

NeuLens: Spatial-based Dynamic Acceleration of Convolutional Neural Networks on Edge

Xueyu Hou*

New Jersey Institute of Technology
Newark, NJ, USA
xh29@njit.edu

Yongjie Guan*

New Jersey Institute of Technology
Newark, NJ, USA
yg274@njit.edu

Tao Han

New Jersey Institute of Technology
Newark, NJ, USA
tao.han@njit.edu

ABSTRACT

Convolutional neural networks (CNNs) play an important role in today's mobile and edge computing systems for vision-based tasks like object classification and detection. However, state-of-the-art methods on CNN acceleration are trapped in either limited practical latency speed-up on general computing platforms or latency speed-up with severe accuracy loss. In this paper, we propose a spatial-based dynamic CNN acceleration framework, NeuLens, for mobile and edge platforms. Specially, we design a novel dynamic inference mechanism, assemble region-aware convolution (ARAC) supernet, that peels off redundant operations inside CNN models as many as possible based on spatial redundancy and channel slicing. In ARAC supernet, the CNN inference flow is split into multiple independent *micro*-flows, and the computational cost of each can be autonomously adjusted based on its tiled-input content and application requirements. These *micro*-flows can be loaded into hardware like GPUs as single models. Consequently, its operation reduction can be well translated into latency speed-up and is compatible with hardware-level accelerations. Moreover, the inference accuracy can be well preserved by identifying critical regions on images and processing them in the original resolution with large *micro*-flow. Based on our evaluation, NeuLens outperforms baseline methods by up to 58% latency reduction with the same accuracy and by up to 67.9% accuracy improvement under the same latency/memory constraints.

CCS CONCEPTS

• **Computing methodologies** → *Neural networks*; • **Human-centered computing** → *Ubiquitous and mobile computing*.

KEYWORDS

convolutional neural networks, dynamic inference, edge computing

ACM Reference Format:

Xueyu Hou*, Yongjie Guan*, and Tao Han. 2022. NeuLens: Spatial-based Dynamic Acceleration of Convolutional Neural Networks on Edge. In *The 28th Annual International Conference on Mobile Computing and Networking (ACM MobiCom '22)*, October 17–21, 2022, Sydney, NSW, Australia. ACM, New York, NY, USA, 14 pages. <https://doi.org/10.1145/3495243.3560528>

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

ACM MobiCom '22, October 17–21, 2022, Sydney, NSW, Australia

© 2022 Association for Computing Machinery.

ACM ISBN 978-1-4503-9181-8/22/10...\$15.00

<https://doi.org/10.1145/3495243.3560528>

1 INTRODUCTION

Computer vision related tasks usually require a large number of computational resources [23]. Many studies focus on reducing the computational cost of CNN inference. Some works propose lightweight network architectures like MobileNets [25, 26, 63], CondenseNet [29], ShuffleNets [51, 93], and EfficientNet [71]. Other studies compress existing networks by pruning [42, 44, 49, 50] or quantization [32, 33, 57]. Recent works propose various ways that allow dynamic computational cost adjustment of CNN inference [19, 43] (details in §2.2). Inspired by human's vision where only a limited portion of visual scene is processed by the visual system, recent works dig into the potential of computational cost reduction based on input spatial information by proposing specialized network architectures [83, 88] or by designing computing flows compatible with general CNN architectures [20, 77, 95]. In video streaming and analytics, regions of interest (RoIs) are determined by cross-frame tracking (Edge-Assisted [47] and Elf [92]) or by low-resolution detection (DDS [9]). By RoI-based encoding, the transmission data sizes of offloaded frames are significantly reduced [47].

In this paper, we propose an adaptive framework, NeuLens, for dynamic CNN inference acceleration on mobile and edge devices. First, we design a novel dynamic mechanism, assemble region-aware convolutional (ARAC) supernet (§4), that effectively reduces inference cost with small accuracy loss. An ARAC supernet is a spatial-split network ensemble. It adaptively selects sub-networks with different sizes for split tiles of an image based on their relevance to the final prediction. Furthermore, we design a lightweight online controller, DEMUX (§5), that dynamically tunes per-tile sub-network selection and the supernet's configurations based on service level objectives (SLOs) in real applications. Finally, we comprehensively evaluate ARAC supernet on different mobile/edge platforms and various applications (§7). Based on our evaluation, ARAC supernet achieves up to 67.9% accuracy improvement over state-of-the-art (SOTA) dynamic inference methods under the same latency/memory constraints (§7.2) and up to 1.23× higher accuracy over SOTA model compression techniques with the same inference latency (§7.4). In addition, applying ARAC supernet into continuous object detection systems boosts the performance by up to 7.7× over SOTA techniques [47] (§7.6).

We summarize the contributions of this paper as follows:

Development of a novel CNN acceleration mechanism for mobile/edge computing platforms (§4). By exploiting spatial and depth redundancy on images and in CNNs, we propose an acceleration mechanism, ARAC supernet, that effectively reduces the consumption of computing resources with slight accuracy reduction. Compared to existing acceleration works, ARAC supernet

*These authors contributed equally to this work.

achieves the Pareto optimality on accuracy-latency trade-off. We highlight the following advanced techniques in ARAC supernet:

- Construction of an ARAC supernet that generally applies to CNN architectures (§4.1). By splitting an input image into tiles, the supernet utilizes sub-networks with different compression levels to analyze them. The outputs from the supernet are concatenated and fed into the rest layers in the CNN model to compute the final results. Such structure allows the supernet to reduce spatial and depth redundancy in computation *without* affecting the overall working schemes of the original CNNs.
- Content-aware per-tile adjustment on computational cost in ARAC supernet (§4.4). A compression guiding gate is designed to effectively analyze the content in each tile and assign a sub-network with proper compression level to analyze them in the supernet. A labeling rule is proposed to automate the training set generation for the compression guiding gate.
- Effective conversion from operation redundancy reduction to on-device latency acceleration. In ARAC supernet, the computation flow is split into multiple independent *micro*-flows. Based on the content of its input (a tile), each *micro*-flow adjusts the operation amount (compression level) in analyzing the input independently. As each *micro*-flow is loaded into a device's computing unit (e.g., GPU) like an individual neural network, its operation reduction is directly converted into latency acceleration.

Design of a lightweight SLO-aware controller adaptive to limited computing budgets on mobile/edge devices (§5). We design an online lightweight controller, DEMUX, to tune ARAC supernet based on user's SLOs with neglectable overhead on mobile and edge devices. Given customized options on the parameters of an ARAC supernet, DEMUX adaptively selects the optimal set of parameters and keeps high accuracy within the user's SLOs.

Implementation of ARAC supernet and performance evaluation on different mobile/edge computing platforms and for various vision applications (§6, 7). We comprehensively evaluate the performance of ARAC supernet from several aspects and proves its effectiveness in boosting the overall performance in CNN-related applications on mobile/edge devices. We highlight our evaluation results as follows:

- Outperforms SOTA techniques in dynamic inference and model compression on mobile/edge devices by up to 67.9% (§7.2) and 1.23× (§7.4), respectively.
- Improves overall performance of SOTA continuous object detection systems on edge by up to 7.7× (§7.6).
- Reduces end-to-end latency by almost 50% on a SOTA 3D object detection system for mixed-reality devices (§7.8).

2 BACKGROUND AND MOTIVATION

2.1 Spatial Related Convolution

As demonstrated in [15, 83], there can be a considerable amount of redundant pixels in an image that are irrelevant to accurate recognition. Several works focus on *reducing convolutional operations of redundant pixels*. The majority of these works propose spatial neural architectures. Compact networks are designed for spatial-redundancy based operation reduction [15, 27, 28, 59, 68, 72, 74, 83]. Sequential networks are designed with multi-scale resolutions [11, 55, 87, 88]. CBAM [80] designs an attention module

that can be inserted into CNNs. Other recent works propose spatial-redundancy-based modifications on computing flows that can be generally applied to popular CNN architectures rather than designing new ones. GFNet [77, 30] dynamically processes a sequence of crops on the image until prediction with sufficient confidence. DRNet [95] predicts optimal resolution for each input image with a resolution predictor. SAR [20] designs a dual-branch network architecture with one analyzing low-resolution input features and selecting high-resolution refined areas for the other in each layer. Compared to these studies, our work proposes a novel computing flow, ARAC supernet, to tackle spatial redundancy. By splitting the input images into tiles, we select different sub-networks in supernet based on their contents. ARAC supernet generally applies to popular CNN architectures like [20, 77, 95]. It is important to note that works like SAR [20], CGNet [27, 28] and ASC [68, 72] are *not* fully supported for practical speed-up by deep learning platforms and require special hardware/framework support. In contrast, our work can be effectively implemented on SOTA deep learning platforms and realize latency speed-up.

2.2 Dynamic Inference

We divide SOTA studies on dynamic inference into two types based on whether they are platform and SLO adaptive.

Platform- and SLO-Agnostic: Some works focus on modifying or designing a single network with the dynamic mechanism. Early-exist topology is proposed for CNNs ([3], MSDNet [31], and ZTW [79]). CNMMs [60] and RNP [45] design networks that can be dynamically pruned. Skipnet [76] and BlockDrop [81] skip blocks in ResNet based on inputs. Similarly, ConvNet-AIG [73] determines whether to skip each layer based on estimated relevance, and CoDiNet [75] optimizes layer skipping based on cross-image similarity. LCCL [8] avoids computing zeros in feature maps by predicting their locations. Other works develop network ensembles for dynamic inference. Russian Doll Network [35] constructs a nested network by embedding smaller sub-networks inside larger ones. HNE [61] designs a hierarchical neural ensemble that allows branch number adjustment. Slimmable networks [43, 89, 90] adjust filter numbers in layers for different inputs. CoE [94] pools a collection of networks trained by mutually exclusive subsets in a dataset. MoE [65] and CondConv [86] construct sub-networks with mixture-of-experts and selects a combination of them for each input. Overall, these works focus on training optimization and architecture modifications. Their designs are hand-crafted *without* considerations on adaptability to platforms and SLOs [41].

Platform- and SLO-Adaptive: In contrast, several works focus on optimizing system-level performance on mobile and edge devices under SLOs. Some works design adaptive frameworks for general CNNs. NestDNN [12] dynamically implements resource-accuracy trade-off inside a compact multi-capacity model. ReForm [85] proposes a resource-aware DNN reconfiguration framework based on the ADMM algorithm. DMS [38] controls resource demand of inference by adaptive pruning. PatDNN [53] designs an efficient DNN framework based on kernel-pattern pruning. LegoDNN [19] dynamically scales DNNs by switching retrained descendant blocks in them. Other works develop adaptive framework by addressing features in specific applications like video analytics (FlexDNN [13])

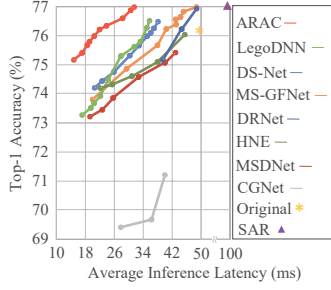


Figure 1: Accuracy-latency trade-off comparison (Base Model: ResNet50, ImageNet [7]), Jetson TX2.

and real-time (video) object detection (AdaVP [48], ApproxDet [84], Remix [37], and [22]).

Compared to these works, our ARAC supernet implements a dynamic mechanism by breaking inputs into tiles and processes them with different compressed sub-networks based on spatial redundancy (§4). Furthermore, we integrate ARAC supernet into mobile and edge systems by designing an SLO-adaptive online controller for on-device inference (§5). In other words, our work also addresses *adaptability* to platforms and SLOs.

2.3 Observations

The accuracy-latency trade-offs of ARAC supernet and SOTA dynamic models are shown in Fig. 1. Both spatial-based works (MS-GFNet [30], DRNet [95], MSDNet [31], CGNet [27, 28], and SAR [20]) and spatial-agnostic works (LegoDNN [19], DS-Net [43], HNE [61]) are included in the comparison. With the same inference latency, the ARAC supernet outperforms SOTA methods by 1.22% to 2.07% in top-1 accuracy. With the same top-1 accuracy, the ARAC supernet takes 19.4% to 47.9% less time per inference. Most SOTA methods on spatial redundancy do not implement practical speed-up [20]. The underlying reason is that they segment away operations on redundant parts in a way that is *not* compatible with the intermediate mapping between layers. For example, SAR [20] (*i.e.*, SOTA pixel-level dynamic network) selects groups of refined patterns from input features on each layer and runs operations on them only. These patterns are irregular, and various patterns are generated per layer, making it impossible for computing efficiency on GPUs [20]. In contrast, ARAC supernet jointly takes advantage of spatially related operation reduction and dynamic inference. By constructing multiple sub-networks with different sizes in the supernet and splitting the CNN inference flow into spatial dimensions, the ARAC supernet can adaptively adjust the computation of each split-flow by selecting the proper sub-network inside the supernet. Each split-flow executes as an independent *micro* network without inter-dependency, *i.e.*, the data flow from layer to layer occurs inside each *micro* network only. Thus, each *micro* network can be loaded to the GPU as one CNN model, and their operation reduction can be effectively translated into latency speed-up. Note that, though GFNet [30, 77] designs spatial-based CNN workflow where its operation reduction leads to real speed-up, its speed-up is limited due to the sequential executions on a series of cropped images; similarly for MSDNet [31].

We demonstrate the performance of two basic CNN acceleration techniques in Fig. 2, where SOTA studies that apply to practical

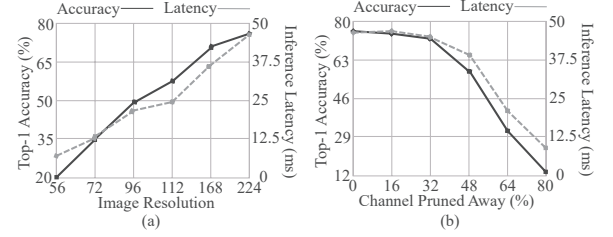


Figure 2: Performance of input resize and channel pruning (Base Model: ResNet50, ImageNet [7]), Jetson TX2.

speed-up are built upon either of them. The first acceleration technique is to resize input image to different scales (Fig. 2 (a)). As more detailed information is lost with down-scaling, the accuracy falls below 50% with resolution 96. Even SOTA work on dynamic resolution, DRNet [95], keeps resolution options above 96. Consequently, adjusting resolution alone cannot generate a wide range of latency while preserving accuracy. The second acceleration technique is to prune away channels in layers (Fig. 2 (b)), adopted by LegoDNN [19] and DS-Net [43]. Note that channel pruning is different from weight pruning; the latter requires special hardware support for acceleration. As shown in Fig. 2 (b), though the accuracy does not drop sharply with a few channels pruned away, there is no significant latency reduction either. When more channels are pruned away, latency greatly decreases, but accuracy also shows an obvious drop.

2.4 Challenges

One way to take benefits of both input downscale and channel pruning is shown in Fig. 3. We prepare four sub-networks with different sizes (compressed on the first three blocks in ResNet-50 with four channel pruning levels). We split the original input image into 3×3 tiles (78×78 pixels). We pair each tile with a sub-network size and the tile is taken as input to the sub-network of the size. All the outputs of the 3×3 tiles are concatenated and fed into the last block in ResNet-50 to generate the final results of the whole image. We exhaustively search over all the possible pairs for the four images in this *preliminary* study, and we mark the optimal pairs¹ of tiles and model sizes for each image in Fig. 3. As shown in Fig. 3, the number of operations and latency are significantly reduced without affecting prediction correctness. The success of such tile-split CNN flow comes from two facts: *First*, there is no resolution resize of the original image. While reducing sizes by several times, the resolution-kept split-tile preserves detailed information. *Second*, the method follows the intuition of human vision flow where the critical parts are analyzed with high intensity and less critical parts are analyzed with low intensity. The sub-networks of all the tiles of an image are loaded independently as tiny models, and their outputs are integrated to generate the image's final predictions. Sub-networks implement the variation of required computations for the tiles with different sizes.

However, though the above preliminary study demonstrates the benefits of tiling image input, there lie multiple challenges to implementing such a beneficial inference flow: *First, how to design and prepare efficient and effective sub-networks with different sizes?*

¹The *optimal* pairs refer to the pairs that generate correct labels with the smallest number of operations.

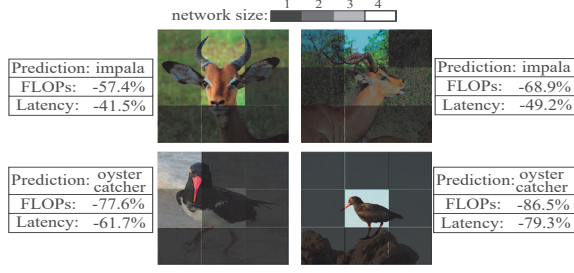


Figure 3: Observations on tiling image input.

We design a compact sub-network ensemble with channel-slicing, called *ARAC supernet* (§4.1) where a smaller sub-network can be directly extracted from the largest using channel-slicing. We jointly train all the sub-networks together with an ensemble bootstrapping scheme (§4.3). *Second, it is impossible to exhaustively search over all the possible pairs to find the optimal one in reality.* We present a lightweight guiding gate to autonomously select sub-networks for tiles (§4.4). *Third, how to apply the proposed inference flow to real applications, especially satisfying service level objectives?* We propose a lightweight online controller for on-device inference (§5). The controller tunes sub-network selection given latency and memory constraints while preserving accuracy.

3 SYSTEM OVERVIEW

The proposed SLO-adaptive deep learning acceleration framework for mobile and edge computing platforms, NeuLens, is shown in Fig. 4. NeuLens contains a one-time offline stage and a lightweight online stage. In the offline stage, a CNN model is modified into an ARAC supernet (supernet generation in Fig 4); a compression guiding gate is designed to select a sub-network for each tile of an image (CGG design in Fig. 4); memory predictor, latency predictor, and accuracy comparator are trained based on profiling data of supernets (profiling in Fig. 4). In the online stage, a lightweight online controller, DEMUX, is designed to find optimal parameters, *i.e.*, tile size, layer number, and per-tile compression levels for ARAC supernet. DEMUX adaptively selects these parameters based on the contents of the tiles of an input image and service-level objectives (SLOs). Based on input images' contents and applications' SLOs, NeuLens splits an CNN inference computing flow into multiple micro-flows and independently controls each micro-flow's computational cost. In this way, NeuLens successfully amplify the benefits of SOTA CNN acceleration techniques, *i.e.*, spatial-redundancy-based computing-operation reduction and channel pruning. With the proposed ARAC supernet technique, NeuLens is able to achieve up to 67.9% accuracy improvement over state-of-the-art (SOTA) dynamic inference methods under the same latency/memory constraints (§7.2) and up to 1.23× higher accuracy over SOTA model compression techniques with the same inference latency (§7.4).

4 DESIGN OF ARAC SUPERNET

4.1 Workflow of ARAC Supernet

As shown in Fig. 5, the input image is split into $k \times k$ tiles. These tiles are fed into ARAC supernet, and they are processed *independently* in parallel. Based on their contents, they are processed by different sub-networks in the supernet. The outputs of these tiles from the supernet are concatenated and are further fed into the

following layers (the layers in gray in Fig. 5) in the model. There can be multiple ways to form such a model with ARAC supernet (*e.g.*, neural architecture search). In this paper, we modify existing CNN models' architectures (*e.g.*, MobileNet [26], ResNet-50 [21], Inception [69]) into architectures with ARAC supernets.

Given an existing CNN model with N layers, we modify the first P layers ($0 \leq P \leq N$) into the ARAC supernet and keep the rest $(N - P)$ layers unchanged. In the supernet, we compress the first $(P - 1)$ layers with different compression levels using techniques like channel pruning [42, 44, 49, 50] in this paper and we set R different compression levels $C = \{c_1, \dots, c_R\}$ ($0 \leq c_1 < \dots < c_R \leq 1$), which represents different ratios of output channels pruned away in the first $(P - 1)$ layers. Thus, given a compression level c_i , the number of output channels in each layer of the first $(P - 1)$ layers is:

$$d_l^o = (1 - c_i) \cdot D_l^o, \quad \forall 1 \leq l \leq (P - 1). \quad (1)$$

where D_l^o is the original number of output channels, and d_l^o is the channel number after compression. When $c_i = 0$, we have $d_l^o = D_l^o$, which means that there is no output channel pruned away; when $c_i = 1$, we have $d_l^o = 0$, which means that all output channels are pruned away (*i.e.*, the whole $(P - 1)$ layers are pruned away). A high compression level represents that a large number of output channels in the first $(P - 1)$ layers are pruned, and vice versa. In other words, *A sub-network with a higher compression level has a lower computational cost.* Based on the content in each tile and the latency requirements (details in §5), the system selects different compression levels for them. *Note* that though an ARAC supernet splits an image into tiles, the tiling will not hurt feature extraction across multiple tiles in the original image. For example, given an object's key features across tiles, the supernet will process all the corresponding tiles with relatively high computation. Following the supernet, the extracted features from different tiles are fed into the shared last layers, and the final results are generated based on the information from all tiles (the original image). Furthermore, as the construction of ARAC supernet does not break down the outputs of the original model, ARAC supernet generally applies to various CNN-based vision tasks (§7.7).

4.2 Intermediate Dimensions in Supernet

For a sub-network in ARAC supernet, the dimensions of input to the $(l + 1)$ -th layer are equal to those of output from the l -th layer. The depth of output (*i.e.*, the number of output channels) from a layer is determined by the compression level as shown in Eq. 1. For a layer l , the relationships between the output's and input's spatial dimensions (height and width) are determined by the layer's configurations [24]:

$$w_l^{in} = (w_l^o - 1) \cdot S_l - \mathbf{b} \cdot A_l + F_l, \quad (2)$$

$$h_l^{in} = (h_l^o - 1) \cdot S_l - \mathbf{b} \cdot A_l + F_l. \quad (3)$$

where \mathbf{b} is a binary: If the tile is on the edge of the original image and layer l is a convolutional layer then $\mathbf{b} = 1$, else $\mathbf{b} = 0$; S_l is stride; F_l is filter size; A_l is padding. As we split the input image into $k \times k$ tiles and process them *independently* with different sub-networks inside the supernet, the $k \times k$ outputs from the supernet need to be concatenated together. To seamlessly concatenate them, the spatial dimensions of each output are set to $(W_p^o/k, H_p^o/k)$, where (W_p^o, H_p^o) are the spatial dimensions of input to the $(P + 1)$ -th layer.

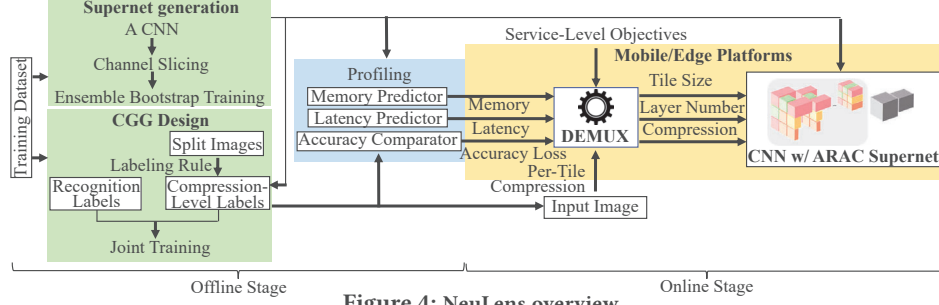


Figure 4: NeuLens overview.

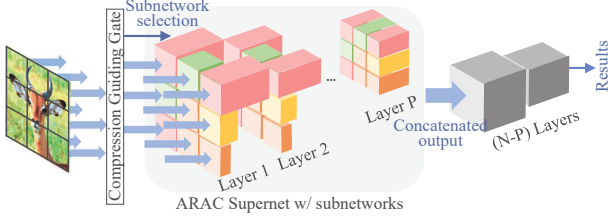


Figure 5: Workflow of ARAC supernet.

Given $(W_p^o/k, H_p^o/k)$, the intermediate spatial dimensions of inputs and outputs in sub-networks of supernet can be determined by Eq. 2. Consequently, we can obtain the spatial dimensions of each split tile, denoted as (w^{in}, h^{in}) . Further, we can determine the coordinates of each tile on the original input image: $x_{1,:} = 0, x_{2,:} = w_{in} - \Delta x_{in}, x_{i,:} = x_{i-1,:} + w_{in} - \Delta x_{in} (3 \leq i \leq k); y_{:,1} = 0, y_{:,2} = h_{in} - \Delta y_{in}, y_{:,j} = y_{:,j-1} + h_{in} - \Delta y_{in} (3 \leq j \leq k)$; where $x_{i,:}$ refers to the x -coordinate of the tile on the i -th row, $y_{:,j}$ refers to the y -coordinate of the tile on the j -th column, $\Delta x_{in} = (k \cdot w_{in} - W_{in})/(k-1)$, and $\Delta y_{in} = (k \cdot h_{in} - H_{in})/(k-1)$. Note that the origin is on the top-left corner of the original input data.

In ARAC supernet, each tile is processed independently by a compression level. We denote the compression level for the tile at position (i, j) ($1 \leq i, j \leq k$) as $c_{i,j}$. For convolutional layers in supernet, the number of operations (multiply-accumulations) is $O_{l \in conv}(c_{i,j}) = (1 - c_{i,j})^2 \cdot F_l^2 \cdot D_{l-1}^o \cdot w_l^o \cdot h_l^o \cdot D_l^o$. For maxpooling layers in supernet, the number of operations (comparisons) is $O_{l \in maxpool}(c_{i,j}) = (1 - c_{i,j}) \cdot F_l^2 \cdot D_{l-1}^o \cdot w_l^o \cdot h_l^o$. Thus, the total number of operations of supernet with $\{c_{i,j}\}$ can be obtained by: $\sum_{j=1}^k \sum_{i=1}^k \sum_{l=1}^P O_l(c_{i,j})$.

4.3 Training of Supernet

For a CNN model with N layers, we offline train R compressed models with compression levels c_i ($1 \leq i \leq R$). A straightforward way is to train the R compressed models individually. However, the models obtained in the such way do not share weights and consume a large amount of memory on edge devices [90, 89, 43]. Thus, we form an ensemble of R models by channel slicing. We denote the weights of layers in the model with compression level c_1 as \mathbf{W}_l ($1 \leq l \leq N$); For the other models with compression levels c_i ($2 \leq i \leq R$) in the ensemble, their weights in layer l are $\mathbf{W}_l[(1 - c_i)/(1 - c_1) \cdot D_l^o]$, $1 \leq l \leq N$. In this way, the number of weights in an ensemble of R models is equal to that in the model with compression level c_1 , and the other models can be directly obtained by slicing its weights on the channel dimension. To train the ensemble of R (compressed) models, we utilize an ensemble bootstrapping scheme similar to IEB [43]. The difference between ours and IEB is that:

Instead of training randomly selected models, we train the models with compression levels ranging from c_2 to c_{R-1} to predict the soft label generated by the model with compression level c_1 , i.e., $\mathcal{L}(y_{c_i}, y_{c_1})$ ($2 \leq i \leq R-1$). The model with compression level c_1 is trained to predict the ground-truth label, i.e., $\mathcal{L}(y_{c_1}, y_{gt})$. The model with compression level c_R is trained to predict the probability accumulation of all the other models, i.e., $\frac{1}{R-1} \sum_{i=1}^{R-1} y_{c_i}$. Similar to training a slimmable neural network [43], we train the ensemble of R models together in each training iteration: We compute the losses defined above individually for all models and accumulate their back-propagation gradients together; Then, we update weights in the ensemble. Note that the inputs to these models during training are images in the training dataset without being split. Given the trained ensemble of R models, a supernet with P layers can be obtained by taking each model's first P layers in the ensemble. The following $(N - P)$ layers are the same as those in the original CNN model.

4.4 Compression Guiding Gate

We design a compression guiding gate (CGG) at the entrance of an ARAC supernet. A CGG selects the compression level for each split image. Specifically, the CGG takes split images as inputs and generates compression levels for them. A CGG can be regarded as a classification model that classifies split images into different compression levels. However, the images in the training dataset are labeled for the original applications, and we do not have *compression-level labels* for split images. Thus, we propose a *Labeling Rule* to generate *compression-level labels* for the split images in the training dataset. With the generated *compression-level labels*, we can train a CGG with supervised learning techniques.

4.4.1 Labeling Rule: We refer to the original labels of images as *A-labels* and the *compression-level labels* for split-images as *C-labels*. Given compression levels $\{c_1, \dots, c_R\}$ ($c_1 < \dots < c_R$), we label all split-images with *C-labels* in the training dataset. For the split-images of an image, the *labeling rule* is: Initially, all the split-images are processed by the sub-network with compression level c_1 . If its final output does *not* match the *A-label*, then all the split-images are labeled as c_1 ; else we raise the compression level of each split-image to c_2 respectively. If one's final output does *not* match the *A-label*, then the corresponding split-image is labeled as c_1 ; else we raise the compression level of it to c_3 . We repeat the *match-label-raise* procedure until we reach compression level c_R . Fig. 6 shows an example of a labeling process given four compression levels.

4.4.2 CGG Architecture: Given an input image X , CGG generates $k \times k$ one-hot R -length vectors $\{\mathbf{v}\}_{k \times k}$. An element in the vector

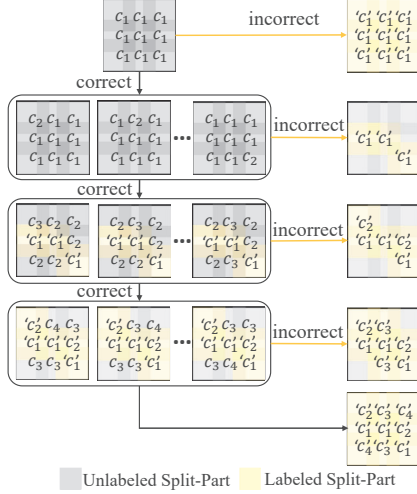


Figure 6: An example of Labeling Rule.

represents a compression level c_i ($1 \leq i \leq R$). The final outputs of CGG is $\{c_{\arg \max v}\}_{k \times k}$. The CGG processes the $k \times k$ split-images of X in a batch. Denoting a split-image as x , we have an encoder \mathcal{E} and a feature-mapping function $\mathcal{F}: v = \mathcal{F}(\mathcal{E}(x))$. For encoder \mathcal{E} , we utilize one convolutional layer and one maxpooling layer to integrate spatial information. we utilize a fully connected layer for feature-mapping function \mathcal{F} . The compression level choice is obtained by applying $\arg \max$ to the output vector of \mathcal{F} .

4.4.3 CGG Training: We label the split-images in the training dataset by our *Labeling Rule*. We define the compression-level loss function as: $\mathcal{L}_{cl} = \mathcal{L}(CGG(x), c_{gt}(x))$, where x is an split-image, $CGG(x)$ is softmax output of x , $c_{gt}(x)$ is one-hot encoding of *compression-level label* of x . The image-recognition loss is $\mathcal{L}_{ir} = \mathcal{L}(y(X), y_{gt})$, where $y(X)$ is softmax output from the whole model, y_{gt} is one-hot encoding of ground-truth label. We optimize CGG jointly by image-recognition loss and compression-level loss:

$$\mathcal{L}_{CGG}(X) = (1 - \alpha) \mathcal{L}_{ir} + \alpha \frac{\sum_{x \in X} \mathcal{L}_{cl}}{k^2} \quad (4)$$

where α controls the effect of the two losses. The back-propagations of $\mathcal{L}(y(X), y_{gt})$ to CGG's parameters are implemented by gumbel-softmax technique [34]. With the loss function (Eq. 4), the CGG is trained to analyze the content of a split-image and selects a compression level to it. The training of the CGG ensures the correct output of the whole model (the first part in Eq. 4) while matching the compression-level label (the second part in Eq. 4) as well. With a higher α , the CGG tends to assign tiles with higher compression levels for more significant computational cost reduction; with a lower α , the CGG tends to assign tiles with lower compression levels for higher accuracy. We set $\alpha = 0.7$ in this paper.

5 CNN INFERENCE W/ SUPERNET

A compression guiding gate (CGG) selects proper compression levels in an ARAC supernet for tiles of an image. The selected compression levels from CGG guarantee neglectable accuracy loss with the smallest number of operations. However, CGG is SLO-agnostic, *i.e.*, it selects compression levels without consideration of latency and memory constraints. Thus, we design another SLO-adaptive component, compression-level gear (§5.2), to tune the selected compression levels online to meet SLOs. Given candidate

controllable parameters (P, k, C), a lightweight online controller, DEMUX (§5.3), is designed to find the optimal set of controllable parameters.

5.1 Problem Definition

As demonstrated in §4, an ARAC supernet contains the first P layers of a CNN model with R compression levels; an input image is split into $k \times k$ tiles and is processed by sub-networks in the supernet independently. Thus, for CNN inference with ARAC supernet, we can dynamically control: (1) the number of layers in the supernet, $P \in \mathcal{P}$; (2) the number of tiles, $k \in \mathcal{K}$; (3) the compression levels for the tiles, $c_{i,j} \in C$, ($1 \leq i, j \leq k$). We denote compression-level matrix for $k \times k$ tiles as $\Omega = \{c_{i,j}\}_{1 \leq i,j \leq k}$. \mathcal{P} , \mathcal{K} , and C are candidate values for P , k , and $c_{i,j}$. In general, optimizations of CNN inference focus on three types of performance: (1) accuracy (Acc), (2) latency (T), and (3) memory consumption (M). In this paper, we focus on maximizing accuracy given latency and memory constraints (\bar{T} and \bar{M}):

$$\arg \max_{P^*, k^*, \Omega^*} Acc(P, k, \Omega), \quad (5)$$

$$s.t. T(P, k, \Omega) \leq \bar{T}, \quad (6)$$

$$M(P, k, \Omega) \leq \bar{M}. \quad (7)$$

As higher compression levels would reduce accuracy (Eq. 5) but get lower latency/memory consumption (Eq. 6 and 7), the trade-off exists in tuning compression level because high compression level leads to low accuracy though latency and memory constraints are satisfied. For the tiles of an input image, the ARAC supernet utilizes different compression levels to process them. In other words, given C compression levels and $k \times k$ tiles of an input image, there are $C^{k \times k}$ combinations of compression level choices for all the tiles of the image, which can be an extremely large number (*e.g.*, 1, 953, 125 with $C = 5$ and $k = 3$). Thus, an efficient solution to Eq. 5 to 7 is necessary rather than exhaustive search.

5.2 Compression-Level Gear

For $k \times k$ tiles of an image, CGG (§4.4) selects compression levels for them. We denote the compression levels of the $k \times k$ tiles selected by CGG as $\Omega_0 = \{c_{i,j}^{(0)}\}_{1 \leq i,j \leq k}$, $c_{i,j}^{(0)} \in C$. The inference latency $T(P, k, \Omega_0)$ and memory consumption $M(P, k, \Omega_0)$ are estimated by latency and memory consumption predictors. When the latency/memory constraints (Eq. 6 and 7) are violated, we adjust compression levels of the $k \times k$ tiles with a compression-level gear (CLG) by adaptively raising the compression levels of the $k \times k$ tiles. There are different ways to adjust the compression levels. We describe the way that we empirically find effective (Fig. 12 (a) in §7.3), which is referred to as *confidence-based stepping* (CS) method. As CGG is trained in a supervised way (§4.4), its confidence indicates the probability of being correctly identified with the compression level. Thus, we design CLG based on the confidences to Ω_0 . Specifically, we set a confidence threshold θ_f and a window length Δw . The tiles with confidences lower than θ_f are sorted by confidences from low to high, and the Δw tiles are cyclically selected among them. The CLG runs in loops until latency and memory constraints (Eq. 6 and 7) are satisfied. In each loop, the CLG raises compression levels of Δw tiles by one.

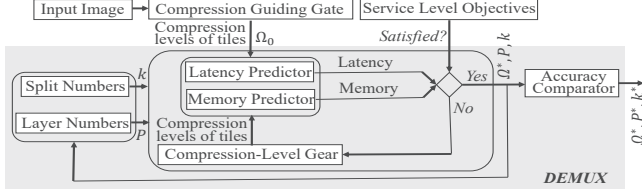
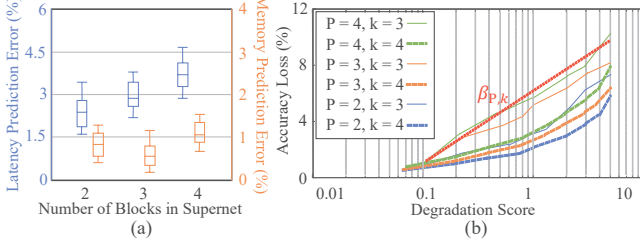


Figure 7: DEMUX.

Figure 8: Predictor observations: (a) prediction errors of latency and memory ($k \in \{3, 4\}$, $R = 4$), (b) Accuracy Loss v.s. Degradation Score. (Base Model: ResNet50, ImageNet [7])

5.3 DEMUX

We design DEMUX to find (P^*, k^*, Ω^*) of Eq. 5 to 7. DEMUX is consisted of five lightweight components: CGG (§4.4), CLG (§5.2), latency predictor, memory predictor, and accuracy comparator. The workflow of DEMUX is shown in Fig. 7. Given $P \in \mathcal{P}$ and $k \in \mathcal{K}$, CGG and CLG find the matrix of compression-levels $\Omega_{P,k}$ that satisfies Eq. 6 and 7 (§5.2). After finding all the matrices of compression-levels $\{\Omega_{P,k}\}_{P \in \mathcal{P}, k \in \mathcal{K}}$, the accuracy comparator selects (P^*, k^*, Ω^*) from $\{P, k, \Omega_{P,k}\}_{P \in \mathcal{P}, k \in \mathcal{K}}$. We profile the computing latency and memory consumption with respect to the compression levels of sub-networks in supernet. We train lightweight linear regression models to predict latency and memory for a supernet as inputs. *Note* that the training of the linear regressors are one-time work and generate small offline overhead, *e.g.*, training time is less than five minutes on a computer with CPU only (Intel i7-8700K). For the rest layers following the supernet, we can simply record their computing latency and memory consumption with respect to the number of layers. The total latency/memory consumption is obtained by combining the two parts. As shown in Fig. 8 (a), the predicted latency and memory consumption match the measured results with less than 5% error. To select the optimal set among the sets of (P, k, Ω) from CLG, We define *degradation-score* as the averaged difference across $k \times k$ tiles between the compression levels selected by CGG and the compression levels adjusted by CLG. As shown in Fig. 8 (b), there is a strong correlation between accuracy losses and *degradation-scores*. Thus, we can offline profile the curves for each pair of (P, k) and regress a coefficient $\beta_{P,k}$. The accuracy loss for each pair of (P, k) can be estimated by $\beta_{P,k}$ and we select (P^*, k^*, Ω^*) that has the lowest accuracy loss. *Note* that, for all pairs of (P, k) , we observe neglectable accuracy loss when processing tiles with the compression levels selected by CGG.

6 IMPLEMENTATION

The implementation of ARAC supernet is as follows:

Testbeds: For *on-device inference* evaluations, we choose three heterogeneous mobile/edge devices: Jetson TX2; Xiaomi 6 Plus;

Alienware 17 R3. The first device runs Linux Ubuntu 18.04 LTS; The second device runs Android 10.0; The third device runs Windows10. **Base CNN models, datasets, and framework:** We mainly evaluate on two most important applications on mobile and edge systems: (1) Image Classification: We build ARAC supernet based on three popular CNN models (ResNet50 [21], MobileNetV3 [25], and Inception-V3 [69]) and we use ImageNet dataset [7]; (2) Object Detection: We build ARAC supernet based on the most commonly used model (YOLOv3 [58]) and we use COCO dataset [46]. We use Pytorch framework [56]. We also evaluate on nine other vision applications in §7.7.

Baselines: We select three SOTA methods that outperform the others in Fig. 1: (i) DS-Net [43] adjusts filter numbers in layers by channel slicing for different inputs. The key difference between the proposed ARAC supernet and DS-Net is that we split an image into small tiles and process them respectively with different compressed sub-networks, but DS-Net processes an image as a whole; (ii) MS-GFNet [30] dynamically processes a sequence of crops on the image until prediction with sufficient confidence. Though MS-GFNet designs spatial-based CNN workflow where its operation reduction leads to real speed-up, its speed-up is limited due to the sequential executions; (iii) LegoDNN [19] dynamically scales DNNs by switching retrained descendant blocks in them. It offline generates sets of blocks by filter pruning. It takes an image as a whole and processes it with adaptively selected block scales. *Note* that DS-Net and LegoDNN are the SOTA methods for dynamic inference *without* spatial-redundancy based acceleration; MS-GFNet is the SOTA method *with* spatial-redundancy based acceleration. We also compare ARAC supernet with SOTA model compression techniques in §7.4.

Candidate controllable parameters: We set three tile sizes (k) by splitting images into 2×2 , 3×3 , and 4×4 tiles. We set five compression levels ($R = 5$) with compression ratios $\{0, 0.25, 0.5, 0.75, 1\}$. *Note* that compression ratio represents the ratio of channels pruned away, *e.g.*, all channels are pruned away with a compression ratio equaling to 1. For ResNet50, we set three layer number options in supernet: two blocks (11 layers), three blocks (23 layers), and four blocks (41 layers). For MobileNet-V3, we set two-layer number options in supernet: eight layers and 13 layers. For Inception-V3, we set three layer number options in supernet: seven layers, after the first Inception module and after the second Inception module. For YOLO-V3, we set three layer number options in supernet: after the first, the second, and the third residual block.

Hyper-parameters in CLG: We study the effects of the two hyper-parameters in CLG (§5.2). We empirically find the optimal settings are: $\Delta w = 2$ for 2×2 tile split, $\Delta w = 4$ for 3×3 tile split, $\Delta w = 5$ for 4×4 tile split; and $\theta_f = 0.5$.

Compression guiding gate: The convolutional layer of the encoder in CGG (§4.4) is Conv2d(3, 64, kernel=(7, 7), stride=(4, 4)). The input tile is resized to 28×28 resolution. We train one CGG general to all tile sizes and layer numbers in supernet. The compression-level prediction accuracy (ground-truth labels generated by the labeling rule in §4.4.1) is 82.6% for ResNet50, 79.5% for MobileNet-V3, 85.7% for Inception-V3, 75.8% for YOLO-V3.

Offline training: We train supernet (§4.3) on two NVIDIA RTX A6000 GPUs. We adopt the fast training approach in [19] to train

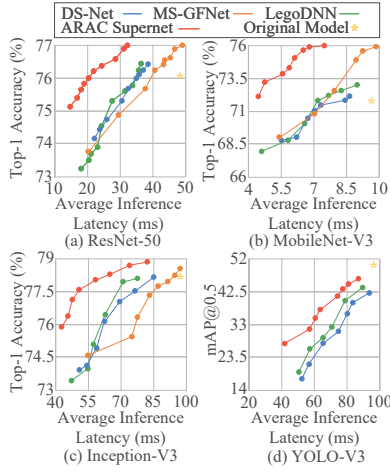


Figure 9: Accuracy v.s. Latency (Jetson TX2).

the sub-networks in the ARAC supernet. For image classification, it takes 17h for ResNet50 based supernet, 13.4h for MobileNet-V3 based supernet, and 23.6h for Inception-V3 based supernet. For object detection, it takes 37.5h for YOLO-V3. For all the supernets, the pre-trained weights in their base models are used as the initial weights in training. We train the *compression guiding gate* (§4.4) on the same platform and it takes 10.5h to train.

Online overhead: We measure the overhead of CGG (§4.4) and DEMUX (§5.3). On Jetson TX2 and Alienware 17 R3, the overhead of CGG and DEMUX is 0.88ms to 1.87ms. On Xiaomi 6 plus, the overhead of CGG and DEMUX is 4.7ms to 7.2ms.

7 EVALUATION

7.1 Accuracy v.s. Latency

The accuracy v.s. latency curves with different base models are shown in Fig. 9. The latency is measured on Jetson TX2 and averaged over 200 times inferences. With the same latency, ARAC supernet outperforms SOTA methods by 0.46% to 76.5% accuracy improvement. With the same accuracy, ARAC supernet outperforms SOTA methods by 2.5% to 58% latency reduction. Compared to ARAC supernet, MS-GFNet [30] shows relatively sharper drop in accuracy when low latency. It is mainly due to the cut-off of its focus sequence, and MS-GFNet mainly relies on the low-resolution glance stage for predictions with low latency. In addition, due to its sequential exiting scheme, GFNet can only be applied to image classification application [77, 30]. The sequential scheme of GFNet cannot tackle with detection of multiple objects in the same image. Thus, we only compare ARAC supernet with LegoDNN and DS-Net in object detection applications. Compared to ARAC supernet, DS-Net [43] and LegoDNN [19] can only adjust computational cost at image-level. The regional awareness of ARAC supernet brings up to 46.2% latency reduction with the same accuracy compared to DS-Net and LegoDNN.

7.2 Latency/Memory Constraints

We evaluate the accuracy of the ARAC supernet and the baseline methods under various latency and memory constraints, as shown in Fig. 10. We implement on three platforms: Jetson TX2 (Fig. 10 (a) to (d)), Xiaomi 6 Plus (Fig. 10 (e) to (h)), Alienware R17 (Fig. 10 (i) to (l)). Under the same latency and memory constraint, ARAC

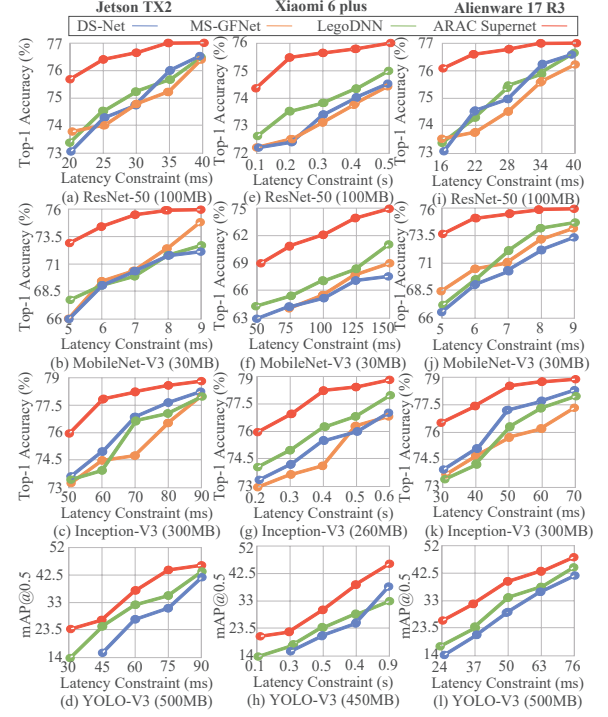


Figure 10: Accuracy under latency/memory constraints.

supernet outperforms the baseline methods by 0.06% to 67.9% accuracy improvement with the four base CNN models across the three platforms. Especially, ARAC supernet achieves high accuracy improvement over the baseline methods when latency constraints are stringent, by 2.2% to 67.9%. Due to the sequential execution of MS-GFNet [30], its performance becomes similar to the spatial-agnostic methods (DS-Net [43] and LegoDNN [19]). The *success* of ARAC supernet in preserving high accuracy under stringent constraints is because it can wisely select regions that are crucial to correct predictions and these regions can be analyzed with high resolution even under stringent conditions.

7.3 Ablation Study

We have the following controllable parameters in the ARAC supernet: (1) candidate layer numbers in the supernet; (2) candidate tile sizes in splitting an image; (3) candidate compression levels of sub-networks. Their default settings are described in §6. We elaborate on the principle of how to set these parameters here. The selection of these parameters is determined by two rules: *First*, different value of the parameter generates a significant change in performance. *Second*, the overhead of DEMUX with the given options is trivial. For example, when selecting candidate layer numbers in the supernet for ResNet-50, we vary the layer number options from one to five, as shown in Fig. 11. Though increasing the layer number options above three does not show a significant performance change, it generates around twice the overhead for DEMUX. Thus, we set three layer number options for ResNet-50. The settings of tile sizes and compression levels are determined similarly. Overall, given a new model, a similar one-time procedure can be applied to find a group of settings for an ARAC supernet.

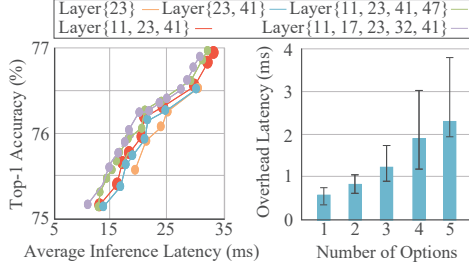


Figure 11: Accuracy v.s. Inference Latency under different layer number options in ARAC supernet. (Base Model: ResNet50, ImageNet [7]), Jetson TX2.

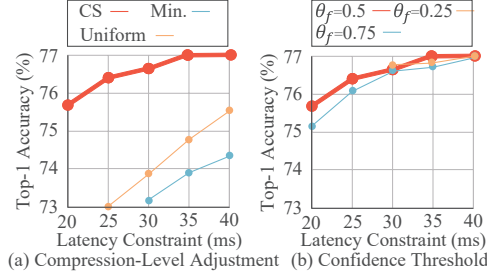


Figure 12: Accuracy v.s. Inference Latency under different hyper-parameters in Compression-Level Gate. (Base Model: ResNet50, ImageNet [7]), Jetson TX2.

We study the effect of the starting layer of the ARAC supernet. We observe that starting the supernet from an intermediate layer demonstrates poorer performance (e.g., 0.2% to 2.7% accuracy reduction under the same inference latency for a ResNet-50 based ARAC supernet). The reason is that the spatial dimensions of the feature maps in a CNN model decrease with the increase of layers. For example, the input’s spatial dimensions to ResNet-50 are 224×224 , the input feature’s spatial dimensions to the 12-th layer in ResNet-50 are only 14×14 . Such a low-dimension indicates a lack of spatial redundancy on the intermediate features. Thus, starting the ARAC supernet from the intermediate layer (especially from a very deep layer in a model) lowers its performance compared to starting from the input layer.

We observe the performance of ARAC supernet with different settings of CLG in Fig. 12. In Fig. 12 (a), we compare different compression-level adjustment methods regarding the performance under different latency constraints (memory constraint= 100MB). The CS method (details in §5.2) is compared to two other potential methods, i.e., uniform (increasing the compression levels of all the tiles together) and min (increasing the compression level of the tile with the lowest compression level). Overall, the CS method outperforms them by up to 2.0% to 4.5%, respectively. In Fig. 12 (b), we observe the effect of the hyper-parameter, confidence threshold (θ_f), on the performance of ARAC supernet. While more tiles are selected to be tuned by the CLG with a higher confidence threshold, it is risky to tune more than needed. Under the same latency constraint, the accuracy with $\theta_f = 0.75$ is up to 0.67% lower than that with $\theta_f = 0.5$; the latency constraints under 30ms cannot be satisfied with $\theta_f = 0.25$ due to lack of tunable tiles when constraints are violated.

It is important to note that, though these hyper-parameters have an effect on the performance of ARAC supernet, their effects are trivial except for the compression-level adjustment methods in Fig. 12 (a). On the one hand, it justifies the rationality of the design of the ARAC supernet in §4 and the CLG in §5.2. On the other hand, it demonstrates the robustness of the ARAC supernet against the hyper-parameter settings, which reduces the potential engineering work in applying ARAC supernet in practice.

7.4 ARAC and Model Compression Techniques

As a dynamic inference technique, we have shown that ARAC supernet outperforms SOTA works in §7.1. We further illustrate the relationship between ARAC supernet and existing model compression techniques. Unlike dynamic inference, model compression techniques target generating one compressed model with the lowest computational cost and the highest accuracy [39, 1, 40]. We specifically compare ARAC supernet with SOTA model compression techniques that apply to latency acceleration on general deep learning platforms. We demonstrate that: (1) ARAC supernet, by exploiting spatial redundancy on images, outperforms SOTA model compression techniques; (2) ARAC supernet is complementary to model compression techniques and can collaboratively boost the performance with them.

Comparison: We compare ARAC supernet with SOTA model compression techniques that keep high accuracy with effective latency reduction. While each type of technique includes a series of works (e.g., over 20 methods in channel pruning), we select the one that shows the Pareto optimality on accuracy-latency trade-off. Specifically, for channel pruning, we select PruneNet [39]; for early-exit, we select ZTW [79]; for low-rank decomposition, we select [40]. The comparison is shown in Fig. 13, where the performance of the original model is also marked. Overall, benefit from the spatial-redundancy reduction, ARAC supernet achieves the highest accuracy with the same latency, i.e., by 1.1% to over 6.9% higher accuracy (ResNet-50) and by 8.8% to 1.23× higher accuracy (YOLO-V3). Correspondingly, it achieves the lowest latency with the same accuracy, i.e., by 60.2% (ResNet-50) and by 3.4% to 18.4% (YOLO-V3) latency reduction compared to channel pruning. Besides the performance improvement, the comparison in Fig. 13 also shows another advantage over the model compression techniques, higher granularity in tuning accuracy-latency tradeoff. As an ARAC supernet splits the original input image into small tiles and prepares different levels of compressed sub-networks to process them, the accuracy v.s. latency curve generated by ARAC supernet demonstrates higher granularity than model compression techniques. For example, in Fig. 13 (a), in the latency range of 18ms to 34ms, there are only two points on the curve generated by channel pruning, but there are eight points on the curve generated by ARAC supernet. The higher granularity of ARAC supernet largely improves its adaptivity to dynamic conditions like changing contention on devices, i.e., the performance of inference can be adaptively tuned by ARAC supernet according to the device conditions (details in §7.5).

Integration: In Fig. 14 (a), we show the performance of ARAC supernet that integrates with low-rank decomposition [40]. The integration follows the same procedure described in §4 besides that, the sub-network with different compression levels is formed by

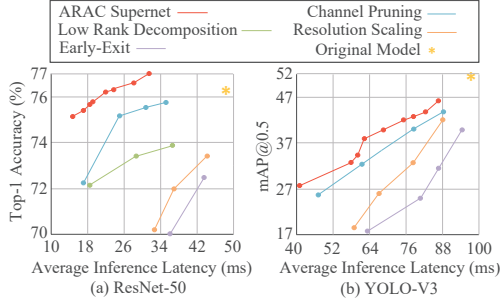


Figure 13: Comparison w/ model compression techniques.

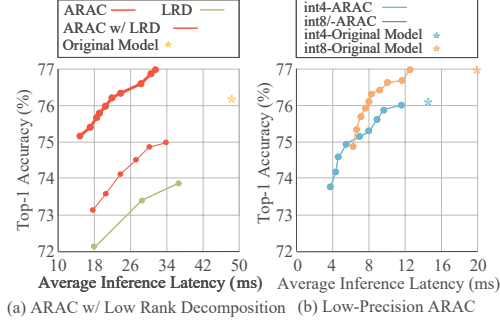


Figure 14: ARAC supernet: (a) with low rank decomposition, and (b) in low precision.

low-rank decomposition rather than channel pruning. As shown in Fig. 14 (a), the ARAC supernet with low-rank decomposition (dashed red curve) raises the accuracy by 1.1% compared to basic low-rank decomposition. Its performance is 2% lower than the ARAC supernet with channel pruning (solid red curve), which justifies the rationality of utilizing channel pruning to generate an ARAC supernet in our design (§4).

Low-Precision: Like other studies on model compression and dynamic inference, ARAC supernet is also a technique that is complementary to the studies on low-precision. As shown in Fig. 14 (b), we integrate low-precision technique [1] into ARAC supernet and reduce the precision of weights in ARAC supernet to *int8* and *int4*. The performance of (low-precision) ARAC is compared with that of the original low-precision model (marked in star symbols). Similar to the *float32* results, ARAC supernet realizes a wide range of accuracy and latency trade-offs above the original (low-precision) model, which proves the complementarity between ARAC supernet and low-precision technique. Note that most studies on DNN model design [21, 25, 58] and compression [39, 40, 79] keep in the original precision (*float32*) because the generation of low-precision model suffers from high engineering complexity including configuration calibration procedure, sensitivity to training settings, and frequent tuning of hyper-parameters during training [1, 52, 67].

7.5 ARAC under Background Loads

We examine the performance of ARAC supernet under background loads by changing the running conditions of the device with a GPU-intensive application (Gaussian Elimination in the Rodinia Benchmark Suite [4]), which is widely used by mobile computing evaluation [84]. For the latency and memory predictors in DEMUX (§5.3), we can easily modify and train them to predict under varying system contention (memory bandwidth and CPU/GPU

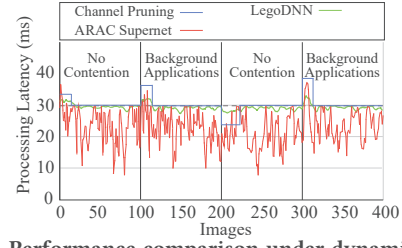


Figure 15: Performance comparison under dynamic device contention (ResNet-50 [21], Jetson TX2, 30ms latency constraint).

usage) by adding them as inputs [84]. In Fig 15, we show the performance comparison of ARAC supernet and two baseline methods, LegoDNN [19] and channel pruning [39]. As shown in Fig. 15, all three methods are adaptive to the changing contention on the device and keep most of their inference execution within the latency constraint (30ms). The processing latency of ARAC supernet fluctuates in a wide range (under 30ms) because it is a content-aware method that makes an individual decision on how much computation is assigned to each image according to its content. We also observe that, the average accuracy of ARAC supernet is 1.5% higher than that of LegoDNN and 2.8% higher than that of channel-pruning in the test.

7.6 ARAC in Continuous Object Detection

Recent works on continuous object detection [5, 36, 47, 84] exploit the temporal correlation between consecutive frames and reduce the overall computation of video streams by combining model detection with tracking algorithms [47, 84]. In contrast, ARAC supernet, by modifying the detection model's computation directly, reduces computation per frame in video streams. In general, applying ARAC supernet into a continuous object detection system: (1) significantly reduces model inference latency in processing a frame (Fig. 16); (2) increases the system's adaptivity to various conditions like cross-frame similarity, sampling rate (FPS), and processing deadline (Fig. 18). We implement three continuous object detection systems: offloading (all frames are processed by the detection model on the server), Edge-Assisted [47], and local [84]. The detection model is YOLO-V3, the server is with RTX 2070, and the local device is Jetson TX2. The network is WiFi 2.4GHz. As shown in Fig. 16, with ARAC supernet, the end-to-end latencies of processing a frame with the detection model show a significant reduction (36.0% to 54.1%) in all three systems.

We further demonstrate how ARAC supernet contributes to the overall performance of a continuous object detection system in two videos. Video-1 and Video-2 are two one-hour videos taken on the side of a vehicle at 60FPS. As shown in Fig. 17, with different moving speeds (Video-1: at 45 to 70 mph, Video-2: at around 20mph), the similarities (SSIM) of the two videos deviates from each other. In Fig. 18 (a), we observe the accuracy trend along with different sampling rate (latency constraint= 30ms). As ARAC supernet reduces computation in the detection model, it allows more frames to be processed within a fixed period. Thus, the systems with ARAC supernet show high adaptivity to the increase of sampling rate. For example, the accuracy of offloading with ARAC supernet keeps the highest in all cases. In contrast, the accuracy of Edge-Assisted is up to 88.5% lower than that of offloading with ARAC supernet. In

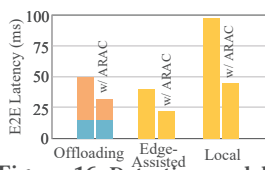


Figure 16: Detection model latency.

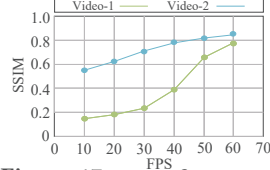


Figure 17: Cross-frame similarity.

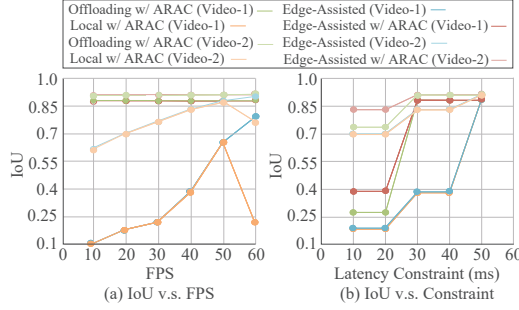


Figure 18: Performance comparison of continuous object detection.

Fig. 18 (b), we observe the performance (IoU as metric [47]) under different per-frame latency constraints. Specifically, the detection results of a frame are required to be obtained within a latency constraint once a frame is sampled². To satisfy a small latency constraint, the low-accuracy detection results can be frequently generated by the tracker. Consequently, Edge-Assisted has poor performance with small latency constraints on videos with low similarity (e.g., Video-1). For example, for Video-1, its IoU is < 0.4 when the latency constraint is less than 40ms. In contrast, offloading with ARAC supernet outperforms Edge-Assisted by up to 1.25 \times when the latency constraints are 30 and 40ms. Local with ARAC supernet also achieves comparable performance with Edge-Assisted in Fig. 18 (b). When the server has poorer GPU (e.g., GTX 1070), local with ARAC supernet even outperforms Edge-Assisted by up to 95% when the latency constraint is 30ms. We also observe that the combination of ARAC supernet with Edge-Assisted achieves the highest performance in all cases. Compared to Edge-Assisted, the Edge-Assisted with ARAC supernet increases accuracy by up to 1.25 \times and 7.7 \times in Fig. 18 (a) and (b), respectively.

7.7 ARAC in Various Vision Applications

Besides image classification and object detection, we also evaluate the performance of ARAC supernet in other vision applications, including action recognition [16], traffic accident detection [2], abnormal activity detection [10], traffic sign recognition [66], flower classification [82], vehicle detection [64], fall detection [6], vehicle make and model recognition [70]. As shown in Fig. 19, ARAC supernet achieves the highest accuracy under the same constraints in all applications. Reducing computation based on spatial redundancy allows ARAC supernet to preserve high accuracy with lower latency and less memory consumption.

²Such latency constraint is common in real applications because the detection results are often utilized by other applications like augmented reality [18] and autonomous driving [14].

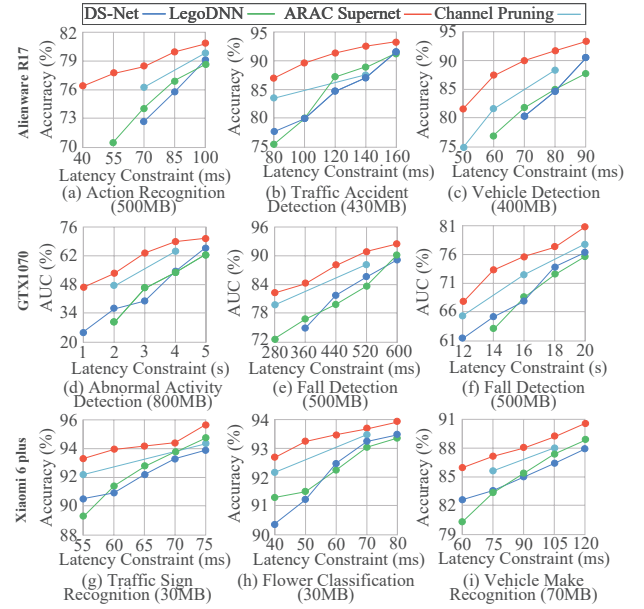


Figure 19: Accuracy under latency/memory constraints in 9 different vision applications.

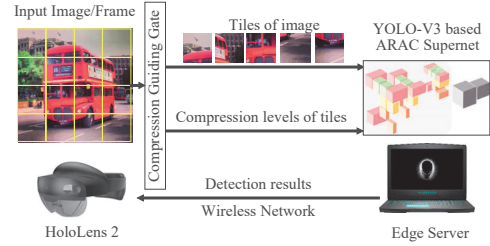


Figure 20: DeepMix [18] w/ ARAC supernet.

7.8 Evaluation on Mixed-Reality Platform

We implement ARAC supernet on the SOTA Mixed-Reality platform for 3D object detection, DeepMix [17, 18], as shown in Fig. 20. DeepMix implements 3D object detection based on detection results of 2D object detection models like YOLO [18]. We run CGG on the MR device (HoloLens), and only the tiles with a compression ratio higher than 0 are sent to the edge server (Alienware 17 R3) for processing. Neglecting computing tiles with zero compression ratio only causes less than 4.7% accuracy loss. The server then sends the detection results (bounding boxes and objects' classes) back to the MR device. We observe that both transmission latency and computing latency on the server is significantly reduced with ARAC supernet under different network conditions. As shown in Fig. 21, ARAC supernet reduces end-to-end latency by 32.2% to 49.3% compared to the original DeepMix implementation under different network conditions. The data processing latency on the server is reduced by 50.7%, and the data transmission latency is reduced by 56.5% to 66.7%. The overhead of the computing guiding gate on the MR device is less than 4.2ms.

7.9 Cross-Domain and Multi-Scale Performance

We evaluate the performance of the compression guiding gate (CGG) that is trained on one domain (e.g., a dataset) but is applied to

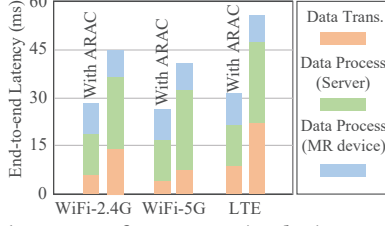


Figure 21: Latency of DeepMix w/ and w/o ARAC supernet.

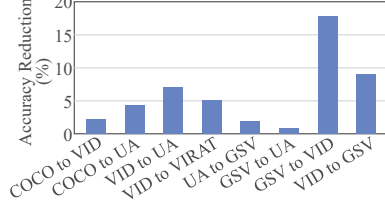


Figure 22: Cross-domain performance (Base Model: YOLO-V3, Jetson TX2, memory constraint= 500MB, latency constraint= 75ms).

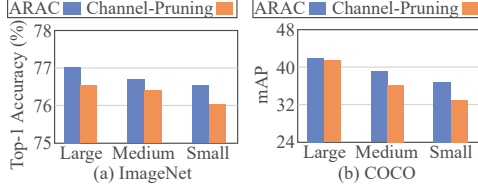


Figure 23: Multi-scale observation: (a) ResNet-50, (b) YOLO-V3.

another domain (e.g., another dataset). Besides COCO dataset [46], we include four other datasets that are collected on various locations and with different contents: VID [62], UA [78] (high-way traffic in China), VIRAT [54] (on-campus), and GSV [91] (google street view). As shown in Fig. 22, applying a CGG from another domain generates an accuracy reduction of $< 10\%$ in most cross-domain cases, which indicates the transferability of the CGG. *Note* that the poor cross-domain performance between GSV and VID is due to the large difference between the two datasets' contents. In practice, a common deep learning solution is to train the model on large datasets like ImageNet and COCO, then to finetune the model on the customized dataset to make the model adaptive to the specific application.

We observe the accuracy distribution of ARAC supernets regarding the sizes of objects in images. For the ImageNet dataset (image classification), we group objects into large, medium, and small size as $> 70\%$, between 40% and 70% , $< 40\%$ area of the whole image, respectively. For the COCO dataset (object detection), we group objects into large, medium, and small size as $> 50\%$, between 25% and 50% , $< 25\%$ area of the whole image, respectively. As shown in Fig. 23, ARAC supernet outperforms channel-pruning in all sizes. Especially, it shows significant accuracy improvement on small-size objects by 0.6% (image classification) and 12% (object detection). The underlying reason is that the recognition on small objects is easily affected by other pixels of the image when processed by spatial-agnostic inference like channel-pruning. As ARAC supernet adaptively allocates the computation on different pixels in an image, it prevents interference from redundant pixels on the final results.

8 DISCUSSION

• *Training overhead of ARAC supernet:* In Table 1, we compare the training time of the ARAC supernet with channel-pruning [39]

ARAC Supernet (five levels+CGG)	Channel-Pruning (four levels)	Original Model
27.5h	67h	218h

Table 1: Comparison of training time (base model: ResNet-50).





Example images				
Original output	Brittany spaniel	Lynx	Spider monkey	Albatross
ARAC output	English setter	Persian cat	Titi	Mink
Ground truth	English setter	Persian cat	Titi	Mink

Figure 24: Image examples from ImageNet validation dataset [7], original model: ResNet-50 [21].

and the original model. We utilize ResNet-50 as the base model and train on ImageNet dataset [7]. As ARAC supernet integrates sub-networks with different compression levels and trains them together with one loss function (§4.3), the sharing of model parameters among sub-networks allows all sub-networks to converge synchronously. The channel-pruned model with different compression levels can only be trained individually. Thus, compared to channel-pruning [39], the ARAC supernet can finish training within less than 50% of the training time for channel-pruning.

• *Online Overhead:* We observe the latency breakdown of the end-to-end processing latency with ARAC supernet. Overall, the average overhead of CGG and DEMUX (including CLG) is less than 7.3% of the total execution latency.

• *Accuracy improvement by ARAC supernet:* As our approach makes the network focus on processing regions in an image that contain key features related to the vision task, the side-effect of redundant pixels is weakened or eliminated, which is equivalent to strengthen the effects of key-feature pixels on the final output. For cases when the side-effect from redundant pixels are so strong that the original network is misled to a wrong output, the ARAC supernet may help the model to remove the side-effect and output correct results. We show four such examples in Fig. 24. Similar observations are reported by [95] and [43].

9 CONCLUSION

In this paper, we proposed a spatial-based dynamic CNN acceleration framework, NeuLens, adaptive to users' SLOs for mobile and edge platforms. A novel dynamic inference mechanism, ARAC supernet, was presented. Splitting image into tiles can adaptively select computational cost for each tile and ensure speed-up on general platforms. We also proposed an online controller to tune ARAC supernet based on users' SLOs with neglectable overhead on mobile/edge devices. NeuLens outperforms baseline methods by up to 58% latency reduction with the same accuracy and by up to 67.9% accuracy improvement under the same latency/memory constraints. Significant performance improvements are observed in mobile vision applications like continuous object detection and 3D object detection for MR devices.

10 ACKNOWLEDGEMENTS

We sincerely thank our anonymous shepherd and reviewers for their valuable comments. This work is partially supported by the US National Science Foundation under Grant No. 2147821, No. 2147623, No. 2047655, and No. 2049875.

REFERENCES

- [1] AmirAli Abdolrashidi et al. "Pareto-optimal quantized resnet is mostly 4-bit". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2021, pp. 3091–3099.
- [2] Aman Kumar Agrawal et al. "Automatic traffic accident detection system using ResNet and SVM". In: *2020 Fifth International Conference on Research in Computational Intelligence and Communication Networks (ICRCICN)*. IEEE. 2020, pp. 71–76.
- [3] Tolga Bolukbasi et al. "Adaptive neural networks for efficient inference". In: *International Conference on Machine Learning*. PMLR. 2017, pp. 527–536.
- [4] Shuai Che et al. "Rodinia: A benchmark suite for heterogeneous computing". In: *2009 IEEE international symposium on workload characterization (IISWC)*. Ieee. 2009, pp. 44–54.
- [5] Tiffany Yu-Han Chen et al. "Glimpse: Continuous, Real-Time Object Recognition on Mobile Devices". In: *Proceedings of the 13th ACM Conference on Embedded Networked Sensor Systems, SenSys 2015, Seoul, South Korea, November 1-4, 2015*. Ed. by June-hwa Song, Tarek F. Abdelzaher, and Cecilia Mascolo. ACM, 2015, pp. 155–168. doi: 10.1145/2809695.2809711. URL: <https://doi.org/10.1145/2809695.2809711>.
- [6] Sagar Chhetri et al. "Deep learning for vision-based fall detection system: Enhanced optical dynamic flow". In: *Computational Intelligence 37.1* (2021), pp. 578–595.
- [7] Jia Deng et al. "Imagenet: A large-scale hierarchical image database". In: *2009 IEEE conference on computer vision and pattern recognition*. Ieee. 2009, pp. 248–255.
- [8] Xuanyi Dong et al. "More is less: A more complicated network with less inference complexity". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2017, pp. 5840–5848.
- [9] Kuntai Du et al. "Server-driven video streaming for deep learning inference". In: *Proceedings of the Annual conference of the ACM Special Interest Group on Data Communication on the applications, technologies, architectures, and protocols for computer communication*. 2020, pp. 557–570.
- [10] Shikha Dubey, Abhijeet Boragule, and Moongu Jeon. "3d resnet with ranking loss function for abnormal activity detection in videos". In: *2019 International Conference on Control, Automation and Information Sciences (ICCAIS)*. IEEE. 2019, pp. 1–6.
- [11] Gamaleldin Elsayed, Simon Kornblith, and Quoc V Le. "Saccader: Improving accuracy of hard attention models for vision". In: *Advances in Neural Information Processing Systems 32* (2019).
- [12] Biyi Fang, Xiao Zeng, and Mi Zhang. "Nestdnn: Resource-aware multi-tenant on-device deep learning for continuous mobile vision". In: *Proceedings of the 24th Annual International Conference on Mobile Computing and Networking*. 2018, pp. 115–127.
- [13] Biyi Fang et al. "FlexDNN: Input-adaptive on-device deep learning for efficient mobile vision". In: *2020 IEEE/ACM Symposium on Edge Computing (SEC)*. IEEE. 2020, pp. 84–95.
- [14] Di Feng et al. "A review and comparative study on probabilistic object detection in autonomous driving". In: *IEEE Transactions on Intelligent Transportation Systems* (2021).
- [15] Michael Figurnov et al. "Spatially adaptive computation time for residual networks". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2017, pp. 1039–1048.
- [16] Deeptha Girish, Vineeta Singh, and Anca Ralescu. "Understanding action recognition in still images". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*. 2020, pp. 370–371.
- [17] Yongjie Guan et al. "DeepMix: A Real-time Adaptive Virtual Content Registration System with Intelligent Detection". In: *IEEE INFOCOM 2021-IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*. IEEE. 2021, pp. 1–2.
- [18] Yongjie Guan et al. "DeepMix: Mobility-aware, Lightweight, and Hybrid 3D Object Detection for Headsets". In: *arXiv preprint arXiv:2201.08812* (2022).
- [19] Rui Han et al. "LegoDNN: block-grained scaling of deep neural networks for mobile vision". In: *Proceedings of the 27th Annual International Conference on Mobile Computing and Networking*. 2021, pp. 406–419.
- [20] Yizeng Han et al. "Spatially adaptive feature refinement for efficient inference". In: *IEEE Transactions on Image Processing* 30 (2021), pp. 9345–9358.
- [21] Kaiming He et al. "Deep residual learning for image recognition". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 770–778.
- [22] Seonyeong Heo et al. "Real-time object detection system with multi-path neural networks". In: *2020 IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS)*. IEEE. 2020, pp. 174–187.
- [23] Xueyu Hou et al. "TrustServing: A quality inspection sampling approach for remote DNN services". In: *2020 17th Annual IEEE International Conference on Sensing, Communication, and Networking (SECON)*. IEEE. 2020, pp. 1–9.
- [24] Xueyu Hou et al. "Distredge: Speeding up convolutional neural network inference on distributed edge devices". In: *2022 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*. IEEE. 2022, pp. 1097–1107.
- [25] Andrew Howard et al. "Searching for mobilenetv3". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2019, pp. 1314–1324.
- [26] Andrew G Howard et al. "Mobilenets: Efficient convolutional neural networks for mobile vision applications". In: *arXiv preprint arXiv:1704.04861* (2017).
- [27] Weizhe Hua et al. "Boosting the performance of cnn accelerators with dynamic fine-grained channel gating". In: *Proceedings of the 52nd Annual IEEE/ACM International Symposium on Microarchitecture*. 2019, pp. 139–150.
- [28] Weizhe Hua et al. "Channel gating neural networks". In: *Advances in Neural Information Processing Systems 32* (2019).
- [29] Gao Huang et al. "Condensenet: An efficient densenet using learned group convolutions". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018, pp. 2752–2761.
- [30] Gao Huang et al. "Glance and Focus Networks for Dynamic Visual Recognition". In: *arXiv preprint arXiv:2201.03014* (2022).
- [31] Gao Huang et al. "Multi-scale dense networks for resource efficient image classification". In: *arXiv preprint arXiv:1703.09844* (2017).
- [32] Itay Hubara et al. "Binarized neural networks". In: *Advances in neural information processing systems* 29 (2016).
- [33] Benoit Jacob et al. "Quantization and training of neural networks for efficient integer-arithmetic-only inference". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018, pp. 2704–2713.
- [34] Eric Jang, Shixiang Gu, and Ben Poole. "Categorical reparameterization with gumbel-softmax". In: *arXiv preprint arXiv:1611.01144* (2016).
- [35] Borui Jiang and Yadong Mu. "Russian Doll Network: Learning Nested Networks for Sample-Adaptive Dynamic Inference". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2021, pp. 336–344.
- [36] Junchen Jiang et al. "Chameleon: scalable adaptation of video analytics". In: *Proceedings of the 2018 Conference of the ACM Special Interest Group on Data Communication, SIGCOMM 2018, Budapest, Hungary, August 20-25, 2018*. Ed. by Sergey Gorinsky and János Tapolcai. ACM, 2018, pp. 253–266. doi: 10.1145/3230543.3230574. URL: <https://doi.org/10.1145/3230543.3230574>.
- [37] Shiqi Jiang et al. "Flexible high-resolution object detection on edge devices with tunable latency". In: *Proceedings of the 27th Annual International Conference on Mobile Computing and Networking*. 2021, pp. 559–572.
- [38] Woonchul Kang, Daeyeon Kim, and Junyoung Park. "Dms: Dynamic model scaling for quality-aware deep learning inference in mobile and embedded devices". In: *IEEE Access* 7 (2019), pp. 168048–168059.
- [39] Ashish Khethan and Zohar Karnin. "Prunenet: Channel pruning via global importance". In: *arXiv preprint arXiv:2005.11282* (2020).
- [40] Yong-Deok Kim et al. "Compression of deep convolutional neural networks for fast and low power mobile applications". In: *arXiv preprint arXiv:1511.06530* (2015).
- [41] Stefanos Laskaridis et al. "SPINN: synergistic progressive inference of neural networks over device and cloud". In: *Proceedings of the 26th Annual International Conference on Mobile Computing and Networking*. 2020, pp. 1–15.
- [42] Yann LeCun, John Denker, and Sara Solla. "Optimal brain damage". In: *Advances in neural information processing systems* 2 (1989).
- [43] Changlin Li et al. "Dynamic slimmable network". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2021, pp. 8607–8617.
- [44] Hao Li et al. "Pruning filters for efficient convnets". In: *arXiv preprint arXiv:1608.08710* (2016).
- [45] Ji Lin et al. "Runtime neural pruning". In: *Advances in neural information processing systems* 30 (2017).
- [46] Tsung-Yi Lin et al. "Microsoft coco: Common objects in context". In: *European conference on computer vision*. Springer. 2014, pp. 740–755.
- [47] Luyang Liu, Hongyu Li, and Marco Gruteser. "Edge assisted real-time object detection for mobile augmented reality". In: *The 25th Annual International Conference on Mobile Computing and Networking*. 2019, pp. 1–16.
- [48] Miaomiao Liu, Xianzhong Ding, and Wan Du. "Continuous, real-time object detection on mobile devices without offloading". In: *2020 IEEE 40th International Conference on Distributed Computing Systems (ICDCS)*. IEEE. 2020, pp. 976–986.
- [49] Zhuang Liu et al. "Learning efficient convolutional networks through network slimming". In: *Proceedings of the IEEE international conference on computer vision*. 2017, pp. 2736–2744.
- [50] Zhuang Liu et al. "Rethinking the value of network pruning". In: *arXiv preprint arXiv:1810.05270* (2018).
- [51] Ningning Ma et al. "Shufflenet v2: Practical guidelines for efficient cnn architecture design". In: *Proceedings of the European conference on computer vision (ECCV)*. 2018, pp. 116–131.
- [52] Jeffrey L McKinstry et al. "Discovering low-precision networks close to full-precision networks for efficient embedded inference". In: *arXiv preprint arXiv:1809.04191* (2018).

- [53] Wei Niu et al. "Patdnn: Achieving real-time dnn execution on mobile devices with pattern-based weight pruning". In: *Proceedings of the Twenty-Fifth International Conference on Architectural Support for Programming Languages and Operating Systems*. 2020, pp. 907–922.
- [54] Sangmin Oh et al. "A large-scale benchmark dataset for event recognition in surveillance video". In: *CVPR 2011*. IEEE. 2011, pp. 3153–3160.
- [55] Athanasios Papadopoulos, Pawel Korus, and Nasir Memon. "Hard-attention for scalable image classification". In: *Advances in Neural Information Processing Systems* 34 (2021).
- [56] Adam Paszke et al. "PyTorch: An Imperative Style, High-Performance Deep Learning Library". In: *Advances in Neural Information Processing Systems* 32. Ed. by H. Wallach et al. Curran Associates, Inc., 2019, pp. 8024–8035. URL: <http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf>.
- [57] Mohammad Rastegari et al. "Xnor-net: Imagenet classification using binary convolutional neural networks". In: *European conference on computer vision*. Springer. 2016, pp. 525–542.
- [58] Joseph Redmon and Ali Farhadi. "Yolov3: An incremental improvement". In: *arXiv preprint arXiv:1804.02767* (2018).
- [59] Mengye Ren et al. "Sbnet: Sparse blocks network for fast inference". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018, pp. 8711–8720.
- [60] Adria Ruiz and Jakob Verbeek. "Adaptive inference cost with convolutional neural mixture models". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2019, pp. 1872–1881.
- [61] Adria Ruiz and Jakob Verbeek. "Anytime inference with distilled hierarchical neural ensembles". In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 35. 11. 2021, pp. 9463–9471.
- [62] Olga Russakovsky et al. "ImageNet Large Scale Visual Recognition Challenge". In: *International Journal of Computer Vision (IJCV)* 115.3 (2015), pp. 211–252. doi: 10.1007/s11263-015-0816-y.
- [63] Mark Sandler et al. "Mobilenetv2: Inverted residuals and linear bottlenecks". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018, pp. 4510–4520.
- [64] Jun Sang et al. "An improved YOLOv2 for vehicle detection". In: *Sensors* 18.12 (2018), p. 4272.
- [65] Noam Shazeer et al. "Outrageously large neural networks: The sparsely-gated mixture-of-experts layer". In: *arXiv preprint arXiv:1701.06538* (2017).
- [66] Johannes Stallkamp et al. "The German traffic sign recognition benchmark: a multi-class classification competition". In: *The 2011 international joint conference on neural networks*. IEEE. 2011, pp. 1453–1460.
- [67] Qigong Sun et al. "Effective and fast: A novel sequential single path search for mixed-precision quantization". In: *arXiv preprint arXiv:2103.02904* (2021).
- [68] Wenyu Sun et al. "A 112-765 GOPS/W FPGA-based CNN Accelerator using Importance Map Guided Adaptive Activation Sparsification for Pix2pix Applications". In: *2020 IEEE Asian Solid-State Circuits Conference (A-SSCC)*. 2020, pp. 1–4. doi: 10.1109/A-SSCC48613.2020.9336115.
- [69] Christian Szegedy et al. "Going deeper with convolutions". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015, pp. 1–9.
- [70] Faezeh Tafazzoli, Hichem Frigui, and Keishin Nishiyama. "A large and diverse dataset for improved vehicle make and model recognition". In: *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*. 2017, pp. 1–8.
- [71] Mingxing Tan and Quoc Le. "Efficientnet: Rethinking model scaling for convolutional neural networks". In: *International conference on machine learning*. PMLR. 2019, pp. 6105–6114.
- [72] Chen Tang et al. "Adaptive Pixel-wise Structured Sparse Network for Efficient CNNs". In: *arXiv preprint arXiv:2010.11083* (2020).
- [73] Andreas Veit and Serge Belongie. "Convolutional networks with adaptive inference graphs". In: *Proceedings of the European Conference on Computer Vision (ECCV)*. 2018, pp. 3–18.
- [74] Thomas Verelst and Tinne Tuytelaars. "Dynamic convolutions: Exploiting spatial sparsity for faster inference". In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2020, pp. 2320–2329.
- [75] Huanyu Wang et al. "CoDiNet: Path Distribution Modeling with Consistency and Diversity for Dynamic Routing". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2021).
- [76] Xin Wang et al. "Skipnet: Learning dynamic routing in convolutional networks". In: *Proceedings of the European Conference on Computer Vision (ECCV)*. 2018, pp. 409–424.
- [77] Yulin Wang et al. "Glance and focus: a dynamic approach to reducing spatial redundancy in image classification". In: *Advances in Neural Information Processing Systems* 33 (2020), pp. 2432–2444.
- [78] Longyin Wen et al. "UA-DETRAC: A new benchmark and protocol for multi-object detection and tracking". In: *Computer Vision and Image Understanding* 193 (2020), p. 102907.
- [79] Maciej Wolczyk et al. "Zero Time Waste: Recycling Predictions in Early Exit Neural Networks". In: *Advances in Neural Information Processing Systems* 34 (2021).
- [80] Sanghyun Woo et al. "Cbam: Convolutional block attention module". In: *Proceedings of the European conference on computer vision (ECCV)*. 2018, pp. 3–19.
- [81] Zuxuan Wu et al. "Blockdrop: Dynamic inference paths in residual networks". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018, pp. 8817–8826.
- [82] Xiaoling Xia, Cui Xu, and Bing Nan. "Inception-v3 for flower classification". In: *2017 2nd international conference on image, vision and computing (ICIVC)*. IEEE. 2017, pp. 783–787.
- [83] Zhenda Xie et al. "Spatially adaptive inference with stochastic feature sampling and interpolation". In: *European conference on computer vision*. Springer. 2020, pp. 531–548.
- [84] Ran Xu et al. "ApproxDet: content and contention-aware approximate object detection for mobiles". In: *Proceedings of the 18th Conference on Embedded Networked Sensor Systems*. 2020, pp. 449–462.
- [85] Zirui Xu et al. "Reform: Static and dynamic resource-aware dnn reconfiguration framework for mobile device". In: *Proceedings of the 56th Annual Design Automation Conference 2019*. 2019, pp. 1–6.
- [86] Brandon Yang et al. "Condconv: Conditionally parameterized convolutions for efficient inference". In: *Advances in Neural Information Processing Systems* 32 (2019).
- [87] Le Yang et al. "Resolution adaptive networks for efficient inference". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2020, pp. 2369–2378.
- [88] Jaehyoung Yoo et al. "RaScaNet: Learning Tiny Models by Raster-Scanning Images". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2021, pp. 13673–13682.
- [89] Jiahui Yu and Thomas S Huang. "Universally slimmable networks and improved training techniques". In: *Proceedings of the IEEE/CVF international conference on computer vision*. 2019, pp. 1803–1811.
- [90] Jiahui Yu et al. "Slimmable neural networks". In: *arXiv preprint arXiv:1812.08928* (2018).
- [91] A.R. Zamir and M. Shah. *Image Geo-localization Based on Multiple Nearest Neighbor Feature Matching using Generalized Graphs*. 2014. doi: 10.1109/TPAMI.2014.2299799.
- [92] Wuyang Zhang et al. "Elf: accelerate high-resolution mobile deep vision with content-aware parallel offloading". In: *Proceedings of the 27th Annual International Conference on Mobile Computing and Networking*. 2021, pp. 201–214.
- [93] Xiangyu Zhang et al. "Shufflenet: An extremely efficient convolutional neural network for mobile devices". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018, pp. 6848–6856.
- [94] Yikang Zhang, Zhuo Chen, and Zhao Zhong. "Collaboration of experts: Achieving 80% top-1 accuracy on imagenet with 100m flops". In: *arXiv preprint arXiv:2107.03815* (2021).
- [95] Mingjian Zhu et al. "Dynamic Resolution Network". In: *Advances in Neural Information Processing Systems* 34 (2021).