M1 are not necessarily the best design points for M2 and vice versa. Thus, DSE may provide different design points for two models of same size but pruned via different techniques.

Required: Our motivational analysis demonstrates that a key challenge is to devise a uniﬁed methodology which, provided an accuracy constraint or the desired pruning ratio, performs co-exploration of a DNN model and its associated design space while minimizing the off-chip memory access volume, considering a predeﬁned on-chip buffer size.

Limitations of the state-of-the-art: There has been a considerable effort towards structured [15], [16], [21], [18], [22] and unstructured pruning schemes [13], [14] [23] for model compression. Structured pruning is preferred as it does not require extra overhead to handle sparse matrices and un-structured sparsity [13]. Unfortunately, there is no structured pruning scheme that prunes the network while considering the DSE's design constraints. Similarly, while there has been reasonable prior work in relation to DSE [7], [9]–[12], [24] for the design of deep learning accelerators, none of the state-of-the-art considers model pruning as part of DSE in order to reduce its off-chip memory access volume.

Novel Contribution in a Nutshell: In this paper, we present SuperSlash, a uniﬁed co-exploration methodology that:

1) Employs a layer-wise ranking function to rank the pruning suitability of each layer by considering its off-chip memory access volume overhead that is estimated via an in-tandem DSE. In particular, the DSE evaluates multiple data reuse strategies for each layer, the associated tile sizes, and the possibility to apply layer fusion across any two consecutive layers.
2) Provided an accuracy constraint, it distributes the available pruning budget across multiple layers of a DNN model, by incorporating the proposed ranking function, for predeﬁned on-chip buffer size.

Open Source Contributions: To facilitate further research and reproducibility of the presented results, we will provide a public release of our pruned networks and framework at [25].

II. Literature Survey:

Design Space Exploration: Most of the DSE techniques for Deep Learning Accelerators are aimed at reducing either the latency, throughput, or off-chip memory access volume by considering different data reuse strategies. A dominant class of these techniques relate to speciﬁc hardware architectures and rely on cycle-accurate models to predict latency and throughput for DSE. Zhang et al. [9] proposed a rooﬂine based analytical model which performs DSE to search for a design point with the best performance and lowest resource requirement, speciﬁcally for FPGAs. They considered ORO based design reuse to maximize the computation-to-communication ratio. Rahman et al. [10] improved upon Zhang et al. by performing an IRO based DSE and utilized an input reuse network, a systolic array inspired 2D grid of registers, to exploit local interconnections. Chen et al. [7] highlighted the merits of various data reuse strategies and proposed a complete accelerator architecture utilizing an energy-efﬁcient row-stationary data-ﬂow operating ORO on Eyeriss chip [24] which, uses a 12 14 systolic array. Lu et al. [11] presented FlexFlow, a customized systolic array-based hardware exploiting parallel processing, and data reuse opportunity for outputs and weights. Recently, Li et al. [12] proposed a layer-wise adaptive data reuse strategy selection methodology that performed layer-wise scheduling decision to use either of the ORO or WRO based on network architecture and hardware constraints. The scheme was orthogonal in the sense that it can be easily integrated to other accelerator designs.

Layer Fusion: Layer fusion aims at reducing the off-chip memory access volume by avoiding the intermediate off-chip storage of a layer's output, and directly using it for the processing of the subsequent layer. Applying layer fusion between two layers thus avoids the shuttling of the data to and fro from the off-chip memory. Alwani et al. [26] proposed the seminal work in this direction. However, rather than performing a generic DSE to evaluate the feasibility of layer fusion under the constraint of xed on-chip storage (which is a typical case of an embedded system), they aimed at evaluating the additional on-chip memory required to fuse a certain number of layers. Yu et al. [27] proposed a cross-layer scheduling scheme speci cally for Winograd based computing and did not support convolution with strides. Furthermore, they did not focus on providing a generic DSE methodology to explore the sizes of the tiles. Chung et al. [28] proposed a CNN architecture where layer fusion was employed for reducing the off-chip memory access volume. However, the scheme was speci c to a particular CNN being employed for Full-HD Super Resolution. Hao-Ning [29] demonstrated layer fusion speci cally for Depth-wise Separable Convolutions. Sim et al. [30] proposed a dedicated DNN accelerator incorporating layer fusion by keeping the intermediate activation data on-chip rather than sending it off-chip. However, they do not perform any DSE for reducing the off-chip memory access volume. Thus, while the techniques mentioned above explore multi-layer fusion, they do not perform a generic DSE exploring multiple reuse patterns for minimizing off-chip memory access volume [26], [27], and neither consider the opportunity for model compression.

Model Compression via Pruning: Pruning can be categorized into the ne-grained and coarse-grained pruning. In ne-grained pruning schemes [13] [23] individual weights from the network are removed, and hence they result in sparse connections and require dedicated sparse libraries [31] and hardware [32] to achieve actual speed up. Wen et al. [14] introduced structured sparsity in the network using group LASSO regularization. However, it still required dedicated libraries to gain bene ts. Instead of pruning individual lter weights, in coarse-grained (or the so-called structured) pruning schemes [33] [15] [16] [22] whole 3D lter/s with their corresponding channels are removed. This preserves the regularity in the structure of a CNN, and hence no dedicated libraries/accelerators [32] are required. Li et al. in [15] is the most prominent work in this area. It ranks each lter within a layer according to its L1-norm and then prunes the lowest ranking lter from the layer-based upon layer-wise sensitivity analysis. In a recent approach, Yu et al. [16] removed lters from all layers jointly based upon a prede ned pruning ratio by minimizing the reconstruction error in the nal response layer (FRL). Salama et al. [17] demonstrated a structured class-blind pruning strategy by simultaneously pruning lters according to a global threshold. Recently, Li et al. [18] proposed HeadStart, in which they used reinforcement learning to explore different groups of lters for pruning and demonstrated convincing results as compared to [15]. None of the aforementioned structured pruning schemes incorporate the knowledge of the accelerator design goal. To overcome these limitations of state-of-the-art, we propose a uni ed DSE and Model Compression Methodology focusing on reduced off-chip memory accesses for deep learning accelerators.

III. Proposed SuperSlash Methodology

System Model and Problem De nition: Our proposed SuperSlash methodology supports the generic architectural topology (Figure 3) of various on-chip deep learning accelerators. Such accelerators typically comprise of a global on-chip buffer (GBUF) to store portions of IFM, OFM, and WHTs being processed as well as the intermediate results. Computations are carried out in a Neural Processing Array, which is simply a 2D array of multiple Processing Elements (PEs) with some pre-de ned connectivity/routing. Regardless of the internal architecture of the PE and its connectivity, the Neural Processing

Fig. 3. A generic DNN accelerator architecture.

Array fetches the data from GBUF, applies the convolutional operations, pooling/activation functions, and stores the result back in the GBUF. Since size of GBUF (B) is small, there is a limit to the portion of IFM, OFM, and WHTs that can be stored in GBUF. Thus, small portions i.e., (*tiles*) of each of these data sections are brought from the off-chip memory for processing. Since Deep CNNs require computations on a large spatial neighborhood, the smaller the GBUF, the smaller are the tile sizes, and hence the larger is the volume of data that needs to be shuttled between off-chip and on-chip memory. A Controller is typically responsible for fetching the IFM, WHTs, and OFM tiles from off-chip DRAM in a scheduled manner, as dictated by the chosen data reuse strategy. There are three predominant data reuse strategies [12], namely input reuse oriented (IRO), output reuse oriented (ORO), and weight reuse oriented (WRO), which exploit the reuse of IFM, OFM, and WHTs tiles, respectively. IRO ensures that an IFM tile is completely processed for all the lters before being removed from the GBUF. Similarly, ORO schedules the on-chip accumulation of all the partial sums that relate to an OFM tile before it is nally moved off-chip. WRO ensures that a tile of WHTs is completely applied to all the IFM before it is discarded. Layer fusion is another design reuse strategy that may provide bene t in terms of reduced off-chip memory access volume depending upon the GBUF size. DSE is thus required to be carried out to select a layer-wise data reuse strategy and the corresponding tile sizes for IFM, OFM, and WHTs

SuperSlash Overview: Our SuperSlash methodology performs this DSE along with a model compression scheme that is aimed at reducing the off-chip memory access volume. In particular, provided a DNN model, the available on-chip GBUF size, B, and a minimum accuracy constraint, A_{th}, our methodology minimizes the off-chip memory access volume via pruning the model. The SuperSlash methodology comprises four steps and is illustrated in Figure 4. We provide an overview of our methodology here, followed by a detailed description of each of these steps.

Each iteration of SuperSlash explores various hardware and software design variables, such as tile sizes, data reuse strategies, and per-layer parameter pruning candidates. The methodology takes as an input the original model W^o along with the provided constraints. The off-chip memory access volume of each layer is computed in Step 1 In Step 2a, a rank is assigned to each layer based on its off-chip memory access volume. Step 2b employs the layer-wise ranking to prune layers in a progressive manner. As long as the model does not violate the accuracy threshold, it is sent back to Step-1 for further model compression in an iterative way. If at any point, the accuracy of the pruned model falls below the accuracy threshold A_{th}, a ne-tuning (Step 3) is performed to regain the lost accuracy. Thus, Steps 1-3 de nes one SuperSlash iteration, where Step-3 runs only if the accuracy constraint is violated. In case the network can not be compressed further within the accuracy constraints, the last known acceptable model is passed over to Step 4 for ne-tuning and nal DSE. SuperSlash also allows using the pruning ratio C and off-chip memory access level V as the terminating condition. In this case, a relaxed accuracy threshold can be provided, and the iterations are terminated when the desired pruning ratio or off-chip memory access level is reached. In the following, we discuss each of these steps in detail.

A. DSE Model and Off-chip Memory Access Computation

Let us consider a DNN model W^i, with L number of layers, that is being provided as an input to the i^{th} iteration of Super-Slash. The model is completely de ned by $M^i = f m_1^i; :::; m_L^i g$, $N^i = f n_1^i; :::; n_L^i g$, $R^i = f r_1^i; :::; r_L^i g$, $C^i = f c_1^i; :::; c_L^i g$, $K^i = f k_1; :::; k_L g$, $S^i = f s_1; :::; s_L g$, and $P^i = f p_1; :::; p_L gi$. Here m_l^i, n_l^i, r_l^i, c_l^i, k_l, s_l and p_l denote the number of lters (or OFM channels), IFM channels, rows of OFM, columns of OFM, kernel size, stride and pooling kernel size of the l^{th} layer of the W^i model, respectively. For a batch of inputs, the layer-wise data volume $(_l^i)$ required to accommodate all the IFM, OFM and WHTs can be computed using,

$$_{I}{}_l^i = dn_l^i r_l^i c_l^i s_l^2 \tag{1a}$$
$$_{O}{}_l^i = dm_l^i r_l^i c_l^i = p_l^2 \tag{1b}$$
$$_{W}{}_l^i = m_l^i n_l^i k_l^2 \tag{1c}$$

Since GBUF size is limited, only tiles of these volumes can be brought in the on-chip memory. Since s_l^2, p_l and k_l are speci ed by the model network, the DSE parameters for tile size con guration are $t_{m}{}_l^i; t_{n}{}_l^i; t_{r}{}_l^i$ and $t_{c}{}_l^i$. Speci cally, the tile size for IFM, OFM and WHTs is given by $(t_{n}{}_l^i t_{r}{}_l^i t_{c}{}_l^i s_l^2)$, $(t_{m}{}_l^i t_{r}{}_l^i t_{c}{}_l^i = p_l^2)$ and $(k_l^2 t_{m}{}_l^i t_{n}{}_l^i)$, respectively. The total memory in GBUF required to store one instance of each of these tiles is given by,

$$B_l^i = t_{n}{}_l^i t_{r}{}_l^i t_{c}{}_l^i s_l^2 + t_{m}{}_l^i t_{r}{}_l^i t_{c}{}_l^i = p_l^2 + k_l^2 t_{m}{}_l^i t_{n}{}_l^i \tag{2}$$

Using the layer-wise data volume and tile size parameters $(t_{m}{}_l^i; t_{n}{}_l^i; t_{r}{}_l^i; t_{c}{}_l^i)$, we can compute the off-chip memory access factor, i.e. the number of times a tile of data is required to be shuttled, for each of the IFM, OFM and WHTs. These access factors depend upon the data reuse strategy being explored and are given by,

$$f_{I}{}_l^i = \begin{cases} 1 & IRO \\ dm_l^i = t_{m}{}_l^i e & Others \end{cases} \tag{3a}$$
$$f_{O}{}_l^i = \begin{cases} 1 & ORO \\ 2dn_l^i = t_{n}{}_l^i e \quad 1 & Others \end{cases} \tag{3b}$$
$$f_{W}{}_l^i = \begin{cases} 1 & WRO \\ ddr_l^i = t_{r}{}_l^i e dc_l^i = t_{c}{}_l^i e & Others \end{cases} \tag{3c}$$

Note, the access factor for IFM, OFM and WHT is 1 for IRO, ORO and WRO, respectively, since for each of these cases the tiles are required to be shuttled only once. For a particular data reuse strategy, the total layer-wise off-chip memory access volume V_l^i is equal to the sum of products of the corresponding data volumes and access factors, and is given by,

$$V_{IRO}{}_l^i = {}_{I}{}_l^i + {}_{O}{}_l^i f_{O}{}_l^i + {}_{W}{}_l^i f_{W}{}_l^i \tag{4a}$$
$$V_{ORO}{}_l^i = {}_{I}{}_l^i f_{I}{}_l^i + {}_{O}{}_l^i + {}_{W}{}_l^i f_{W}{}_l^i \tag{4b}$$
$$V_{WRO}{}_l^i = {}_{I}{}_l^i f_{I}{}_l^i + {}_{O}{}_l^i f_{O}{}_l^i + {}_{W}{}_l^i \tag{4c}$$

The above equations are valid for Single Layer Processing (SLP). SuperSlash DSE also investigates the possibility of Multi Layer Fusion (MLF) by utilizing each OFM tile of l^{th} layer as an IFM tile for $l + 1^{th}$ layer. For MLF, SuperSlash considers ORO on the rst layer and IRO on the second layer. In this way, the off-chip transfer of OFM of l^{th} layer and IFM of $l + 1^{th}$ layer can be saved. Since layer fusion requires combined processing of two layers on the accelerator, it does require some additional memory in the GBUF to store the OFM tile of $l + 1^{th}$ layer with size$(k_{l+1}^2 \; t_{m}{}_{l+1}^i \; t_{m}{}_l^i)$, and WHTs tile of $l + 1^{th}$ layer with size$(t_{m}{}_{l+1}^i \; t_{r}{}_l^i t_{c}{}_l^i = (p_l^2 p_{l+1}^2 \; s_{l+1}^2))$. Thus for any two layers being fused, their layer wise access volume is given by:

$$V_{MLF}{}_l^i = V_{ORO}{}_l^i = {}_{I}{}_l^i f_{I}{}_l^i + {}_{W}{}_l^i f_{W}{}_l^i \tag{5a}$$
$$V_{MLF}{}_{l+1}^i = V_{IRO}{}_{l+1}^i = {}_{O}{}_{l+1}^i f_{O}{}_{l+1}^i + {}_{W}{}_{l+1}^i f_{W}{}_{l+1}^i \tag{5b}$$

Equations 4 and 5 thus provide the per-layer off-chip data access volume for all the possible data reuse strategies D_l^i during the i^{th} iteration. Here D_l^i can be either of the IRO, ORO, WRO, $MLF{}_l^i$ or $MLF{}_{l+1}^i$. We now de ne the SuperSlash optimization problem formally. Provided a network model (W^i), the values of Accuracy threshold A_{th} and GBUF size B constraints, SuperSlash DSE minimizes the off-chip memory access volume by exploring all the possible data reuse strategies for all the possible tile sizes. We thus de ne our objective function as:

$$\begin{array}{ll} \underset{<t_{m}{}_l^i; t_{n}{}_l^i; t_{r}{}_l^i; t_{c}{}_l^i; D_l^i; \; {}_l^i>}{\text{minimize}} & V^i = \min(V(W^i)) \\ \text{s.t.} & A_{W^i} \quad A_{th} \\ & B_l^i \quad B \end{array} \tag{6}$$

Here, A_{W^i} is the accuracy of model W^i and ${}_l^i$ is the pruning ratio of l^{th} layer of i^{th} iteration. $V^i = \sum_{l=1}^{L} v_l^i$ is the combined access volume for all the layers of the model, where the per-layer off-chip memory access volume is given by:

$$v_l^i = \begin{cases} \min(V_{IRO}{}_l^i; V_{ORO}{}_l^i; V_{WRO}{}_l^i) & SLP \\ V_{MLF}{}_l^i \text{ or } V_{MLF}{}_{l+1}^i & MLF \end{cases} \tag{7}$$

For MLF, the layers are considered in pair, i.e. $v_l^i = V_{MLF}{}_l^i$ only if $v_{l+1}^i = V_{MLF}{}_{l+1}^i$. The decision to select MLF for two layers is made in case $(V_{MLF}{}_l^i + V_{MLF}{}_{l+1}^i)$ is less than $(\min(V_{IRO}{}_l^i; V_{ORO}{}_l^i; V_{WRO}{}_l^i) + \min(V_{IRO}{}_{l+1}^i; V_{ORO}{}_{l+1}^i; V_{WRO}{}_{l+1}^i))$. The cumulative design space of SuperSlash is large. We thus perform this co-exploration in two steps First, we perform a hardware DSE on the input model using Eq. 6 for all hardware design parameters. In the next step (Step-2) we perform model exploration by utilizing v_l^i to rank the suitability of each layer for pruning, and explore ${}_l^i$ for minimal off-chip memory access volume.

TABLE I
SELECTION OF DATA REUSE STRATEGY FOR SINGLE LAYER PROCESSING

Condition	D_l^i	Ranges
${}_{W}{}_l^i \quad {}_{I}{}_l^i$ and ${}_{W}{}_l^i \quad {}_{O}{}_l^i$	WRO	I & II
${}_{O}{}_l^i \quad {}_{I}{}_l^i$ and ${}_{O}{}_l^i \quad {}_{W}{}_l^i$	ORO	I, III & IV
${}_{I}{}_l^i \quad {}_{O}{}_l^i$ and ${}_{I}{}_l^i \quad {}_{W}{}_l^i$	IRO	II, III & IV

TABLE II
SUPERSLASH SEARCH RANGES

#	$t_{m}{}_l^i$	$t_{n}{}_l^i$	$t_{r}{}_l^i$	$t_{c}{}_l^i$	For MLF: $t_{m}{}_{l+1}^i$
I	m_l^i	$1; :::; n_l^i$	$k_l; :::; r_l^i$	$k_l; :::; c_l^i$	$1; :::; m_{l+1}^i$
II	$1; :::; m_l^i$	n_l^i	$k_l; :::; r_l^i$	$k_l; :::; c_l^i$	-
III	$1; :::; m_l^i$	$1; :::; n_l^i$	r_l^i	$k_l; :::; c_l^i$	$1; :::; m_{l+1}^i$
IV	$1; :::; m_l^i$	$1; :::; n_l^i$	$k_l; :::; r_l^i$	c_l^i	$1; :::; m_{l+1}^i$

Search Space Reduction: To reduce the search space of DSE, we make use of the *reuse factors* for each of the IFM, OFM and WHTs for each layer. The reuse factor is the number of times a value of IFM, OFM or WHTs is accessed. The reuse factors are provided in Eq. 8

$$_{I}{}_l^i = m_l^i k_l^2 \tag{8a}$$
$$_{O}{}_l^i = n_l^i k_l^2 \tag{8b}$$
$$_{W}{}_l^i = dr_l^i c_l^i \tag{8c}$$

Fig. 4. A detailed ow of SuperSlash methodology illustrating different steps and their interconnection

As an example, each value of IFM is used to compute k_l^2 values for each of them m_l^i lter channels. Similarly, each value of OFM is computed by accumulating $n_l^i k_l^2$ partial sums. For WHTs, each parameter has to be accessed $r_l^i c_l^i$ times, for all the IFM in a batch. Thus, rather than evaluating all the data reuse strategies, we only compute the data reuse factor (Eq. 8) for each layer, and then select the data reuse strategy with the maximum reuse factor for that particular layer. This is elaborated in Table I. To justify our approach, we consider the rst row of Table I that relates to limiting the design space to IRO. Eq. 8a shows that a higher number of OFM channels (m_l^i) leads to a higher data reuse factor of IFM (γ_l^i). And from Eq. 4 and 3a, it can be concluded that higher m_l^i results in higher off-chip memory access volume for ORO and WRO, and does not affect IRO. Therefore, a decision to select IRO based on its higher data reuse factor is justi ed.

Table I also provides reference to the search ranges for tile size parameters ($t_{m_l^i}; t_{n_l^i}; t_{r_l^i}; t_{c_l^i}$) that are detailed in Table II. These search ranges provide a reduced search space, as compared to an exhaustive search, by removing some sub-optimal design points. Again, we justify the use of these search spaces by considering IRO. A reduced search space (Range II, III, and IV) is suggested as per Table II. This search space is supported by Eq. 3 and 4, which dictate that higher values of $t_{m_l^i}$, $t_{r_l^i}$ or $t_{c_l^i}$ require lower off-chip memory access volume. We thus keep these values to their maximum in II, III, and IV search range. This is because greater number of IFM channels processed together can reduce the off-chip transfer for partial sums. For ORO, higher values of $t_{m_l^i}$, $t_{r_l^i}$ and $t_{c_l^i}$ favor lower off-chip memory access volume since more number of lters processed together can reduce the off-chip transfer of IFM tiles. We thus keep these values to their maximum in I, III and IV search range. A similar justi cation supports the search range for WRO. We also provide the algorithmic description of these search spaces in Figure 5. For MLF, the l^{th} layer employs ORO and hence the Ist, IInd and IIIrd search range is explored. For MLF, the search space of two consecutive layers being evaluated for layer-fusion is explored together. Thus, the last column in Table III provides the search range for every $l+1^{th}$ layer that is being fused with l^{th} layer. Algorithm 1 summarises the procedure of solving DSE for off-chip memory access volume of a given model W^i. By selecting the data reuse strategy, Algorithm 1 comes-up with the tile con guration offering the least off-chip memory access within the speci ed search space.

B. Ranking Function & Model Compression

In step 2 of our methodology, SuperSlash assigns a rank q_u^i to each u^{th} layer of the network (W^i). The ranking function is thus de ned by $Q^i = \{q_1^i; q_2^i; ::q_u^i; :::; q_L^i\}$ such that:

$$Q^i = \{q_u^i \mid V_{q_u^i}^i > V_{q_{u+1}^i}^i\} \text{ where } u = \{1; 2; 3; ::::; L\} \quad (9)$$

Thus, q_1^i and q_L^i provide the layer number of the network layer with the highest and lowest off-chip memory access volume, respectively. The ranking function is intended to guide the pruning process. However, there is a caveat. Model compression based solely on the proposed layer ranking does not consider the loss in accuracy incurred due to the pruning of a particular layer. So, in Step 2, we apply our "Model Exploration" technique that leverages the DSE information provided by the ranking function and also considers the accuracy loss due to pruning. Our model exploration initially selects the layer, which is at the top of the rank (i.e. q_1^i), and then computes the signi cance scores of all the lter parameters of this layer using the L_1 norm. The parameters of this layer are then pruned by discarding a certain number of lters with the lowest L_1 norm, as de ned by a pre-de ned pruning ratio (PR). We empirically de ned a progressively increasing order of pruning ratio, with $PR = \{5\%; 10\%; 15\%; 20\%; 25\%\}$. Small pruning ratio increases the number of pruning iterations, while larger pruning ratios decrease the learning capability of the network since it tends to remove a large number of weights at once. Our progressively increasing PR allows us to gradually increase the pruning ratio as long as the accuracy loss is within the tolerable limits. For the special case, when the accuracy drops below the accuracy tolerance just by pruning 5% lters, the next candidate layer, as per the ranking function, is selected for pruning. As the PR is increased for a particular layer, if the accuracy of the pruned model falls below the accuracy tolerance, the pruned model W^i is sent again for off-chip memory access volume computation, and Step 1-3 is repeated. The process continuous until removing lters from any layer results in the violation of the accuracy constraint. At this stage, the pruned model is sent to Step 4 for global ne-tuning. Algorithm 2 describes the complete ow of this Model Exploration step.

C. Global Fine Tuning

Whenever a pruned model obtained during the previous step violates the given accuracy constraint, the model is globally ne-tuned for a few epochs to regain the lost accuracy. After ne-tuning, the model is further pruned by following Steps 1, 2, and 3 repeatedly. Since the accuracy is restored in this step, it allows more weights and connections to be pruned under a given accuracy threshold. In case ne-tuning is unable to bring back the accuracy within the accuracy constraint, the last-known acceptable model is passed on to the nal step.

Fig. 5. SuperSlash search ranges of i^{th} layer for different data reuse strategies (D_l^i).

D. Retraining and DSE

When the desired memory/accuracy constraint is reached, the entire model is retrained for some epochs, and the nal model is passed on to the DSE block to determine the nal design con guration for a minimum number of off-chip memory accesses. SuperSlash can also be used to prune the network to a desired pruning ratio (C) or off-chip memory access level (V), such that $\sum_{l=1}^{L} \; {}^i_l \quad C$ or $\sum_{l=1}^{L} v_l^i \quad V$. This is achieved by putting a relaxed accuracy threshold, and using the desired pruning ratio or off-chip memory access level as the terminating condition.

```
Algorithm 1: DSE for off-chip memory access volume for
the model at i^th cycle
   Input: $M^i; N^i; R^i; C^i; K^i; S^i; P^i; B$
   Output: $V_{min}^i; t_m^i; t_n^i; t_r^i; t_c^i; B^i; D^i$
1  for (all layers) do
2      Select data reuse strategy ($D_l^i$) and search ranges using
        Tables I and II
3      for all ranges do
4          Choose a tile con guration $\{t_m, t_n, t_r, t_c\}$
5          Calculate $B_l^i$ and $V_l^i$ using equations 1-5
6          if ($V_l^i \quad V_{min}{}_l^i$ and $B_l^i \quad B$) then
7              $V_{min}{}_l^i \quad V_l^i$
8              $\{t_m{}_l^i; t_n{}_l^i; t_r{}_l^i; t_c{}_l^i\} \quad \{t_m, t_n, t_r, t_c\}$
           end
       end
   end
```

```
Algorithm 2: Model Exploration for model at i^th cycle
   Input: $X; Y; W^i; R^i; A_{th}$
   Output: $W^i$
   begin
1      $l$    rst layer in rank($Q^i$)
2      for x in PR do
3          $W_l^i(x)$    x% $l_{th}$ layer pruned model
4          $A_p$    Accuracy($X; Y; W_l^i(x)$)
5          if $A_p < A_{th}$ then
               if x is not rst element in PR then
6                  return $W_l^i(x)$
               else
                   if l is last in rank $Q^i$ then
7                      return $W^i$ for global ne tuning
                   else
8                      $l$    next layer in rank $Q^i$
9                      Go to step 2
                   end
               end
           end
       end
10     Return $W_l^i$
   end
```

IV. Results and Discussions

Since SuperSlash provides a uni ed scheme for DSE and pruning based model compression, we demonstrate the bene ts achieved due to this co-exploration by performing multiple evaluations. First, we provide our experimental settings and training details. Afterwards, we provide an in-depth comparative analysis of the results of our DSE with that of SmartShuttle [12]. We further demonstrate that for a number of popular DNNs (VGG16 [19], ResNet56/110 [1] and MobileNetV1 [34]), our pruned models are comparable, in terms of accuracy, to that of state-of-art pruning schemes [15] [17] [16] for the same pruning ratio, while providing lower off-chip memory access volume.

A. Experimental Setup

In order to provide a fair comparison, we selected four networks that have been mostly evaluated by the state-of-the-art pruning and DSE schemes to report their results. These networks are VGG16, ResNet56, ResNet110 and MobileNetV1, trained on CIFAR-10 dataset.

VGG16 is a high capacity network originally designed for the large scale ImageNet dataset. Li et al. [15] pruned a modi ed version of VGG16 (13 convolutional layers and 2 fully connected layers) on small scale image classi cation dataset, CIFAR-10, and achieved state of the art accuracy. We use the same modi ed VGG16 architecture, applied batch normalization after each layer except for classi cation layer and trained the network with data augmentation, learning rate decay, SGD optimizer and a batch size of 128.

ResNet architecture for CIFAR-10 comprises three stages of residual blocks. Input feature map sizes for rst, second, and third stages are $32 \quad 32$, $16 \quad 16$ and $8 \quad 8$, respectively. The same number of residual blocks are present in each stage, with two convolutional layers in each residual block. In the ResNet architecture, the input of each block is added to the output of the same block via a skip connection. Hence, the dimensions of the input and output of the residual block should remain the same. We thus prune only the first layer of a residual block as this results in the removal of only the corresponding intermediate channels and it does not change the number of output feature maps. For training, We use data augmentation, learning rate decay, ADAM optimizer, and a batch size of 32 to train the baseline model of ResNet56/110.

We have also evaluated SuperSlash on MobileNetv1, which comprises depth-wise and point-wise convolutional layers. Although most of the pruning techniques do not commonly prune such small networks, we still evaluate SuperSlash on small networks like MobileNetv1, which is designed for embedded systems. For training, We use data augmentation, learning rate decay, SGD optimizer, and a batch size of 128 to train the baseline model of MobileNetv1.

SuperSlash has been evaluated by synthesizing a Neural Processing-Array based accelerator (such as shown in Figure 3) using Verilog hardware description language in Xilinx's Vivado Design Suite. GBUF size for all results presented throughout the upcoming text is 32KB unless specified. The off-chip memory access volume has been computed by assuming 16-bit numbers, and by logging all the data requests to DRAM. The unified DSE of the SuperSlash was implemented in Keras [35]. The co-exploration is provided in the form of a push-button framework for optimizing the deep learning accelerators. The network parameters, test dataset, training code, and constraints are provided to the framework, which returns the design parameters along with the compressed model without any manual input.

B. Detailed Comparison with SmartShuttle on VGG16

SmartShuttle [12] provides a DSE scheme for selecting the most suitable tile size (for IFM, WHTs, and OFM) and data reuse strategy for each layer of a DNN. They demonstrated substantial benefits of this layer-wise adaptive design reuse selection. Furthermore, they also demonstrated that exploring such a design space for optimizing the off-chip memory access volume is orthogonal to the DSE techniques being used in computing engine designs. This is because their technique was only aimed at reducing the off-chip memory access volume, and it does not directly explore the design for throughput, area, or latency. For model compression, SmartShuttle has to rely on independently pruned models, such as the one by Li et al. [15]. As compared with SmartShuttle, the Design Space of SuperSlash is much wider. This is because we consider layer-fusion, reuse based search space exploration and a closely coupled layer-wise pruning ratio. This allows us to further reduce the off-chip memory access volume by guiding the pruning to remove the parameters that require greater off-chip memory access volume. Li17 [15] reported their detailed results by employing a 64% pruned model for VGG16. Thus, to perform a fair comparison, we employed our SuperSlash methodology to perform the unified DSE and model compression for 64% pruning ratio. SuperSlash iteratively explores multiple pruned models and designs the DNN model with minimum access volume with comparable accuracy. Table III shows that:

1) For a comparable pruning ratio, our model provides 49% lower memory access volume as compared to SmartShuttle with the model of Li17 [15].
2) The layer-wise analysis reveals that while the total number of pruned parameters are similar, our scheme applies a different pruning ratio for each layer as compared to that of [15].

The table also provides the layer-wise details of the data reuse strategy (D_l^i), as suggested by SuperSlash.

For further insights, in Figure 6, we plot the layer-wise pruning ratio of SuperSlash and compare it with that of Li et al. [15]. Furthermore, for each layer, we plot the off-chip memory access volume as required by the original network with SmartShuttle DSE, Li et al. [15] network with SmartShuttle DSE, and Li et al. [15] network with SuperSlash DSE. It can be observed that SuperSlash applied a greater pruning ratio to the layers that have a higher off-chip memory access volume in the original network.

Fig. 6. Comparison of layer-wise pruning ratio [right] and off-chip memory access volume[left]. SuperSlash provides a layer-wise pruning ratio that is more responsive to the layer-wise off-chip memory access volume.

Fig. 7. Effectiveness of data reuse strategy (D_l^i) selection for VGG16 on CIFAR-10, based on (a) data reuse factor: select D_l^i with higher data reuse factor (b) data volume: select D_l^i with higher data volume. It can be observed that decision based on data volume is only suitable for earlier layers, whereas data reuse factor based decision is effective on the whole network.

Unlike SmartShuttle, which selects ORO or WRO depending upon the data volume of OFM or WHTs, SuperSlash utilizes the data reuse factor of IFM, OFM, and WHTs to choose a particular reuse order. We demonstrate the benefit of using the data reuse factor using Figure 7. In Figure 7a, we plot layer-wise off-chip memory access volume for IRO, ORO, and WRO, alongside the data reuse factor. In Figure 7b, we plot layer-wise off-chip memory access volume for ORO and WRO, alongside the data volume. Considering the 9th convolutional layer, it can be observed that the data reuse-based decision favors ORO or IRO, while the data volume-based decision favors WRO. However, as shown, the off-chip memory access volume is much lower for ORO (and IRO) as compared to WRO. Thus, the figure highlights that, particularly for deeper layers, data reuse provides a better decision metric. For the cases when the data reuse factors are the same, we make use of the layer-wise volume to resolve the decision.

Next, we evaluated the global buffer size impact on the off-chip memory access volume for VGG16. We performed our SuperSlash DSE for various values of global buffer size varying from 32KB to 512KB, and report the total off-chip memory access volume in Figure 8. We observe that our SuperSlash provides greater gains when the global buffer size is small. This is because, for smaller global buffer size, more data is required to be shuttled, and hence there is a relatively greater benefit of off-chip memory aware pruning.

In Figure 9, we show that SuperSlash provides a broader design space by effectively adding another design axis, the pruning ratio. We thus plot the DSE for VGG16, evaluated over multiple global

TABLE III
VGG16 ON CIFAR-10: OUR PRUNING CANDIDATES PROVIDE BETTER ACCURACY WITH REDUCED OFF-CHIP MEMORY ACCESS VOLUME

Original Network			Remaining Filters		Remaining Parameters (Millions)		Remainig FLOPs (Millions)		DRAM access Volume (MB)				
#	IFM	Filters	Li17	Ours	Li17	Ours	Li17	Ours	Smart-Shuttle	Super-Slash	D_l^i	MLF	Tile Con gurations {tm, tn, tr, tc}
C1	3 32 32	64	32	46	0.001	0.001	0.88	1.27	0.08	0.02	ORO	YES	16,3,22,32
C2	64 32 32	64	64	64	0.018	0.027	18.87	27.13	0.52	0.76	IRO	YES	6,16,11,16
C3	64 16 16	128	128	128	0.074	0.074	18.87	18.87	0.83	0.36	IRO	NO	1,58,16,16
C4	128 16 16	128	128	128	0.148	0.148	37.75	37.75	1.72	0.38	ORO	NO	11,44,16,16
C5	128 8 8	256	256	256	0.295	0.295	18.87	18.87	1.95	0.61	IRO	NO	8,111,8,8
C6	256 8 8	256	256	139	0.590	0.320	37.75	20.50	3.87	0.66	ORO	NO	1,221,8,8
C7	256 8 8	256	256	256	0.590	0.321	37.75	20.50	3.87	0.64	IRO	NO	1,221,8,8
C8	256 4 4	512	256	306	0.590	0.705	9.44	11.28	1.81	1.36	IRO	NO	6,220,4,4
C9	512 4 4	512	256	176	0.590	0.485	9.44	7.76	1.81	0.94	ORO	NO	3,323,4,4
C10	512 4 4	512	256	359	0.590	0.569	9.44	9.10	1.81	1.10	IRO	NO	6,216,4,4
C11	512 2 2	512	256	227	0.590	0.734	2.36	2.93	1.30	1.40	ORO	NO	5,308,2,2
C12	512 2 2	512	256	216	0.590	0.442	2.36	1.77	1.30	0.85	ORO	NO	6,253,2,2
C13	512 2 2	512	256	512	0.590	0.996	2.36	3.98	1.30	1.90	IRO	NO	5,324,2,2
FC1	512 1 1	512	512	512	0.132	0.263	0.13	0.26	0.27	0.41	ORO	YES	112,142,1,1
FC2	512 1 1	10	10	10	0.005	0.005	0.01	0.01	0.01	0.01	IRO	YES	1,112,1,1
Total					5.39	5.38	206.3	181.98	22.44	11.39			

Fig. 8. Comparison with SmartShuttle for various GBUF sizes

buffer sizes as the network is pruned for four pruning ratios (22%, 36%, 50% and 64%). Since there are millions of design points for each model, we represent the range of the design points explored by SuperSlash on a vertical line for each of the global buffer sizes. It shall be noted that all these points are explored automatically without any manual intervention. For the sake of comparison, we also plot the best design point obtained by performing SmartShuttle DSE over the SuperSlash model. It can be observed that SuperSlash DSE is always providing points with lower off-chip volume as compared to SmartShuttle for all the cases being considered. This is due to the extended design space traversed by SuperSlash owing to its better empirical rules for search space minimization and also due to the higher number of reuse orders supported. Thus, SuperSlash allows us to include some extra points in our DSE that are not present in SmartShuttle's design space and hence provides us candidate design points that are closer to the best design points from exhaustive search (Green * in Figure 9). Hence, our evaluations demonstrate that:

1) SuperSlash provides an extended design space by incorporating the accuracy dimension in the DSE. This is achieved by adding the pruning ratio in our uni ed exploration.
2) Uni ed design space and model compression (via pruning) methodology provides us with design points that have lower off-chip memory access volume as compared with the state-of-art, without any manual intervention.

C. Analysis of SuperSlash DSE and Model Exploration

SuperSlash provides a push-button methodology for uni ed Design Space Exploration and Model Compression aiming at reduced off-chip memory access. Here we provide an in-depth analysis of the ef cacy of our automated methodology. We set an accuracy threshold of 91% for VGG16 and got 9 intermediate models with model accuracy ranging from 93.4% to 91.3%. We then performed a thorough analysis to demonstrate that we provide design points with lower off-chip memory access volume as compared to state of the art. A combination of a network model (depending upon the pruning technique) and its corresponding DSE (SmartShuttle or SuperSlash DSE) provides us with a particular design point in the Accuracy vs Off-chip memory-access volume design space. In Figure 10, we thus provide the accuracy of various variants (original and pruned) of VGG16 network against their off-chip memory access volume depending upon the chosen DSE (SuperSlash DSE or SmartShuttle). Here SuperSlash DSE means that only Step-1 of SuperSlash is carried out. It can be observed that for the same model (both for Original and Li17) SuperSlash DSE provides lower off-chip memory access volume. It can also be observed that our uni ed methodology provides two design points (A and B) that provide similar accuracy (93.20% and 93.40%) while requiring (52% and 31%) lower off-chip memory access volume as compared to Li17 Model + SuperSlash DSE (93.40% Accuracy while requiring 11.54MB off-chip memory access volume). For completeness, we also plot the off-chip memory access volume, as required by DSE of SmartShuttle for the 9 VGG16 models generated by our uni ed SuperSlash methodology. It can

Fig. 9. SuperSlash design space exploration. The black and red triangles show the best design points provided by SuperSlash and SmartShuttle, respectively. Blue crosses represent the entire design space searched by SuperSlash. The green asterisks indicate the best design points as returned by exhaustive search.

TABLE IV
COMPARISON WITH STATE-OF-ART PRUNING TECHNIQUES

Baseline	Network	Pruning Ratio (%)	Accuracy (%)	FLOPs (Millions)	Off-Chip Access Volume (MB)	
					SmartShuttle [12]	Ours
VGG16 on CIFAR-10	Original	0	93.40	313.46	56.14	-
	Li17 [15]	64	93.40	206.28	22.44	-
	Our-64	64	93.30	181.98	20.14	11.39
	Salama19 [17]	86	93.39	169.41	11.34	-
	Our-86	86	93.00	106.13	8.54	4.79
	Our-94	94	91.30	89.75	3.55	2.36
ResNet56 on CIFAR-10	Original	0	92.49	125.75	6.60	-
	Li17 [15]	14	93.06	89.94	5.77	-
	Our-14	14	92.63	116.30	5.95	4.58
	NISP18 [16]	42	92.46	92.96	4.77	-
	Our-42	42	92.60	75.35	4.86	3.60
ResNet110 on CIFAR-10	Original	0	92.65	253.15	13.10	-
	Li17 [15]	32	93.50	170.50	10.21	-
	Our	32	92.60	150.48	9.88	6.91
MobileNetV1 on CIFAR-10	Original	0	86.70	32.35	11.89	-
	Liu'19 [36]	43	85.01	13.51	6.11	-
	Our	43	85.02	21.78	6.84	4.33

be observed that despite using the models generated by SuperSlash, the bene t in terms of off-chip access volume is lagging to that of SuperSlash DSE. Thus, SuperSlash not only provided models that provide accuracies that are comparable with state-of-the-art but also require a lower off-chip memory access volume. Furthermore, unlike Li17, the pruning scheme of SuperSlash is completely automated. As per Li17, the pruned network was manually tuned for better accuracy.

Fig. 10. Design space of Accuracy vs Off-Chip Access Volume plotted for VGG16. Provides design points corresponding to various combination of Model and DSE. Also compares the 9 points provided by the SuperSlash uni ed methodology to that of Smart Shuttle DSE on the 9 SuperSlash models. It can be observed that SuperSlash provides models that provide accuracies that are comparable with state-of-the-art while requiring a lower off-chip memory access volume.

D. Comparison with SOA pruning methods

We performed a comparison with multiple state-of-art pruning techniques [15]–[17] for popular networks [37], [38]. For a fair comparison, we considered only those pruning schemes for which the authors provided a layer-wise pruning ratio for their chosen network. Furthermore, for our SuperSlash exploration, we aimed for the same pruning ratio as achieved by the scheme with which we are comparing. Table IV provides the accuracy achieved, parameters remaining, and FLOPs remaining after the application of pruning. After obtaining desired models, we converted weights of each these to 16-bit by applying post-training quantization and obtain negligible loss in accuracy. We combined each of the pruning schemes to SmartShuttle and compared the results with a SuperSlash alternative.

For VGG-16, SuperSlash is applied on a pre-trained VGG16 model with a constraint on model size. Pruning ratio is set to 64% and 86% for the direct comparison with the state of the art pruning algorithms [15] and [17] respectively. Our 64% pruned model's accuracy 93.2% is slightly less than Li's Accuracy 93.4% [15], but our model consumes just 11.39 MB of off-chip memory access volume while Li's model [15] consumes almost double 22.44 MB. Due to our layer ranking step, SuperSlash allocates higher pruning ratio to the layers with more off-chip memory access volume as shown earlier in Figure 6. Similarly, Our 86% pruned model achieves an accuracy of 93.00% and off-chip memory access volume of 4.79 MB. Salama et al. [17] at the same pruning ratio manages to achieve an accuracy of 93.39%, which is slightly more than SuperSlash, but it consumes 11.34 MB of off-chip memory access volume. Hence it shows that with comparable accuracy and model size, our model consumes signi cantly less amount of off-chip memory access volume as compared to state of the art [15] [17].

For ResNet56, the pruning ratio is set to 14% and 42% for their comparison with [15] and [16], respectively. The accuracy of our 14% pruned model is 92.63%, which is slightly less than Li's [15] 14% pruned model. But off-chip memory access volume of our model is 4.58 MB, which is considerably low than Li's [15] model which consumes 5.77 MB. Our 42% pruned model outperforms [16] in the accuracy as well as in-terms of off-chip memory access volume. Our model's accuracy (92.6%) is slightly better, and off-chip memory access volume (3.60 MB) is signi cantly less as compared to Yu's model [16] accuracy (92.45%) and off-chip memory access volume (4.77MB).

For ResNet110, we set pruning ratio to 32% to make it comparable with the model of Li et al. [15]. From Table IV, it can be observed that SuperSlash comes up with a model with less off-chip memory access volume even on the DSE of SmartShuttle. Moreover, our DSE on Li17 model also performs better than SmartShuttle.

For MobileNetv1 [34], pruning ratio is set to 43% for its comparison with [36]. Liu et al. [36] used ImageNet for its results. Due to lack of resources, we used Liu's pruned model [36] and test it on the CIFAR-10 dataset. Table IV shows that with comparable accuracy, off-chip memory access volume of our pruned model is 4.33 MB while that of SmartShuttle [12] on Liu's [36] is 6.11 MB. Hence, we have saved 29.02% more off-chip memory access volume as compared to the state-of-the-art. [36] [12]. In summary for all the evaluated models, SuperSlash outperforms the state of the arts in off-chip memory access volume with less than 1% accuracy drop. Thus, we demonstrate that:

1) SuperSlash provides comparable accuracy for a number of popular DNNs as compared to state-of-art
2) Our technique provides design points that require low off-chip memory access volume as compared to state-of-art, thus demonstrating the robustness of the methodology.

E. Applicability of SuperSlash for DNNs with stacked layers

SuperSlash can be applied to DNN structures that group multiple convolutional layers in the form of stacks. A stack within ResNet is de ned as, the set of consecutive layers with identical feature map dimensions. In contrast to VGG16, for ResNet, we prune the layers based on the off-chip memory access volume of a stack, and thus the ranking function is applied on the stack. An equal number of parameters are then removed from each layer of the stack depending upon the stack's rank. Thus, to show the applicability of SuperSlash on such DNNs, we provide stack wise pruning analysis in Figure 11 and compare it with that of state of the art. SuperSlash-14 and SuperSlash-42 refer to the model points generated by SuperSlash with 14% and 42% pruning ratio, respectively. The bar plots provide the stack-wise off-chip memory access while the line plots provide the pruning ratio as suggested by a particular pruning method. For further insights, we plot the stack-wise pruning ratio of ResNet56 and compare it with that of Li17 and NISP18. For each stack, we plot off-chip memory access volume as required by the Original network with SmartShuttle, Li17 network with SmartShuttle, NISP18 network with SmartShuttle, SuperSlash-14 and SuperSlash-42, respectively. It can be observed that SuperSlash applies a greater pruning ratio to the layers that have a higher off-chip memory access volume in the original network.

Fig. 11. The gure provides stack wise pruning analysis by comparing SuperSlash with state-of-art for ResNet56. The bar plots provide the stack-wise off-chip memory access while the line plots provide the pruning ratio as suggested by a particular pruning method. SuperSlash-14 and SuperSlash-42 refer to the model points generated by SuperSlash with 14% and 42% pruning ratio, respectively. It can be observed that SuperSlash applied a higher pruning ratio to stacks which required a higher off-chip memory access volume in the original model.

F. Timing Analysis of SuperSlash

We have performed a detailed timing analysis of our proposed SuperSlash methodology. The time to perform our uni ed methodology depends upon the model size, model structure and design constraints (A_{th}, GBUF). Each iteration of SuperSlash comprises two major steps: DSE (Step 1, in Figure 4) and Model Exploration (Step 2-4, in Figure 4). In Table V, we report the time taken by these two steps (DSE and ME) during each iteration of SuperSlash, as the VGG16 network is pruned up to 64.3% with GBUF = 32KB. It can be observed that most of the time is taken by Model Exploration step (ME) since it also involves ne-tuning of the pruned network. The time to perform DSE reduces as the network size reduces. We also report the number of design points explored by SuperSlash DSE and compare that with that of an exhaustive exploration. Since SuperSlash DSE considers SLP and MLF, we report the number of design points for each of these, if the exhaustive evaluation was performed. SuperSlash considerably reduces the total number of design points (92% search space reduction) by utilizing the search space reduction strategies provided in Table I and Table II of paper.

G. Evaluation of SuperSlash for existing Hardware Accelerators

SuperSlash provides a DSE scheme for selecting the most suitable tile size (for IFM, WHTs, and OFM) and data reuse strategy aimed at reducing the off-chip memory access. A similar related approach, SmartShuttle, demonstrated that such a DSE aimed at minimizing the off-chip memory access is orthogonal to the DSE techniques being used in computing engine designs. This is because their technique was only aimed at reducing the off-chip memory access volume and it does not directly explore the internal access pattern of the on-chip memory. To demonstrate this, they integrated their methodology to FlexFlow [11]. As a case study, we also demonstrate the results of SuperSlash methodology when integrated with two hardware accelerators, i.e. FlexFlow and MPNA [39]. Various data reuse strategies of SuperSlash can be enabled or disabled depending upon the hardware accelerator being considered. Furthermore, if a hardware accelerator has a separate GBUF for IFM and OFM, it can be conveniently incorporated in SuperSlash by forcing individual constraints of buffer size on IFM, OFM or WHTs. Below we provide architectural details of each one of them along with the consideration used to integrate SuperSlash for reducing the off-chip memory access for these hardware accelerators.

FlexFlow: CNNs are parallel by nature, and most hardware accelerators are designed to exploit this parallelism to accelerate the processing. However, due to the limitations of the data reuse orders and other static design decisions, these accelerators can support only a limited number of types of parallelism and the associated data reuse orders. Lu et al. in [11] proposed a exible data ow architecture, FlexFlow, that overcomes this limitation by exploiting the complementary effects among different types of parallelism. It is composed of three key components: a convolutional unit, a pooling unit, and three on-chip buffers (two neuron/activation buffers and one kernel buffer). The convolutional unit is an array of processing elements (PEs), where PEs are interconnected in a manner that the unit is recon gurable and is able to support any parallelism. Each PE consists of logic to perform multiply-and-accumulate (MAC) operation and local memory to store kernel and activations. The pooling unit consists of a 1D array of lightweight ALUs to perform sub-sampling after convolution, wherever required. The on-chip kernel and neuron buffers are arranged in the form of banks. Each buffer has D number of banks, where D is the dimensions of the processing element (PE) array. The data movement between on-chip buffers and DRAM is de ned as follows: Data can be read from DRAM and moved to neuron buffers and vice versa, and Weights can only be moved from DRAM to the kernel buffer. To fully exploit SuperSlash we assume that the role of the two neuron buffers can be interchanged, i.e., the neuron buffer that was providing the input activations will now store partial sums/output activations, and the other will start providing input activations (i.e., the output stored before the interchange). This allows us to exploit the MLF for SuperSlash. The sizes of weight and data buffers are 32KB and 32KB, respectively.

MPNA: In CNNs, the convolutional layers are considered the most

TABLE V
TIMING ANALYSIS FOR SUPERSLASH METHODOLOGY FOR VGG16 NETWORK. THE TABLE ALSO PROVIDES THE NUMBER OF DESIGN POINTS EXPLORED BY SUPERSLASH, SMARTSHUTTLE AND EXHAUSTIVE DSE CONSIDERING A GBUF OF 32KB. SMARTSHUTTLE PROVIDES 92% REDUCTION IN SEARCH SPACE AND EXECUTION TIME FOR DSE AS COMPARED TO EXHAUSTIVE SIMULATIONS

VGG16 on CIFAR-10	SuperSlash Iteration							
	1	2	3	4	5	6	7	-
Pruning Ratio (%)	27	42.5	46.4	57.5	57.8	62	64	-
	Time (Minutes)							Total
SuperSlash DSE	3.66	2.64	2.29	1.91	1.66	1.49	1.46	15.11
SuperSlash ME	5.13	5.23	4.9	5.24	6.32	5.58	4.92	37.32
SuperSlash Total	8.79	7.87	7.19	7.15	7.98	7.07	6.38	52.43
Exhaustive DSE	37.54	29.32	29.28	24.79	26.63	23.36	25.4	196.33
	Design Points (Million)							Total
SuperSlash	2465	1635	1088	920	669	581	498	7856
SmartShuttle	0.0436	0.0402	0.0378	0.0366	0.0356	0.0342	0.0335	0.2615
Exhaustive DSE (SLP)	35	29	25	23	22	20	19	173
Exhaustive DSE (MLF)	25298	18128	13902	11968	10760	9091	8681	97828

TABLE VI
PERFORMANCE OF SUPERSLASH WHEN INTEGRATED WITH STATE OF THE ART ARCHITECTURES THE TABLE PROVIDES THE VALUES OF ACCURACY, FLOPS AND OFF-CHIP MEMORY ACCESS VOLUME FOR ORIGINAL AND PRUNED NETWORKS WITH SMARTSHUTTLE DSE, COMPARED WITH THAT OF SUPERSLASH

VGG16 for CIFAR-10	Pruning Ratio (%)	Accuracy (%)	FLOPS (Millions)	Volume (MB)
FlexFlow				
Original Model + SmartShuttle DSE	0	93.40	313.46	42.40
Li17 Model + SmartShuttle DSE	64	93.40	206.28	16.25
SuperSlash-64	64	93.00	181.56	10.76
SuperSlash-78	78	92.50	128.18	6.94
MPNA				
Original Model + SmartShuttle DSE	0	93.40	313.46	31.99
Li17 Model + SmartShuttle DSE	64	93.40	206.28	12.06
SuperSlash-64	64	93.38	203.47	9.88
SuperSlash-78	78	93.10	125.08	6.69

compute-intensive layers. Therefore, typically the hardware accelerators for CNNs mainly focus on accelerating only the convolutional layers and overlook the fully connected layers. MPNA [39] proposed an accelerator composed of two heterogeneous systolic arrays to accelerate both the convolutional layers and the fully connected layers in embedded settings. Each systolic array is composed of 8 8 processing elements (PEs), where each PE contains logic to compute multiply-and-accumulate (MAC) operations and some registers to store weights and activations. Apart from heterogeneous systolic arrays, the accelerator consists of the two disjoint buffers: one for weights and the other for data (i.e., for IFMs and OFMs), an accumulation unit for accumulating the partial sums, and a pooling and activation unit for performing pooling and activation operations. The sizes of weight and data buffers are 36KB and 256KB, respectively. For SuperSlash, the following data movement through the memory hierarchy of MPNA was followed: Weights and input activations are read from the DRAM and always moved to the on-chip weight and data buffers, respectively. Partial sums are always kept on-chip in the accumulation unit. Once the output activations are ready, they are moved to the on-chip data memory and then to the DRAM. This particular hardware accelerator effectively supports only ORO, and accordingly, IRO and WRO were not considered for MPNA. Also, since MPNA considers only one layer at a time, the MLF was not considered while performing SuperSlash DSE. In Table VI, we provide the bene t achieved in terms of reduction in off-chip memory access. For each of the hardware, we evaluated the results of SmartShuttle DSE with the original and pruned model (Li17 Model) and compared with that of two SuperSlash models. In the table, SuperSlash-64 and SuperSlash-78 incorporate 64% and 78% pruning, respectively. It can be observed that SuperSlash models require lower off-chip memory access as compared to the state-of-the-art on all the hardware under consideration.

V. CONCLUSIONS

We have devised a uni ed methodology for model compression and design space exploration. Provided a DNN model, we iteratively prune the layers based on layer-wise off-chip memory access volume estimate. This allows the pruning to be steered towards design points that require low off-chip memory access volume. We have achieved 64% reduction in VGG16 model size and saved 49% off-chip memory access volume with negligible loss in the accuracy. Similarly, we pruned ResNet56/110 and MobileNetV1 on CIFAR-10 and achieved a signi cant reduction in memory with an accuracy drop of less than 1%.

ACKNOWLEDGEMENT

This work is supported by the Higher Education Commission (HEC) of Pakistan, under NRPU project # 10150, titled, “AxVision - Application-Speci c Data-Aware, Approximate-Computing for Energy Ef cient Image and Vision Processing Applications.”

REFERENCES

[1] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778. 1, 6
[2] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, “Going deeper with convolutions,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 1–9. 1

[3] M. Bojarski, D. Del Testa, D. Dworakowski, B. Firner, B. Flepp, P. Goyal, L. D. Jackel, M. Monfort, U. Muller, J. Zhang *et al.*, "End to end learning for self-driving cars," *arXiv preprint arXiv:1604.07316*, 2016. 1

[4] S. Wu, D. Hu, S. Ibrahim, H. Jin, J. Xiao, F. Chen, and H. Liu, "When fpga-accelerator meets stream data processing in the edge," in *2019 IEEE 39th International Conference on Distributed Computing Systems (ICDCS)*. IEEE, 2019, pp. 1818–1829. 1

[5] S. Du, T. Huang, J. Hou, S. Song, and Y. Song, "Fpga based acceleration of game theory algorithm in edge computing for autonomous driving," *Journal of Systems Architecture*, vol. 93, pp. 33–39, 2019. 1

[6] X. Xu, Y. Ding, S. X. Hu, M. Niemier, J. Cong, Y. Hu, and Y. Shi, "Scaling for edge inference of deep neural networks," *Nature Electronics*, vol. 1, no. 4, pp. 216–222, 2018. 1

[7] Y.-H. Chen, J. Emer, and V. Sze, "Eyeriss: A spatial architecture for energy-efficient dataflow for convolutional neural networks," in *ACM SIGARCH Computer Architecture News*, vol. 44, no. 3. IEEE Press, 2016, pp. 367–379. 1, 2

[8] T. Chen, Z. Du, N. Sun, J. Wang, C. Wu, Y. Chen, and O. Temam, "Diannao: A small-footprint high-throughput accelerator for ubiquitous machine-learning," *ACM SIGARCH Computer Architecture News*, vol. 42, no. 1, pp. 269–284, 2014. 1

[9] C. Zhang, P. Li, G. Sun, Y. Guan, B. Xiao, and J. Cong, "Optimizing fpga-based accelerator design for deep convolutional neural networks," in *Proceedings of the 2015 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*. ACM, 2015, pp. 161–170. 1, 2

[10] A. Rahman, J. Lee, and K. Choi, "Efficient fpga acceleration of convolutional neural networks using logical-3d compute array," in *2016 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 2016, pp. 1393–1398. 1, 2

[11] W. Lu, G. Yan, J. Li, S. Gong, Y. Han, and X. Li, "Flexflow: A flexible dataflow accelerator architecture for convolutional neural networks," in *2017 IEEE International Symposium on High Performance Computer Architecture (HPCA)*. IEEE, 2017, pp. 553–564. 1, 2, 10

[12] J. Li, G. Yan, W. Lu, S. Jiang, S. Gong, J. Wu, and X. Li, "Smartshuttle: Optimizing off-chip memory accesses for deep learning accelerators," in *2018 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 2018, pp. 343–348. 1, 2, 3, 6, 7, 9

[13] S. Han, J. Pool, J. Tran, and W. Dally, "Learning both weights and connections for efficient neural network," in *Advances in neural information processing systems*, 2015, pp. 1135–1143. 1, 2, 3

[14] W. Wen, C. Wu, Y. Wang, Y. Chen, and H. Li, "Learning structured sparsity in deep neural networks," in *Advances in neural information processing systems*, 2016, pp. 2074–2082. 1, 2, 3

[15] H. Li, A. Kadav, I. Durdanovic, H. Samet, and H. P. Graf, "Pruning filters for efficient convnets," in *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*, 2017. 1, 2, 3, 6, 7, 9

[16] R. Yu, A. Li, C.-F. Chen, J.-H. Lai, V. I. Morariu, X. Han, M. Gao, C.-Y. Lin, and L. S. Davis, "Nisp: Pruning networks using neuron importance score propagation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 9194–9203. 1, 2, 3, 6, 9

[17] A. Salama, O. Ostapenko, T. Klein, and M. Nabi, "Ieee international conference on acoustics, speech and signal processing (icassp), brighton, united kingdom," in *Advances in neural information processing systems*, 05 2019, pp. 2802–2806. 1, 3, 6, 9

[18] N. Lin, H. Lu, X. Wei, and X. Li, "Headstart: Enforcing optimal inceptions in pruning deep neural networks for efficient inference on gpgpus," in *Proceedings of the 56th Annual Design Automation Conference 2019*. ACM, 2019, p. 23. 1, 2, 3

[19] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014. 1, 6

[20] A. Krizhevsky, G. Hinton *et al.*, "Learning multiple layers of features from tiny images," Citeseer, Tech. Rep., 2009. 1

[21] J.-H. Luo, H. Zhang, H.-Y. Zhou, C.-W. Xie, J. Wu, and W. Lin, "Thinet: pruning cnn filters for a thinner net," *IEEE transactions on pattern analysis and machine intelligence*, 2018. 2

[22] H. Kim, M. U. K. Khan, and C.-M. Kyung, "Efficient neural network compression," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 12 569–12 577. 2, 3

[23] T.-J. Yang, Y.-H. Chen, and V. Sze, "Designing energy-efficient convolutional neural networks using energy-aware pruning," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 5687–5695. 2, 3

[24] Y.-H. Chen, T. Krishna, J. S. Emer, and V. Sze, "Eyeriss: An energy-efficient reconfigurable accelerator for deep convolutional neural networks," *IEEE Journal of Solid-State Circuits*, vol. 52, no. 1, pp. 127–138, 2016. 2

[25] "Superslash," "Available at http://vispro.itu.edu.pk/open-source-lib/", 2020 (accessed July 15, 2020). 2

[26] M. Alwani, H. Chen, M. Ferdman, and P. Milder, "Fused-layer cnn accelerators," in *2016 49th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*. IEEE, 2016, pp. 1–12. 3

[27] J. Yu, Y. Hu, X. Ning, J. Qiu, K. Guo, Y. Wang, and H. Yang, "Instruction driven cross-layer cnn accelerator with winograd transformation on fpga," in *2017 International Conference on Field Programmable Technology (ICFPT)*. IEEE, 2017, pp. 227–230. 3

[28] C.-Y. Chih, S.-S. Wu, J. P. Klopp, and L.-G. Chen, "Accurate and bandwidth efficient architecture for cnn-based full-hd super-resolution," in *2018 IEEE International Symposium on Circuits and Systems (ISCAS)*. IEEE, 2018, pp. 1–5. 3

[29] H.-N. Wu and C.-T. Huang, "Data locality optimization of depthwise separable convolutions for cnn inference accelerators," in *2019 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 2019, pp. 120–125. 3

[30] J. Sim, S. Lee, and L.-S. Kim, "An energy-efficient deep convolutional neural network inference processor with enhanced output stationary dataflow in 65-nm cmos," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 2019. 3

[31] M. Naumov, "Incomplete-lu and cholesky preconditioned iterative methods using cusparse and cublas," *Nvidia white paper*, 2011. 3

[32] 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 *2016 ACM/IEEE 43rd Annual International Symposium on Computer Architecture (ISCA)*. IEEE, 2016, pp. 243–254. 3

[33] H. Hu, R. Peng, Y.-W. Tai, and C.-K. Tang, "Network trimming: A data-driven neuron pruning approach towards efficient deep architectures," *arXiv preprint arXiv:1607.03250*, 2016. 3

[34] 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," *arXiv preprint arXiv:1704.04861*, 2017. 6, 9

[35] F. Chollet *et al.*, "Keras," "Available at https://keras.io", 2015. 7

[36] Z. Liu, H. Mu, X. Zhang, Z. Guo, X. Yang, K.-T. Cheng, and J. Sun, "Metapruning: Meta learning for automatic neural network channel pruning," in *Proceedings of the IEEE International Conference on Computer Vision*, 2019, pp. 3296–3305. 9

[37] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778. 9

[38] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv 1409.1556*, 09 2014. 9

[39] M. A. Hanif, R. V. W. Putra, M. Tanvir, R. Hafiz, S. Rehman, and M. Shafique, "Mpna: a massively-parallel neural array accelerator with dataflow optimization for convolutional neural networks," *arXiv preprint arXiv:1810.12910*, 2018. 10, 11