







EdgeCompress: Coupling Multi-Dimensional Model Compression and Dynamic Inference for EdgeAI

Hao Kong , Di Liu , *Member, IEEE*, Shuo Huai , Xiangzhong Luo , Ravi Subramaniam , Christian Makaya, Qian Lin, Weichen Liu , *Member, IEEE*

Abstract—Convolutional neural networks (CNNs) have demonstrated encouraging results in image classification tasks. However, the prohibitive computational cost of CNNs hinders the deployment of CNNs onto resource-constrained embedded devices. To address this issue, we propose *EdgeCompress*, a comprehensive compression framework to reduce the computational overhead of CNNs. In *EdgeCompress*, we first introduce dynamic image cropping, where we design a lightweight foreground predictor to accurately crop the most informative foreground object of input images for inference, which avoids redundant computation on background regions. Subsequently, we present compound shrinking to collaboratively compress the three dimensions (depth, width, and resolution) of CNNs according to their contribution to accuracy and model computation. Dynamic image cropping and compound shrinking together constitute a multi-dimensional CNN compression framework, which is able to comprehensively reduce the computational redundancy in both input images and neural network architectures, thereby improving the inference efficiency of CNNs. Further, we present a dynamic inference framework to efficiently process input images with different recognition difficulties, where we cascade multiple models with different complexities from our compression framework and dynamically adopt different models for different input images, which further compresses the computational redundancy and improves the inference efficiency of CNNs, facilitating the deployment of advanced CNNs onto embedded hardware. Experiments on ImageNet-1K demonstrate that *EdgeCompress* reduces the computation of ResNet-50 by 48.8% while improving the top-1 accuracy by 0.8%. Meanwhile, we improve the accuracy by 4.1% with similar computation compared to HRank, the state-of-the-art compression framework. The source code and models are available at <https://github.com/ntuliuteam/edge-compress>

Index Terms—Embedded systems, neural network compression, hardware/software co-design, dynamic neural network

I. INTRODUCTION

Convolutional neural networks (CNNs) have gained popularity in image classification tasks [1]. Benefiting from the advances in high-quality datasets [1], [2] and network architecture designs [3], [4], the accuracy of modern CNNs has been constantly improved. Nevertheless, such accuracy improvement comes at higher computational overhead [3]–[5].

Corresponding author: Weichen Liu (email: liu@ntu.edu.sg)

Hao Kong, Shuo Huai are with the School of Computer Science and Engineering, Nanyang Technological University (NTU), Singapore, and also with HP-NTU Digital Manufacturing Corporate Lab, Nanyang Technological University (NTU), Singapore.

Di Liu is with Department of Computer Science, Norwegian University of Science and Technology (NTNU), Trondheim, Norway.

Xiangzhong Luo and Weichen Liu are with the School of Computer Science and Engineering, Nanyang Technological University (NTU), Singapore.

Ravi Subramaniam, Christian Makaya, and Qian Lin are with HP Inc., Palo Alto, CA, USA.

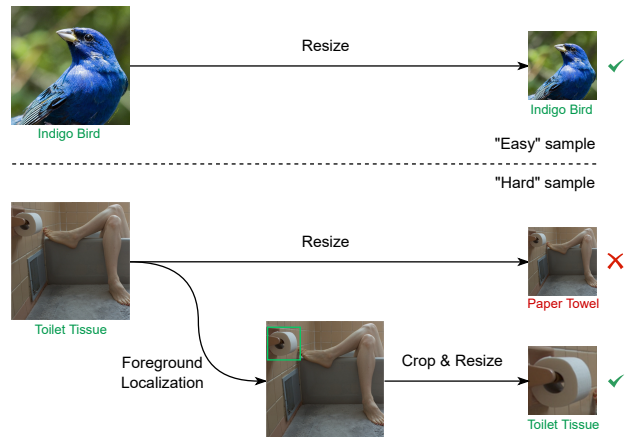


Fig. 1. The predictions from ResNet-50. For easy samples, the network can still generate correct predictions at a smaller resolution (e.g., 112×112 for ImageNet-1K). For hard samples, simply resizing images to a smaller resolution can lead to misclassification, while using dynamic cropping can correctly classify hard samples at a smaller resolution.

Recently, to mitigate data transmission latency and respond to the growing concern about data privacy, a new paradigm named EdgeAI has emerged, which deploys CNNs onto embedded devices near users to process data locally instead of uploading data to the cloud [6], [7]. However, embedded devices are usually resource-constrained and in turns are unable to accommodate resource-hungry CNNs. To facilitate the deployment of advanced CNNs onto resource-constrained embedded devices, efforts have been made to compress the computational overhead of CNNs.

The computation of a CNN, i.e., the Multiply-Accumulate Operations (MACs), mainly results from two aspects: 1) high-resolution input images and 2) gigantic network architectures. To reduce the computational redundancy in input images (i.e., spatial redundancy), many works propose to reduce the resolution (i.e., the height or width of input images) for inference [8]–[10]. However, as shown in the motivational example in Fig. 1, this coarse spatial redundancy reduction approach is only effective for images with clear foreground. For images in which the foreground only occupies a small portion of the whole image, directly shrinking the whole image will lose important features of the foreground, leading to a wrong prediction. This observation triggers our first motivation:

Motivation 1: Can we reduce the inference resolution of input images without sacrificing accuracy?

On the other hand, network pruning [11]–[16] is also proposed to compress the depth (i.e., the number of layers) and width (i.e., the number of channels in each layer) of the network architecture. Specifically, to optimize the efficiency of CNNs, width pruning [11], [12], [14], [16] devotes to removing less sensitive channels in each layer to yield ‘thinner’ networks, while depth pruning [15] conducts pruning at a coarser granularity (i.e., layer), which directly removes unimportant layers to construct ‘shallower’ networks. However, the above techniques only reduce the redundancy in a single dimension of CNNs while ignoring the redundancy in the other dimensions. Such single-dimensional compression approaches can only achieve a very limited compression rate. This phenomenon brings us to the second motivation:

Motivation 2: Can we combine the compression of all three dimensions of CNNs to achieve a higher compression rate while maintaining high accuracy?

In addition, given a resource budget, existing approaches usually yield a fixed compressed neural network and resolution for all images. However, as discussed in [10], [17], different images are of distinct recognition difficulties, using a static model and resolution to process all images can lead to inefficient utilization of computation, achieving only sub-optimal efficiency and accuracy. Practically, for images with simple features, a small CNN model is adequate to generate correct results. For complex images, a larger model with higher capability should be used to extract high-level features for a correct prediction. This inspires our last motivation:

Motivation 3: Can we dynamically adjust the model and resolution for different images during inference to further optimize inference efficiency and accuracy?

To address the above questions for more efficient image classification with CNNs, we, in this paper, propose a novel inference framework, *EdgeCompress*, to comprehensively reduce the inference overhead of CNNs, thereby optimizing the classification efficiency of CNNs and facilitating the deployment of advanced CNNs onto edge devices. In *EdgeCompress*, we first propose a two-stage multi-dimensional model compression framework to coordinately compress all three dimensions of CNNs. In the first stage, we introduce a novel dynamic image cropping (DIC) strategy to accurately remove the spatial redundancy in input images, in which we design a lightweight foreground predictor to efficiently localize the most discriminative foreground of input images, then only the detected foreground will be preserved for classification and the redundant background will be discarded. As shown in Figure 1, through the dynamic image cropping strategy, we are capable of generating fine-cropped images with less spatial redundancy, thereby achieving satisfactory classification accuracy even at a smaller resolution. In the second stage, we present a compound shrinking (CS) strategy to jointly compress the three dimensions of CNNs, thereby further reducing the redundancy in input images and net-

work architectures. We first quantify the impact of shrinking different dimensions on model complexity and accuracy, according to which we automatically calculate a shrinking coefficient for each dimension to coordinate the shrinking of different dimensions to achieve a higher compression rate while still maintaining the accuracy. By the means of the two-stage multi-dimensional compression framework, given a computation budget, we are able to comprehensively reduce redundant computation to meet the budget without sacrificing accuracy obviously. Based on the compression framework, we further propose a novel dynamic inference framework to adaptively process different input images with different models and resolutions at runtime. First, we utilize the compound shrinking strategy to compress the give baseline network and generate multiple sub-networks with diverse model sizes and accuracy, which are then cascaded in ascending order of the model size and then each input image will be processed by those models sequentially. At the end of the inference of each model, we propose a novel metric to evaluate the confidence of the prediction result. Once a confident prediction is obtained, the dynamic inference will be terminated without executing subsequent models. In practice, most input images can be confidently recognized by early models with small computational overhead, while large models will be activated only for a few hard samples. Consequently, compared to static inference with a single model, the overall computational complexity of our dynamic inference is reduced significantly without compromising accuracy. Our main contributions are summarized as follows:

- 1) We propose dynamic image cropping to reduce the spatial redundancy in images, where we design a lightweight detector to efficiently localize the foreground area of an image and conduct instance-aware dynamic cropping. Those finely cropped images can be correctly recognized even at a smaller resolution, which greatly reduces the computational cost of CNNs.
- 2) We also propose compound shrinking to jointly compress the three dimensions of a CNN. We first quantify the impact of each dimension on accuracy and model complexity, and then generate the optimal joint compression strategy accordingly. By this means, we greatly reduce the redundancy in both input images and network architectures for a higher compression rate.
- 3) We further introduce a dynamic inference framework to efficiently process input images with different recognition difficulties. We cascade multiple models from our compression framework and adaptively utilize different models and resolutions for different images. In this way, we effectively adjust the computational cost for different input images, reducing the overall computational cost without compromising the final accuracy.
- 4) We seamlessly integrate the dynamic image cropping, compound shrinking, and dynamic inference into a deep compression framework (i.e., *EdgeCompress*) for efficient deep learning inference, which can optimally adapt the model cost to meet different resource constraints of embedded hardware while maximizing model accuracy.

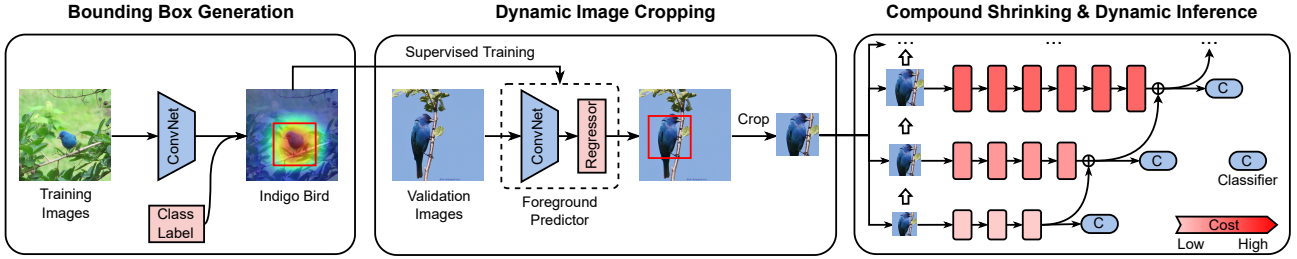


Fig. 2. The overview of the proposed *EdgeCompress* framework, which mainly consists of four components: bounding box generation (BBG), dynamic image cropping (DIC), compound shrinking (CS), and dynamic inference (DI).

Extensive experiments demonstrate the advantages of the proposed *EdgeCompress* over other SOTA model compression approaches. Specifically, *EdgeCompress* reduces the MACs of ResNet-50 by 48.8% while improving the top-1 accuracy by 0.8% on ImageNet-1K. Moreover, compared to the SOTA compression framework, HRank [11], *EdgeCompress* also achieves 4.1% higher accuracy with similar model MACs.

II. RELATED WORK

EdgeCompress makes innovative contributions mainly in three areas: 1) object localization, 2) CNN compression, and 3) dynamic neural networks. Therefore, we discuss the related works in this section.

Object localization: Object localization algorithms focus on efficiently detecting the location of foreground objects in an image, which can be mainly divided into supervised object localization (SOL) and weakly supervised object localization (WSOL). SOL [18]–[21] achieves promising accuracy with the help of well annotated datasets like COCO [2] and Pascal VOC [22]. However, the difficulties in building larger detection datasets hinder the further development of SOL. WSOL [23]–[27] can coarsely localize objects of interest with only image-level labels, which makes WSOL applicable to more large-scale datasets without position annotations, such as ImageNet-1K (also known as ImageNet or ILSVRC-2012) [1]. Specifically, CAM [25] and Grad-CAM [26] utilize a well-trained CNN to quantify the importance of each pixel of an image and determine the position of the most contributing part accordingly, i.e., the foreground. Further, ACoL [27] presents a novel CNN with two branches to adversarially learn the full region of objects, improving the localization accuracy. However, the huge computational cost and latency make both the SOL (e.g., SSD) and WSOL inapplicable in our context, i.e., resource-constrained embedded hardware.

CNN compression: Computational redundancy widely exists in CNNs [28]. To achieve a better trade-off between model accuracy and execution efficiency, effort have been made to reduce the redundancy from different dimensions of CNNs. Specifically, depth pruning [15], [29] devotes to compressing layer-level redundancy, which removes the entire layer with low sensitivity. Channel pruning [8], [11], [13], [14], [16] conducts pruning at a finer granularity, which builds compact CNNs by removing unimportant channels from each layer, which reduces the computation and memory footprint of

CNNs. Among channel pruning approaches, MobileNetV2 [8] and Slimmable networks [13] remove channels from all layers uniformly, while Taylor pruning [14], HRank [11], and DECORE [16] evaluate the global importance of each channel and then prune channels in a layer-wise manner. Both depth pruning and channel pruning focus on compressing the network architecture, while resolution pruning [8]–[10], [17] optimizes the spatial redundancy in input images by shrinking images to smaller resolutions or selectively cropping images for inference. MNasNet [9] reduces the spatial redundancy by utilizing a fixed small resolution for all images during inference. Instead, DR-ResNet [10] introduces a dynamic resolution strategy to dynamically assign different resolutions to different images according to their recognition complexity. Moreover, GFNet [17] introduces a Glance-and-Focus inference strategy, which utilizes both the shrunk version and small local patches of an image for inference to accelerate CNNs while preserving high accuracy. However, all above methods only consider reducing the redundancy in a single dimension and thus only achieve a limited compression rate. In contrast, jointly compressing all dimensions promises a better trade-off between the compression rate and accuracy.

Dynamic neural networks: To efficiently process input images corresponding to diverse classification difficulties and reduce the computational redundancy, dynamic inference approaches are proposed to adaptively utilize different models or different parts of a model for different images. Early-exit networks [30]–[32] insert multiple intermediate classifiers inside the network and allow easy samples to exit at shallow layers without executing deeper layers. Different from Early-exit networks that execute layers densely, Layer-skipping networks [33]–[35] selectively skip less important intermediate layers to avoid redundant computation and optimize the execution efficiency. Channel-skipping networks [36]–[39] consider reducing redundant computation in the width dimension, which introduce channel gates to control the execution of each channel and different channels will be selectively activated for different samples. More recently, instead of dynamically changing the network architecture at runtime, resolution-level dynamic inference approaches [38], [40] are proposed to dynamically adjust the resolution of input images during inference. Specifically, easy samples are allowed to inference at a small resolution, which greatly optimizes the inference cost. In spite of the efficiency improvement achieved by above



Fig. 3. By applying different salience threshold t , we can obtain different cropped images. The larger the threshold value, the more radical the cropping.

methods, they only focus on adjusting a single dimension at runtime, which can only achieve sub-optimal efficiency and accuracy. Instead, our dynamic inference framework utilizes the models generated from our multi-dimensional compression framework, which enables multi-dimensional dynamic inference and thus achieves higher accuracy and efficiency.

III. THE PROPOSED EDGECOMPRESS FRAMEWORK

In this section, we first outline the design of *EdgeCompress* and then describe each component in detail.

As demonstrated in Fig. 2, before inference, we first utilize Grad-CAM to generate the salience map of all training images in the classification dataset \mathcal{D} , and then we generate a bounding box for each image according to the salience map and form a pseudo bounding box label set \mathcal{B} . Thereafter, we exploit the image-box pairs $\{\mathcal{D}_i, \mathcal{B}_i\}$ to train a lightweight predictor. Meanwhile, we use compound shrinking to jointly compress the three dimensions of a CNN and generate multiple CNNs with different computational complexities, which are then cascaded for dynamic inference. In inference, the input image will be first fed into the trained predictor to efficiently localize the foreground object. Thereafter, the foreground object will be cropped and sequentially sent to the CNN models generated by compound shrinking for dynamic inference. Once a confident prediction is obtained, the inference will be terminated immediately without executing subsequent models.

A. Bounding Box Generation

As aforementioned, dynamically cropping the foreground for inference is promising in reducing computation and improving classification accuracy. However, for classification datasets like ImageNet-1K, there is no out-of-the-box position annotation for the foreground object. Moreover, the position of the foreground object varies in different images, which makes it difficult to efficiently localize the foreground object.

To address this limitation, we first use Grad-CAM [26] to automatically generate the position annotations. Specifically, let the class label of the given image be c . We first perform forward inference with a well-trained CNN (e.g. ResNet-50) to obtain the prediction score p^c for class c , and then conduct backpropagation to compute the gradient of the score p^c with respect to each activation of the last convolutional layer. Thereafter, the gradients are aggregated within each channel via global average pooling. The obtained scalar for each channel can be seen as the weight of the channel, which can be calculated as follows:

$$a_k^c = \frac{1}{Z} \overbrace{\sum_i}^{\text{pooling}} \overbrace{\sum_j}^{\text{gradients}} \frac{\partial p^c}{A_{ij}^k} \quad (1)$$

TABLE I
THE IMPACT OF USING DIFFERENT SALIENCE THRESHOLDS ON PREDICTION ACCURACY. THE MODEL IS TRAINED AND EVALUATED ON IMAGE-1K. $t = 0$ MEANS USING THE ORIGINAL IMAGES WITHOUT GRAD-CAM CROPPING.

Model	#Params (M)	#MACs (B)	t	Top-1 Acc. (%)
ResNet-50	25.6	4.1	0.00 (Baseline)	76.02
			0.25	76.45
			0.50	76.88
			0.75	76.32

where a_k^c is the weight of channel k for class c , and A_{ij}^k is a single activation indexed by i and j in the 2-D feature map of channel k . With the weights of all channels determined, the salience map for class c can be obtained by computing the weighted sum of all feature maps over the channel dimension, which is formulated as:

$$L_{Grad_CAM}^c = ReLU \left(\underbrace{\sum_k a_k^c A^k}_{\text{linear combination}} \right) \quad (2)$$

where A^k is the 2-D feature map of channel k , and ReLU is used to eliminate the impact of negative activations. Finally, the obtained salience map is upsampled to the same size as the input image via bi-linear interpolation algorithm.

With the salience map generated, we then introduce a simple yet effective strategy to determine the bounding box of the foreground object. Initially, we set the box as the boundary of the image. Subsequently, we shrink the four sides of the box simultaneously, and once a side reaches our preset salience threshold t , the side is frozen. The bounding box is determined after all sides are frozen. Note that it is crucial for the final result to appropriately select the value of t . As demonstrated in Fig. 3, a too small threshold will result in residual background redundancy, while a too large threshold will lose some important features. Therefore, we conduct empirical experiments to determine the optimal threshold value. As shown in Table I, we achieve the highest accuracy when the threshold t is set to 0.5. Therefore, we set $t = 0.5$ in our experiments. Note that more fine-grained searching for t may further improve the accuracy, but it also increases the search cost. Finally, the generated box annotations are saved in the form of $[X_{min}, Y_{min}, X_{max}, Y_{max}]$, which denotes the boundary of the foreground in the image.

B. Dynamic Image Cropping

Fig. 4 shows that we are capable of accurately localizing the foreground of images with Grad-CAM. However, Grad-CAM cannot be directly applied to edge applications because of the time-consuming backpropagation process. Moreover, Grad-CAM requires the class label as weak supervision, which is unavailable for validation images. To address these issues, we design a foreground predictor to efficiently localize the foreground of input images.

1) *Predictor Architecture*: Existing detection models, such as Faster R-CNN [20], are mainly proposed for object detection tasks (e.g., MS COCO [2]), which usually contain a large number of parameters and computation to accurately localize



Fig. 4. The bounding boxes generated with the saliency threshold $t = 0.5$, which accurately localize the key object in each image.

and identify the multiple objects in each input image. However, we focus on classification tasks, where each input image contains only one object and thus the localization difficulty is much lower than in detection tasks. Moreover, to achieve dynamic cropping, we only need to output the position of the foreground without predicting its label. Consequently, existing detection models become redundant and inefficient in our context. To this end, we design a novel lightweight foreground predictor to efficiently localize the unique foreground object of each input image. The details of the proposed foreground predictor is summarized in Table II, which consists of several residual bottleneck blocks [41] and a fully connected layer. A residual bottleneck contains two convolutional layers with 1×1 kernels and one convolutional layer with 3×3 kernels in the middle. The computational cost mainly results from the 3×3 convolutional layer. Therefore, to reduce the cost and accelerate the predictor, we only stack two residual bottleneck blocks in each stage and each block is only equipped with a small number of channels. Consequently, the proposed predictor only contains 0.27M parameters and 0.09B MACs, which is negligible compared to popular object detectors (e.g., Faster R-CNN with 134.7M (499 \times) parameters and 15.1B (167.8 \times) MACs [42]).

2) *Training of Foreground Predictor*: We train the predictor in a supervised manner. First, we generate a bounding box label set \mathcal{B} for all training images as described in Subsection III-A, then the labels are utilized to train the predictor. We use the mean square error (MSE) as the loss function. Let $\mathcal{P}_i = [X_{min}^p, Y_{min}^p, X_{max}^p, Y_{max}^p]$ be the output of the predictor, and $\mathcal{G}_i = [X_{min}^g, Y_{min}^g, X_{max}^g, Y_{max}^g]$ be the generated box label, the loss function can be formulated as:

$$\begin{aligned} \mathcal{L}_{box} &= MSELoss(\mathcal{P}_i, \mathcal{G}_i) \\ &= \frac{1}{4}((X_{min}^g - X_{min}^p)^2 + (Y_{min}^g - Y_{min}^p)^2 \\ &\quad + (X_{max}^g - X_{max}^p)^2 + (Y_{max}^g - Y_{max}^p)^2) \end{aligned} \quad (3)$$

To balance the training overhead and prediction accuracy, we train the predictor with Adam [43] optimizer for 40 epochs. The initial learning rate is set to 1e-3, and the learning rate is scheduled using exponential decay [44]. The training of the box predictor is decoupled with backbone networks. Once the predictor is trained, it can be directly applied to different classification backbones without any training overhead. During inference, the trained predictor will quickly localize the foreground object of the input image and generate a finely

TABLE II
THE ARCHITECTURE OF THE PROPOSED BOX PREDICTOR. #C DENOTES THE NUMBER OF CHANNELS AND #L DENOTES THE NUMBER OF LAYERS.

Stage	Block	Resolution	#C	#L
1	Conv 3×3	224×224	16	1
2	Residual Bottleneck	112×112	16	2
3	Residual Bottleneck	56×56	32	2
4	Residual Bottleneck	28×28	32	2
5	Residual Bottleneck	14×14	64	2
6	Pooling & Linear	7×7	4	1

#Params: 0.27M
#MACs: 0.09B

cropped image, which significantly reduces the redundancy in the input image.

C. Compound Shrinking

The proposed DIC significantly reduces the redundancy in images, improving the computational efficiency. We observe that redundancy also exists in network architectures (e.g., redundant parameters), and only removing the redundancy in images loses the opportunity to further compress the model for embedded hardware. Besides, [3] demonstrates that jointly adjusting different dimensions promises higher accuracy. To this end, we propose a compound shrinking (CS) strategy to jointly compress the three dimensions (depth, width, resolution) of CNNs to further reduce the redundancy in images as well as networks while maintaining the accuracy.

Intuitively, shrinking different dimensions has different impacts on accuracy and model overhead. The core of our compound shrinking strategy is to calculate a shrinking coefficient for each dimension according to their trade-off between accuracy and model overhead. A larger coefficient denotes more radical shrinking. More specifically, the dimension with a steep accuracy drop during shrinking will be assigned a small shrinking coefficient to prevent severe accuracy degradation. To calculate the shrinking coefficients, we first quantify the trade-off of each dimension between accuracy and model overhead. Here we use MACs as the metric to measure the cost of models, because all three dimensions are related to the MACs of a model while only the depth and width can affect the model parameters. Given a MACs budget \mathcal{M} , we first obtain the accuracy drops resulting from separately shrinking different dimensions, which can be represented as:

$$\Delta A_s(\mathcal{M}) = A_0 - A_s(\mathcal{M}) \quad (4)$$

where $s \in \{d, w, r\}$ represents the shrunk dimension, $A_s(\mathcal{M})$ denotes the accuracy of the shrunk model, and A_0 is the accuracy of the original model. To comply with the rule that the steeper the drop in accuracy, the smaller the coefficient of the corresponding dimension, we design the following equation to determine the shrinking coefficient for each dimension:

$$\mathcal{C}_s(\mathcal{M}) = \frac{\sqrt[3]{\Delta A_d(\mathcal{M}) \cdot \Delta A_w(\mathcal{M}) \cdot \Delta A_r(\mathcal{M})}}{\Delta A_s(\mathcal{M})} \quad (5)$$

where $\mathcal{C}_s(\mathcal{M})$ denotes the shrinking coefficient of the dimension s ($s \in \{d, w, r\}$). Through Equation 4 and Equation 5, we are able to efficiently calculate the coefficients once we

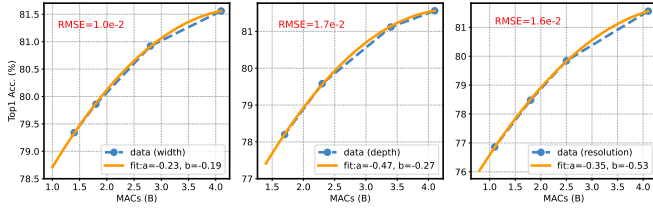


Fig. 5. The actual accuracy (blue dotted line) and the estimated accuracy (yellow line) over MACs by separately shrinking the three dimensions. The low root mean square error (RMSE) indicates that the accuracy estimator can well fit the sampled data.

obtain the accuracy degradation of the three dimensions in the given MACs regime.

However, the training cost of the compressed models to calculate the accuracy drop is still non-negligible. To mitigate the training overhead, we propose a dimension-wise accuracy estimator to quickly estimate the accuracy of the compressed models and calculate the accuracy degradation resulting from shrinking different dimensions in the given MACs regime. First, we sample a couple of models with different MACs by separately shrinking the three dimensions. As demonstrated in Fig. 5, the accuracy distribution of the three dimensions along MACs can be well fitted by a quadratic polynomial. Therefore, we design a simple yet effective polynomial estimator to predict the accuracy with respect to the target MACs \mathcal{M} . The estimator is formulated as follows:

$$A_s(\mathcal{M}) = a_s(\mathcal{M} - \mathcal{M}_0)^2 + b_s(\mathcal{M} - \mathcal{M}_0) + A_0 \quad (6)$$

where \mathcal{M}_0 is the MACs of the original model. a_s and b_s are the hyperparameters to fit for dimension s ($s \in \{d, w, r\}$). Subsequently, we train the dimension-wise estimator using least square regression with the aforementioned sampled data. Fig. 5 shows that the proposed estimator can well fit existing data. Due to the simple and intuitive design of the estimator, we only need to sample and train very few models to train the estimator, and this cost is a one-time cost. With the accuracy estimator established, we are able to quickly estimate the accuracy drop and then calculate the optimal shrinking coefficients for the three dimensions under any given resource constraint. According to the coefficients, we will jointly compress the three dimensions of the baseline network and generate a compact model with optimized efficiency. As the compressed model can be viewed as a subset of the baseline network, we call the compressed model a sub-network.

D. Dynamic Inference

Through dynamic image cropping and compound shrinking, we can optimally compress a CNN model to different complexities to satisfy various resource constraints in edge environments. Given an embedded device, an intuitive deployment strategy is to select a single model that best fits the hardware capabilities (e.g., memory capacity, computing power) to balance the trade-off between accuracy and execution efficiency. However, as different images correspond to distinct recognition difficulties [10], [40], using a single model for all images may over-process simple images and waste resources, while for complex images, the model may

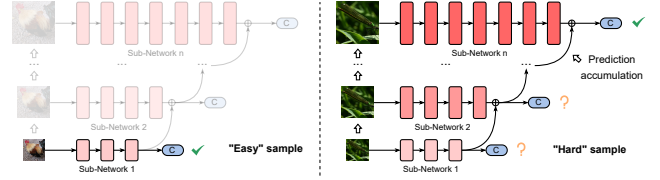


Fig. 6. The proposed dynamic inference framework, which utilizes multiple sub-networks to achieve instance-aware inference. These sub-networks are obtained by compressing the baseline network using compound shrinking.

TABLE III

THE SPECIFICATIONS OF THE SUB-NETWORKS GENERATED BY THE COMPOUND SHRINKING STRATEGY. THE BASELINE NETWORK IS RESNET-50. THE ACCURACY IS MEASURED ON IMAGENET-1K AND THE LATENCY IS MEASURED ON JETSON NANO.

Sub-network Id	#Params (M)	#MACs (B)	Latency (ms)	Top1 Acc. (%)
1	11.65	1.21	27.15	73.86
2	11.79	1.30	28.05	74.21
3	13.53	1.84	41.62	75.56
4	15.40	2.40	49.41	76.32
5	20.30	3.22	56.01	76.81
6	25.90	4.20	57.09	77.20

under-process them and generate wrong predictions, leading to a sub-optimal trade-off between accuracy and efficiency.

To address this problem, we propose a dynamic inference strategy to further optimize the run-time efficiency of CNNs on embedded devices without sacrificing accuracy. As demonstrated in Fig. 6, we first apply different MACs constraints to the compound shrinking strategy to generate multiple sub-networks with different accuracy and overhead. The specifications of all sub-networks are summarized in TABLE III. Thereafter, we deploy the generated sub-networks onto the target hardware before inference and dynamically activate different sub-networks at runtime for better accuracy and efficiency. For easy samples with a distinct foreground, maybe only the smallest sub-network will be activated to efficiently generate the correct prediction, while for hard samples with which small models are unable to produce a confident prediction, larger sub-networks will be gradually activated until a confident prediction is obtained. By doing so, we can avoid unnecessary computation and resource consumption for simple images, improving inference efficiency.

1) *Termination Condition*: Modern large-scale datasets for image classification usually contain millions of images. For example, ImageNet-1K has about 1.3 million images. It is non-trivial to determine when to terminate the inference for each image. Current dynamic inference approaches, such as multi-scale inference [31] and early-exit networks [45], [46], exploit the highest prediction probability among all classes as the prediction confidence to control the termination of dynamic inference. Given a CNN \mathcal{N} and an image x , the prediction confidence of existing methods can be represented as:

$$\begin{aligned} \mathcal{I}_{\mathcal{N}} &= \max(\text{Softmax}(\mathcal{N}(x))) \\ &= \max\left(\frac{e^{z_i}}{\sum_{j=1}^K e^{z_j}}\right) \quad \text{for } i = 1, 2, \dots, K \end{aligned} \quad (7)$$

where z_i denotes the i -th logit (i.e., the i -th output of the fully connected layer) of \mathcal{N} , which is transformed into the predic-

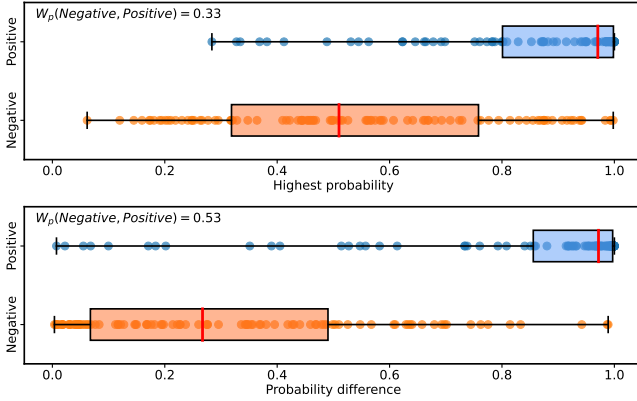


Fig. 7. The distributions of negative results and positive results along different confidence metrics. W_p denotes the Wasserstein distance between negative results and positive results. The larger the value of W_p , the more distinct the two distributions, such that our dynamic inference framework can more accurately determine whether the sample is correctly classified.

TABLE IV

COMPARISON OF DIFFERENT CONFIDENCE METRICS IN TERMS OF THE TRADE-OFF BETWEEN MODEL COMPLEXITY AND ACCURACY. THE ACCURACY IS MEASURED ON IMAGENET-1K.

Architecture	Metric	#MACs (B)	Top-1 Acc. (%)
ResNet-50	Highest probability	2.07	76.44
	Probability difference	2.07	76.68
	Highest probability	2.39	76.91
	Probability difference	2.33	77.07

tion probability for the i -th class with the Softmax function. With Equation 7, the prediction confidence can be efficiently calculated using the output of the network. Generally, if a high prediction confidence score is obtained from the current model, then the image is considered correctly classified and the inference will be terminated immediately.

In this paper, we rethink the efficacy of the confidence metric. First, we randomly sample 100 negative prediction results and 100 positive prediction results from a well-trained model. Subsequently, we summarize the distribution of the sampled data along the highest prediction probability in Fig. 7 and exploit Wasserstein distance to quantify the similarity between the distributions of positive samples and negative samples. The smaller the Wasserstein distance between two distributions, the more similar the two distributions are. As shown in the upper figure of Fig. 7, the negative samples and positive samples are distributed close along the highest probability with a small Wasserstein distance, which reveals that this confidence metric fails to effectively separate the positive predictions and negative predictions of a model, degrading the efficacy of dynamic inference. To address the issue, we introduce a novel metric, the probability difference, to control the termination of dynamic inference. The probability difference is defined as the difference between the highest prediction probability and the second highest probability, which is formulated as:

$$\mathcal{D}_N = \mathcal{I}_N - \mathcal{I}'_N \quad (8)$$

where \mathcal{I}'_N represents the second highest prediction probability. Equation 8 reveals that, unlike existing confidence metric

which only focuses on the highest prediction probability, the proposed confidence metric considers both the highest prediction itself and its advantages over other competitors. The distributions of the negative samples and positive samples along the probability difference are demonstrated in the lower figure of Fig. 7, where we observe that the two distributions are more distinct and the Wasserstein distance between them is much larger compared to existing confidence metric (i.e., the highest probability), which indicates that the proposed confidence metric is able to estimate the correctness of a prediction more accurately, enabling effective control over the dynamic inference. After evaluating the confidence of a prediction with the proposed metric, we will compare the evaluation result with a preset threshold value \mathcal{D}_0 . If the evaluation result is larger than the preset threshold value, the prediction is considered confident and the inference will be terminated. Otherwise, the image will be sent to a larger model for more accurate prediction. By changing the threshold value \mathcal{D}_0 , we are able to flexibly adjust the trade-off of the dynamic inference between inference overhead and accuracy. Specifically, a higher threshold value will force more images to flow to large models, and thus the accuracy will be improved at the cost of higher inference costs. On the contrary, reducing the threshold value will allow more images to exit at small models, thereby saving the inference overhead. To validate the proposed confidence metric, we perform experiments on ImageNet-1K and present the results in TABLE IV, where we observe that the proposed metric remarkably improves accuracy without sacrificing the computational cost.

2) *Prediction Accumulation*: During dynamic inference, hard samples may flow through multiple models. Some approaches directly adopt the output of the last model as the final result [30], which wastes the information from the previously executed models and consequently loses the opportunity to further improve accuracy. Instead, some other methods propose to utilize the information of previous models by merging the feature maps from previous models into the current model for higher accuracy [40]. However, the fusion of feature maps of different models introduces additional computational overhead, reducing the efficiency of dynamic inference.

To address the above concerns, we propose prediction accumulation to effectively utilize the information from different models for higher accuracy. Different from the fusion of feature maps [40] which requires a large amount of additional computation, we efficiently integrate the information from different models without compromising the computational overhead by accumulating the output of the last fully connected layer in each model (i.e., the logits), which is formulated as:

$$\mathcal{Z}'_i = \alpha \mathcal{Z}_i + \mathcal{Z}'_{i-1} \quad (9)$$

where \mathcal{Z}_i denotes the logits of the current model, and \mathcal{Z}'_i represents the accumulated logits of the current model, which will be used to calculate the prediction of the current model. \mathcal{Z}'_{i-1} is the accumulated logits of the previous model and α is a hyperparameter to control the contribution of the prediction of the current model (i.e., \mathcal{Z}_i). The value of α can affect the final accuracy of dynamic inference obviously. To identify

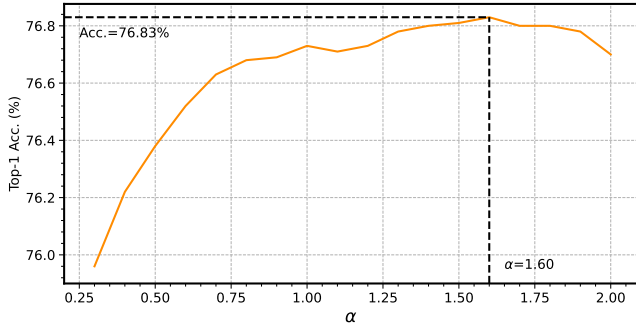


Fig. 8. The impact of the value of α on the final accuracy of dynamic inference. We observe the highest accuracy at $\alpha = 1.60$, and thus we fix $\alpha = 1.60$ for subsequent experiments. The target dataset is ImageNet-1K.

TABLE V

THE IMPACT OF OUR PREDICTION ACCUMULATION STRATEGY ON MODEL COMPUTATION, INFERENCE LATENCY, AND ACCURACY. THE INFERENCE LATENCY IS MEASURED ON JETSON XAVIER, AND THE ACCURACY IS MEASURED ON THE IMAGENET-1K DATASET.

Architecture	Accumulation	#MACs (B)	Latency (ms)	Top-1 Acc. (%)
ResNet-50	✗	2.07	7.84	76.68
	✓	2.07	7.91	76.83
	✗	2.33	8.63	77.07
	✓	2.33	8.60	77.35

the optimal value of α , we sample multiple values of α and summarize the relationship between accuracy and α in Fig. 8, where we observe the highest accuracy when $\alpha = 1.60$. Therefore, we fix $\alpha = 1.60$ in our experiments.

The logits of a model can directly determine the prediction results, and thus accumulating logits can effectively utilize the information from multiple models for higher accuracy. The overhead of accumulating logits is determined by the number of logits in each model and the number of models to accumulate. Specifically, the computational cost of logits accumulation can be calculated as follows:

$$Q_{acc} = n_l \cdot (n_m - 1) \quad (10)$$

where n_l is the number of logits and n_m is the number of models. For example, for two models with 1,000 logits, the computational cost of logits accumulation will be $1,000 \times (2 - 1) = 1,000$ FLOPs, which can be neglected compared to the inference overhead of CNN backbones (e.g., ResNet-50 with 4.1 Billion FLOPs). As shown in the experimental results in TABLE V, the accumulation strategy improves the accuracy remarkably while not increasing the computational cost and inference latency of our framework.

3) *Dynamic Inference Algorithm*: We demonstrate our dynamic inference algorithm in detail in Algorithm 1. Given a series of CNN models ordered from low to high computational complexity and an input image, we initiate the dynamic inference with the smallest model and gradually activate models with higher computational complexity. For each model, we first perform inference with the model to obtain its prediction logits, then, we accumulate the logits of this model with all previously executed models as Equation 9. Subsequently, the proposed termination metric of the current

Algorithm 1: Dynamic Inference Algorithm

Data: CNN models $\{\mathcal{N}_1, \mathcal{N}_2, \dots, \mathcal{N}_n\}$, input image x , termination threshold D_0

Result: Prediction result o

```

i ← 0;
while i < n do
   $\mathcal{Z}_i \leftarrow \mathcal{N}_i(x)$  // Inference with the current model
  if i = 0 then
     $\mathcal{Z}'_i \leftarrow \mathcal{Z}_i$ 
  else
     $\mathcal{Z}'_i \leftarrow \text{AccumulateLogits}(\mathcal{Z}_i, \mathcal{Z}'_{i-1})$  // See Eq. 9
  end
  // Calculate the termination metric, see Eq. 8
   $\mathcal{D}_{\mathcal{N}_i} \leftarrow \text{CalculateMetric}(\mathcal{N}_i, \mathcal{Z}'_i)$ 
  if  $\mathcal{D}_{\mathcal{N}_i} > D_0$  then
    Break // Terminate dynamic inference
  end
  i ← i + 1 // Activate the next model
end
 $o \leftarrow \text{Softmax}(\mathcal{Z}'_i)$  // Calculate the final result

```

model is calculated using the accumulated logits according to Equation 8, which is then compared with the preset threshold D_0 . If the calculated metric is larger than the threshold, the predicted result is considered confident and dynamic inference will be terminated immediately. Otherwise, a larger model will be activated for inference. In practice, we observe that, for most images, dynamic inference is able to produce a confident prediction and be terminated at the smallest model. In this case, the overall latency of dynamic inference is equal to the inference latency of the smallest model. Consequently, we avoid using large models for most images, saving computation and reducing latency significantly compared to static inference. The results are presented in the Experimental Results section.

IV. EXPERIMENTAL RESULTS

In this section, we perform extensive experiments on different benchmarks to validate the efficacy of *EdgeCompress* and demonstrate its advantages over existing SOTA approaches in terms of accuracy, computational complexity (i.e., MACs), and run-time efficiency. Further, we conduct ablation study to show the contribution of each component in our framework.

TABLE VI

HARDWARE SPECIFICATIONS OF THREE PLATFORMS. THE COLUMN “#CORES” DENOTES THE NUMBER OF CUDA CORES AND CPU CORES FOR GPU PLATFORMS (I.E., AGX XAVIER AND JETSON NANO) AND THE CPU PLATFORM (I7-9750H), RESPECTIVELY.

Device	Power	Memory	#Cores	Core Freq.	Performance
AGX Xavier	15 W	32 GB	512	900 MHz	11.0 TOPS
Jetson Nano	5 W	4 GB	128	992 MHz	0.5 TOPS
i7-9750H	45 W	16 GB	6	2600 MHz	0.4 TOPS

A. Hardware Devices

To validate the run-time efficiency of *EdgeCompress*, including inference latency and throughput, we select two representative embedded GPU platforms, NVIDIA AGX Xavier and Jeton Nano, and Intel i7-9750H@2.6GHz CPU to deploy different methods and compare their performance. The specifications of selected devices are shown in TABLE VI.

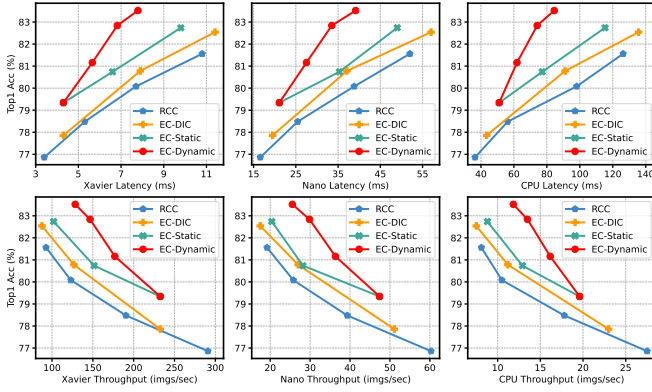


Fig. 9. The real performance of ResNet-50 compressed by different methods on three distinct hardware devices. Accuracy is measured on ImageNet-100.

B. Datasets

We validate the proposed *EdgeCompress* on four representative datasets: 1) CIFAR-10, 2) CIFAR-100, 3) ImageNet-100, and 4) ImageNet-1K [1]. ImageNet-1K (also known as ImageNet or ILSVRC-2012) is one of the most popular large-scale datasets for image classification, which includes 1,000 classes. ImageNet-100 is a subset of ImageNet-1K, which consists of 100 classes randomly selected from ImageNet-1K. The details of ImageNet-100 can be found in the code repository. All images are preprocessed following a simple configuration as [4].

C. Networks

We apply our *EdgeCompress* framework to three widely utilized CNN backbones, VGG16_BN [47], ResNet-50 [41], and RegNet-X [4]. For each model, we employ *EdgeCompress* to remove the spatial redundancy in input images and the architecture redundancy in networks, thereby reducing the computational cost and improving the inference efficiency. As a comparison, we also report the results of other methods.

D. Optimization Settings

All models in our experiments are trained using SGD optimizer with a momentum of 0.9. We first train models for 100 epochs without using dynamic image cropping, where the first 5 epochs are for warmup. For experiments on ImageNet-100 and ImageNet-1K, the learning rate is set to 2.0, which will be decayed by exponential learning rate policy with a decay factor of 0.02. The training batch size is set to 1024. Subsequently, the proposed dynamic image cropping is utilized to fine-tune the pretrained models for 20 epochs. The learning rate for fine-tuning is $5e-4$. In addition, we also use label smoothing with the smoothing factor $\epsilon = 0.1$ [48] to prevent overfitting. For experiments on CIFAR-10 and CIFAR-100, the initial learning rate is 0.1 and the training batch size is 128.

E. Evaluation Methodology

In this paper, we propose three novel approaches to comprehensively reduce the computational cost of CNNs. Thanks to the flexible design of these approaches, they can be used

TABLE VII
RESULTS OF RESNET-50 ON IMAGENET-100. RCC-BASELINE REPRESENTS THE BASELINE RESNET-50 MODEL, WHERE WE CROP AND RESIZE ALL IMAGES TO THE SIZE 224×224 WITH RCC.

Method	#Params (M)	#MACs (B)	↓ MACs (%)	Top1 Acc. (%)
RCC-Baseline	23.7	4.1	0.0	81.6
EC-DIC	24.0	4.2	-2.4	82.5
EC-Static	17.3	3.0	26.8	82.7
EC-Dynamic	13.6	2.0	51.2	83.5
RCC	23.7	3.0	26.8	80.1
EC-DIC	24.0	2.6	36.6	80.8
EC-Static	14.5	2.4	41.5	81.5
EC-Dynamic	11.8	1.7	58.5	82.8
RCC	23.7	1.1	73.2	76.9
EC-DIC	24.0	1.2	70.7	77.9
EC-Static	7.8	1.0	75.6	79.3
EC-Dynamic	8.9	1.2	70.7	80.2

separately or coupled for a higher compression ratio. To better demonstrate the flexibility and efficacy of our design, we evaluate different combinations of the three approaches. Specifically, EC-DIC represents that only the dynamic image cropping component is exploited, while EC-Static denotes both the dynamic image cropping and compound shrinking are adopted. Finally, EC-Dynamic denotes the completed framework that contains all three components, which further integrates the dynamic inference approach based on EC-Static.

F. Results on ImageNet-100

We conduct experiments on ImageNet-100 with ResNet-50 and RegNet-X, where we use different methods mentioned in Subsection IV-E to compress models to different complexities. As a comparison, we use the most popular image cropping method, ResizedCenterCrop (RCC) to crop and resize images to different sizes. Finally, all models are deployed to selected hardware to evaluate their latency and throughput.

1) *ResNet-50*: As shown in TABLE VII, all of our approaches outperform the competitor (i.e., RCC) in terms of the model complexity, on-device execution efficiency, and accuracy. Specifically, compared to the baseline ResNet-50 (RCC-Baseline), EC-DIC improves the accuracy by 0.9% with a negligible increase in model parameters (1.2%) and MACs (2.4%), while EC-Static further pushes up the accuracy improvement to 1.1% with a parameter reduction of 27.0% and a MACs reduction of 26.8%. Finally, EC-Dynamic achieves the best performance, which compresses the MACs by 51.2% while still improving the accuracy by 1.9% compared to RCC-Baseline. In the low complexity regime, EC-Dynamic achieves 3.3% higher accuracy than RCC with only 37.6% model parameters (8.9M v.s. 23.7M) and similar MACs. Meanwhile, Fig. 9 indicates that all of our methods outperform RCC by a large margin across a wide spectrum of inference latency and throughput on different resource-constrained embedded devices. Particularly, EC-Dynamic achieves 83.5% top-1 accuracy with a latency of 7.8 ms on Xavier, which is 1.9% higher in accuracy and 27.7% lower in latency compared to RCC (81.6% top1 accuracy, 10.8 ms). At the same time, the throughput of EC-Dynamic on Xavier is 128.4 imgs/sec , which is 38.5% higher than RCC (92.7 imgs/sec). On Nano and Intel i7-9750H CPU, EC-Dynamic also improves the throughput

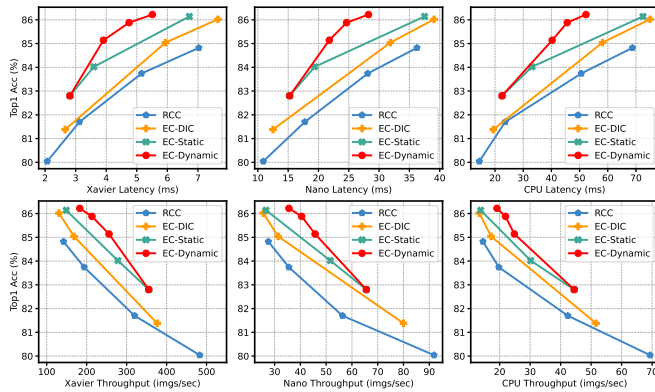


Fig. 10. The real performance of RegNet-X compressed by different methods on three distinct hardware devices. Accuracy is measured on ImageNet-100.

TABLE VIII

RESULTS OF REGNET-X ON IMAGENET-100. RCC-BASELINE REPRESENTS THE BASELINE REGNET-X MODEL WITH ALL INPUT IMAGES CROPPED AND RESIZED TO 224×224 WITH RCC.

Method	#Params (M)	#MACs (B)	↓ MACs (%)	Top1 Acc. (%)
RCC-Baseline	8.4	1.6	0.0	84.8
EC-DIC	8.7	1.3	18.8	85.0
EC-Static	6.3	1.3	18.8	86.1
EC-Dynamic	4.4	0.8	50.0	86.2
RCC	8.4	0.7	56.3	82.3
EC-DIC	8.7	0.8	50.0	83.8
EC-Static	3.6	0.6	62.5	84.0
EC-Dynamic	3.3	0.6	62.5	85.1
RCC	8.4	0.4	75.0	80.0
EC-DIC	8.7	0.5	68.8	81.4
EC-Static	2.5	0.4	75.0	82.8
EC-Dynamic	2.8	0.5	68.8	83.7

by 33.0% and 46.2%, and reduces the latency by 24.7% and 33.1%, respectively.

2) *RegNet-X*: The experimental results of RegNet-X are summarized in TABLE VIII and Fig. 10, where we also observe a significant improvement of our method. EC-Dynamic outperforms the baseline RegNet-X (RCC-Baseline) with an improvement of 1.4% in accuracy and a reduction of 50.0% in MACs. Meanwhile, EC-Dynamic reduces the model parameters by 47.6% (4.4M v.s. 8.4M). In the low MACs regime, EC-Dynamic observes a remarkable 3.7% improvement in accuracy with only 33.3% parameters (2.8M v.s. 8.4M) compared to RCC. As for the real performance on hardware, EC-Dynamic obtains an accuracy of 86.2% with 5.5 ms latency on Xavier, which is 1.4% higher in accuracy and 21.4% lower in latency than RCC (84.8% top-1 accuracy, 7.0 ms). Similarly, the latency reductions of EC-Dynamic on Nano and Intel CPU are 22.0% and 24.0%, respectively. Besides, EC-Static also observes a 29.0% throughput improvement (35.6 *imgs/sec* v.s. 27.6 *imgs/sec*) on Nano and a 31.7% throughput improvement (19.1 *imgs/sec* v.s. 14.5 *imgs/sec*) on CPU compared to RCC.

G. Results on ImageNet-1K

In this subsection, we evaluate our approach on ImageNet-1K, and compare the evaluation results with many SOTA CNN compression frameworks. To enable a comprehensive comparison with more SOTA frameworks, we employ ResNet-50 as the baseline network. In addition, we also compare our

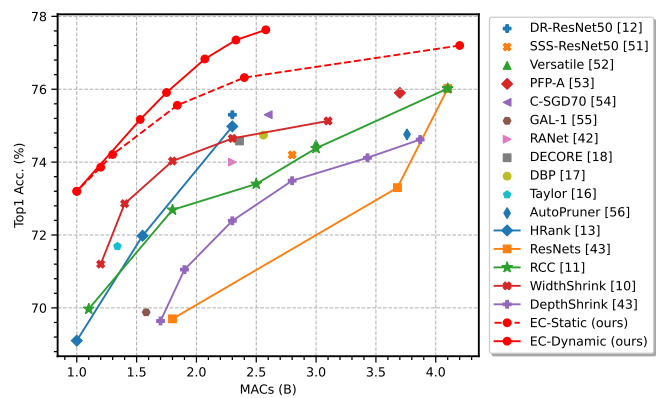


Fig. 11. Comparison of our *EdgeCompress* with other state-of-the-art model compression methods. The baseline model is ResNet-50 and the dataset is ImageNet-1K.

compressed models with many popular backbone architectures in different computation regimes.

1) *Comparison with SOTA Compression Methods*: Starting with the baseline ResNet-50 model, we implement different model shrinking methods, including resolution shrinking (RCC), width shrinking (WidthShrink) [8], [49], depth shrinking (DepthShrink) [41], EC-DIC, EC-Static, and EC-Dynamic to compress the three dimensions of the model to different MACs regimes and compare their performance. In addition, we also report the performance of multiple SOTA model compression techniques from the related papers, including DR-ResNet50 [10], SSS-ResNet50 [50], Versatile [51], PFP-A [52], C-SGD70 [53], GAL-1 [54], HRank [11], AutoPruner [56], Taylor [16], HRank [13], ResNets [43], RCC [11], WidthShrink [10], DepthShrink [43], EC-Static (ours), and EC-Dynamic (ours). The comparison results are summarized in Fig. 11, which shows that our method achieves the highest accuracy across a wide range of MACs. Particularly, compared to the baseline ResNet-50, our EC-Static achieves 1.2% accuracy improvement (76.0% to 77.2%) with a negligible increase in MACs (4.1B to 4.2B). Moreover, EC-Dynamic further improves the accuracy to 77.6% with only 2.6B MACs, which is 1.6% higher in accuracy and 36.6% lower in MACs compared to the baseline ResNet-50. As we continue to reduce the MACs budget, EC-Dynamic reduces the MACs by 48.8% (4.1B to 2.1B) while still achieving 0.8% higher accuracy (76.0% to 76.8%). In the lowest MACs regime, EC-Dynamic and EC-Static achieve similar trade-offs between model MACs and accuracy, both of which remarkably improve the accuracy by 4.2% (70.0% to 74.2%) compared to RCC with similar MACs. In comparison with other SOTA compression methods, our method also achieves the best trade-off between MACs and accuracy. For example, EC-Dynamic achieves 5.3% higher accuracy (75.2% v.s. 69.9%) than GAL-1 [54] with less MACs (1.5B v.s. 1.6B).

2) *Comparison with Popular Backbones*: In this experiment, we compare our results on ResNet-50 with other models from the ResNet family, such as ResNet-101 and ResNet-34, etc. In addition, we also conduct extensive comparisons with other popular backbones like DenseNets [56] and the Inception family [48], [57]. As demonstrated in TABLE IX, in the highest MACs regime, EC-Dynamic uses 67.1% less MACs (2.6B v.s. 7.9B) to achieve 0.5% higher accuracy

TABLE IX
COMPARISON WITH OTHER POPULAR BACKBONE NETWORKS.

Model	#Params (M)	#MACs (B)	↓ MACs (%)	Top1 Acc. (%)
ResNet-50 [41]	25.6	4.1	0.0	76.0
ResNet-101 [41]	44.6	7.9	-92.7	77.4
DenseNet-161 [56]	28.7	7.9	-92.7	77.1
InceptionV3 [48]	27.2	5.8	-41.5	77.3
EC-DIC	25.9	4.2	-2.4	77.2
EC-Dynamic	20.4	2.6	36.6	77.6
ResNet34 [41]	21.8	3.7	9.8	73.3
DenseNet-169 [56]	14.2	3.4	17.1	75.6
EC-DIC	25.9	3.1	24.4	76.3
EC-Static	15.4	2.4	41.5	76.3
EC-Dynamic	17.1	2.1	48.8	76.8
ResNet-18 [41]	11.7	1.8	56.1	69.8
DenseNet-121 [56]	8.0	2.9	29.3	74.6
BN-Inception [57]	11.2	2.1	48.8	73.5
EC-DIC	25.9	1.9	53.7	74.9
EC-Static	13.5	1.8	56.1	75.6
EC-Dynamic	15.1	1.8	56.1	75.9

TABLE X
COMPARISON WITH OTHER DYNAMIC INFERENCE FRAMEWORKS. {D, W, R} DENOTE THE DIMENSIONS INVOLVED FOR DYNAMIC INFERENCE.

Method	d	w	r	#MACs (B)	Top-1 Acc. (%)
ResNet-50 [41]				4.1	76.0
SkipNet [34]	✓			3.6	76.2
ConvNet-AIG [35]	✓			3.1	76.2
Channel Selection [39]		✓		2.5	76.2
DR-ResNet [10]			✓	3.2	77.0
EC-Dynamic (ours)	✓	✓	✓	2.6	77.6
RANet [40]	✓		✓	2.0	75.2
ConvNet-AIG [35]	✓			2.6	75.3
DR-ResNet [10]			✓	2.3	75.3
MSDNet [31]	✓			2.1	75.7
Channel Selection [39]		✓		2.3	76.1
EC-Dynamic (ours)	✓	✓	✓	2.1	76.8

than DenseNet-161, while in the lowest MACs regime, EC-Dynamic also obtains the highest top-1 accuracy (75.9%), which is 6.1% and 2.4% higher than ResNet-18 (69.8%) and BN-Inception (73.5%), respectively. The comparison results with other backbones reveal that our method can achieve promising results without redesigning the network architecture, which avoids the extremely time-consuming exploration of the architecture design space.

3) *Comparison with SOTA Dynamic Inference Frameworks:* We can observe from the above experiments that our compression framework with dynamic inference (i.e., EC-Dynamic) surpasses the one without dynamic inference (i.e., EC-Static) in model efficiency and accuracy. To further validate the advantages of our dynamic inference framework, we compare it with multiple SOTA dynamic inference frameworks. The comparison results are shown in TABLE X, from which we observe that our EC-Dynamic framework achieves significant improvements on accuracy without sacrificing model complexity compared to other approaches. It is worth noting that the most of existing dynamic inference approaches only adjust a single dimension during inference, while our approach enables the joint adaptation of the three dimensions based on our multi-dimensional compression framework, achieving higher accuracy and efficiency. This also reveals that all components of our framework can be seamlessly coupled for a better result.

TABLE XI
STATIC INFERENCE V.S. DYNAMIC INFERENCE IN TERMS OF RUNTIME LATENCY AND ACCURACY. THE LATENCY IS REPRESENTED BY THE AVERAGE INFERENCE LATENCY OF ALL IMAGES.

Method	Threshold	Xavier (ms)	Nano (ms)	CPU (ms)	Top1 (%)
EC-Static	N.A.	8.3	41.6	81.8	75.6
EC-Dynamic	0.1	7.0	34.0	74.8	75.9
EC-Static	N.A.	9.4	49.4	96.9	76.3
EC-Dynamic	0.2	7.9	38.2	84.9	76.8
EC-Static	N.A.	10.8	56.0	117.6	76.8
EC-Dynamic	0.3	8.6	41.6	93.0	77.4
EC-Static	N.A.	11.8	57.1	134.5	77.2
EC-Dynamic	0.4	9.5	44.7	100.4	77.6

4) *On-Device Efficiency of Dynamic Inference:* In this experiment, we demonstrate the running efficiency of our approach on various edge devices and analyze how the preset threshold affects the on-device latency and accuracy. As shown in TABLE XI, we first apply a small threshold (i.e., 0.1) to our dynamic algorithm, which achieves higher accuracy and lower latency than static inference. As we increase the threshold, there are more images whose prediction confidence is smaller than the threshold, and thus more images are sent to the larger model for further inference. Consequently, both the classification accuracy and average inference latency of processing one image increase.

H. Results on CIFAR-10 and CIFAR-100

To better demonstrate the efficacy of our approach on small-scale datasets, we conduct extensive experiments on both CIFAR-10 and CIFAR-100 datasets. The experimental results are shown in TABLE XII, which indicate that our approach has significant advantages over other existing methods on both datasets. For instance, our approach observes 2.09% higher top-1 accuracy with 41.13% less computation on CIFAR-10.

TABLE XII
THE EXPERIMENTAL RESULTS ON CIFAR-10 AND CIFAR-100. THE BASELINE NETWORK IS VGG-16 WITH BATCH NORMALIZATION LAYERS. **CR** IN THE TABLE DENOTES THE COMPRESSION RATIO.

Dataset	Method	#MACs (M)	MACs CR (%)	Top1 Acc. (%)
CIFAR-10	VGG16_BN [47]	313.74	0.00	93.96
	GAL-0.1 [54]	171.89	45.21	90.73
	Hrank [11]	108.61	65.38	92.34
	EdgeCompress (ours)	101.18	67.75	92.82
	SSS [50]	183.13	41.63	93.02
	Zhao et al [58]	190.00	39.44	93.18
	Hrank [11]	145.61	53.59	93.43
CIFAR-100	EdgeCompress (ours)	120.55	61.58	93.64
	SSS [50]	223.13	28.89	71.08
	Zhao et al [58]	256.00	18.42	73.33
	EdgeCompress (ours)	206.34	34.24	73.35

I. Robustness Analysis

To evaluate the robustness of the proposed framework, we perform experiments in two long-tail settings: 1) exponential and 2) step, and compare the results with the normal setting. The experiments are conducted on CIFAR-10 with an unbalancing factor of 0.5. The experimental results are shown in TABLE XIII, where the minor accuracy degradation in long-tail settings validates the robustness of our framework.

TABLE XIII
RESULTS IN DIFFERENT LONG-TAIL SETTINGS, WHERE "NORMAL"
DENOTES THE RESULTS IN THE NORMAL SETTING. THE NETWORK
UTILIZED IS VGG16_BN AND THE DATASET IS CIFAR-10.

Long-tail settings	#Params (M)	#MACs (M)	Top1 Acc. (%)
VGG16_BN [47]	14.98	313.74	93.96
Normal	5.39	101.18	92.82
Exponential	5.92	112.12	91.69
Step	5.79	109.33	91.86
Normal	6.59	126.14	93.54
Exponential	6.90	132.54	92.42
Step	6.71	128.53	92.43

TABLE XIV
COMPARISON WITH DIFFERENT FOREGROUND PREDICTORS IN TERMS OF
THE CLASSIFICATION ACCURACY AND MODEL EFFICIENCY.

Model	#Params (M)	#MACs (B)	Latency (ms)	Top1 Acc. (%)
ResNet-18 [41]	11.23	1.81	3.01	75.59
ResNet-34 [41]	21.33	3.66	5.19	75.50
RegNet-X_800MF [4]	6.65	0.80	4.50	75.50
RegNet-X_1.6GF [4]	8.37	1.60	7.26	75.61
EfficientNet-B0 [3]	4.13	0.38	4.31	75.57
EfficientNet-B1 [3]	6.64	0.57	6.21	75.62
Ours	0.27	0.09	1.04	75.57

J. Analytical Experiments

In this subsection, we show the impact of some important hyperparameters on the final performance of our framework.

1) *The Architecture of The Foreground Predictor*: The design of the foreground predictor can significantly affect the efficiency and accuracy of our framework. To validate the proposed lightweight foreground predictor, we conduct comparison experiments by integrating different advanced CNN into our framework as the foreground predictor. The experimental results are summarized in TABLE XIV, where we observe that our design achieves the best trade-off between accuracy and efficiency. Even though some modern architectures can achieve slightly higher accuracy, they result in magnitudes higher model complexity and latency. For instance, EfficientNet-B1 only achieves a mere 0.05% accuracy improvement with $24.6\times$ parameters and $6.3\times$ MACs compared to our predictor, which significantly reduces the efficiency of the whole framework.

2) *The Number of Models*: The number of models cascaded for dynamic inference is also crucial to the final performance of our framework. We perform comprehensive experiments to identify the optimal number of models from the perspective of accuracy, computational complexity, and actual inference latency. The experimental results in Fig. 12 uncover an interesting insight that using too many models can worsen the trade-off between model efficiency and accuracy. Specifically, we observe that using two models for dynamic inference achieves the optimal trade-off between model efficiency and accuracy among all configurations. Meanwhile, the two-model configuration avoids loading too many models onto the device, optimizing the memory occupation of dynamic inference.

K. Ablation Study

Our framework contains three novel components: 1) Dynamic Image Cropping (DIC), 2) Compound Shrinking (CS),

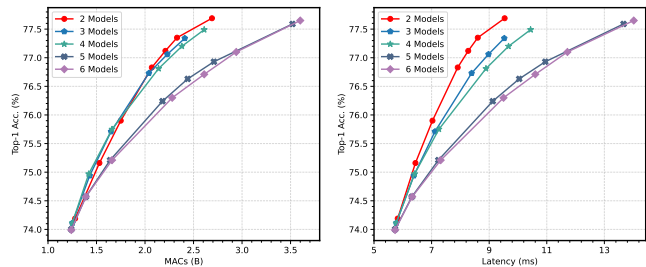


Fig. 12. The impact of the number of models used for dynamic inference on the computational complexity and on-device latency. The latency is quantified as the average latency of all images on AGX Xavier.

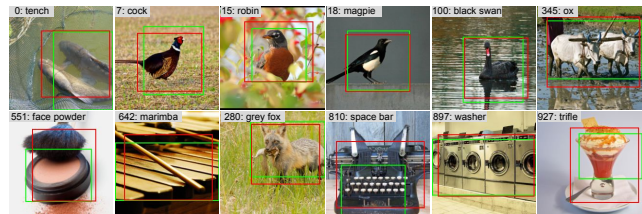


Fig. 13. Visualization of the predicted bounding boxes (red) and the ground truth bounding boxes generated from Grad-CAM (green). Our predictor achieves a high localization accuracy of 62.1% mAP on ImageNet-1K validation set. The images above are randomly selected from ImageNet-1K.

and 3) Dynamic Inference (DI). To validate the efficacy and efficiency of each component separately, we conduct ablation experiments on the ImageNet-1K dataset. The experimental results are shown in TABLE XV, where we observe that our DIC framework (i.e., EC-DIC) outperforms the ResizedCenterCrop strategy by a remarkable 1.5% accuracy improvement. Afterwards, we gradually integrate the CS module and DI strategy with DIC to build EC-Static and EC-Dynamic. The experimental results demonstrate that both EC-Static and EC-Dynamic further improve the accuracy, and the complete framework EC-Dynamic achieves the best accuracy. Thanks to the novel design of our framework, a significant improvement on accuracy is achieved at a slightly higher latency cost, optimizing the trade-off between accuracy and execution efficiency. The ablation experiments reveal that all DIC, CS, and DI components contribute to the final performance.

L. Visualization

1) *Foreground Prediction*: We visualize the bounding boxes generated from both Grad-CAM and our predictor in Fig. 13. We can see that the foreground of most images only occupies part of the whole images, thus performing inference on the whole image is unnecessary and inefficient, which coincides with our motivation. Moreover, Fig. 13 validates that, even though the lightweight foreground predictor only has very limited computation and parameters, it can still accurately and efficiently localize the main object. Due to the design of the efficient foreground predictor, we can remove the spatial redundancy in images, accelerating CNNs on edge devices.

2) *Easy Samples & Hard Samples*: We visualize some easy samples and hard samples from ImageNet to more intuitively demonstrate the difference between them. The visualization results in 14 indicate that the most of easy images have a

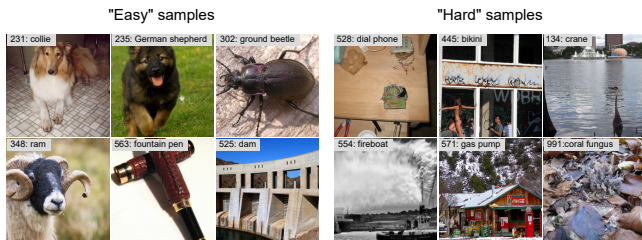


Fig. 14. Visualization of some hard samples and easy samples. Hard samples are considered as the images that cannot be confidently classified by the first model, and easy samples refer to those images that can be confidently classified by the first model. All images are from ImageNet-1K.

TABLE XV
RESULTS OF ABLATION EXPERIMENTS. THE BASELINE NETWORK IS RESNET-50 AND THE TARGET DATASET IS IMAGE-1K.

Method	#MACs (B)	Latency (ms)	Top1 Acc. (%)
ResNet-50	4.1	10.6	76.0
ResizedCenterCrop (RCC)	1.8	6.3	73.4
EC-DIC (DIC only)	1.9	7.3	74.9
EC-Static (DIC + CS)	1.8	8.3	75.6
EC-Dynamic (DIC + CS + DI)	1.8	7.0	75.9

simple and clear foreground, and thus they can be correctly recognized by a small model. For hard samples, the images are more confusing because of their unintuitive foreground, and larger models are needed to mine high-level semantics in images for the correct prediction. Through the proposed dynamic inference framework, images with different difficulties can be processed by the appropriate model, achieving higher run-time efficiency and accuracy.

V. CONCLUSION

In this paper, we propose *EdgeCompress*, a comprehensive CNN compression framework to reduce the computational redundancy in both input images and network architectures, facilitating the deployment of advanced CNNs onto embedded devices. In *EdgeCompress*, we first introduce dynamic image cropping, which effectively and efficiently removes the redundancy in input images. Subsequently, we present compound shrinking to collaboratively compress the three dimensions of CNNs, reducing the computational redundancy in both input images and network architectures. Finally, we design a dynamic inference strategy, which adaptively execute different models for different input images, further improving the inference efficiency of CNNs. Extensive experiments validate the advantages of *EdgeCompress* over existing SOTA approaches.

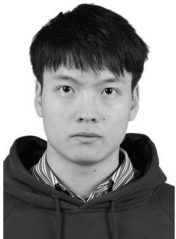
VI. ACKNOWLEDGEMENT

This study is partially supported under the RIE2020 Industry Alignment Fund – Industry Collaboration Projects (IAF-ICP) Funding Initiative, as well as cash and in-kind contribution from the industry partner, HP Inc., through the HP-NTU Digital Manufacturing Corporate Lab (I1801E0028). This work is also partially supported by the Ministry of Education, Singapore, under its Academic Research Fund Tier 2 (MOE2019-T2-1-071), and Nanyang Technological University, Singapore, under its NAP (M4082282).

REFERENCES

- [1] J. Deng *et al.*, “Imagenet: A large-scale hierarchical image database,” in *CVPR*, 2009.
- [2] T.-Y. Lin *et al.*, “Microsoft coco: Common objects in context,” in *ECCV*, 2014.
- [3] M. Tan and Q. Le, “Efficientnet: Rethinking model scaling for convolutional neural networks,” in *ICML*, 2019.
- [4] I. Radosavovic *et al.*, “Designing network design spaces,” in *CVPR*, 2020.
- [5] Z. Liu *et al.*, “A convnet for the 2020s,” in *CVPR*, 2022.
- [6] W. Shi *et al.*, “Edge computing: Vision and challenges,” *IoT Journal*, 2016.
- [7] B. Chen *et al.*, “Update compression for deep neural networks on the edge,” in *CVPR Workshops*, 2022.
- [8] M. Sandler *et al.*, “Mobilenetv2: Inverted residuals and linear bottlenecks,” in *CVPR*, 2018.
- [9] M. Tan *et al.*, “Mnasnet: Platform-aware neural architecture search for mobile,” in *CVPR*, 2019.
- [10] M. Zhu *et al.*, “Dynamic resolution network,” in *NeurIPS*, 2021.
- [11] M. Lin *et al.*, “Hrank: Filter pruning using high-rank feature map,” in *CVPR*, 2020.
- [12] Z. Wang *et al.*, “Convolutional neural network pruning with structural redundancy reduction,” in *CVPR*, 2021.
- [13] J. Yu *et al.*, “Slimmable neural networks,” in *ICLR*, 2019.
- [14] P. Molchanov *et al.*, “Importance estimation for neural network pruning,” in *CVPR*, 2019.
- [15] W. Wang *et al.*, “Dbp: Discrimination based block-level pruning for deep model acceleration,” *arXiv preprint arXiv:1912.10178*, 2019.
- [16] M. Alwani *et al.*, “Decore: Deep compression with reinforcement learning,” in *CVPR*, 2022.
- [17] Y. Wang *et al.*, “Glance and focus: A dynamic approach to reducing spatial redundancy in image classification,” in *NeurIPS*, 2020.
- [18] A. Bochkovskiy *et al.*, “Yolov4: Optimal speed and accuracy of object detection,” *arXiv preprint arXiv:2004.10934*, 2020.
- [19] W. Liu *et al.*, “Ssd: Single shot multibox detector,” in *ECCV*, 2016.
- [20] S. Ren *et al.*, “Faster r-cnn: Towards real-time object detection with region proposal networks,” in *NeurIPS*, 2015.
- [21] K. He *et al.*, “Mask r-cnn,” in *ICCV*, 2017.
- [22] M. Everingham *et al.*, “The pascal visual object classes (voc) challenge,” *IJCV*, 2010.
- [23] X.-S. Wei *et al.*, “Unsupervised object discovery and co-localization by deep descriptor transformation,” *Pattern Recognition*, 2019.
- [24] C.-L. Zhang *et al.*, “Rethinking the route towards weakly supervised object localization,” in *CVPR*, 2020.
- [25] B. Zhou *et al.*, “Learning deep features for discriminative localization,” in *CVPR*, 2016.
- [26] R. R. Selvaraju *et al.*, “Grad-cam: Visual explanations from deep networks via gradient-based localization,” in *CVPR*, 2017.
- [27] X. Zhang *et al.*, “Adversarial complementary learning for weakly supervised object localization,” in *CVPR*, 2018.
- [28] D. Liu *et al.*, “Bringing ai to edge: From deep learning’s perspective,” *Neurocomputing*, 2022.
- [29] K. Zhang and G. Liu, “Layer pruning for obtaining shallower resnets,” *IEEE Signal Processing Letters*, 2022.
- [30] S. Teerapittayanon *et al.*, “Branchynet: Fast inference via early exiting from deep neural networks,” in *ICPR*, 2016.
- [31] G. Huang *et al.*, “Multi-scale dense networks for resource efficient image classification,” in *ICLR*, 2018.
- [32] T. Bolukbasi *et al.*, “Adaptive neural networks for efficient inference,” in *ICML*, 2017.
- [33] A. Graves, “Adaptive computation time for recurrent neural networks,” *arXiv preprint arXiv:1603.08983*, 2016.
- [34] X. Wang *et al.*, “Skipnet: Learning dynamic routing in convolutional networks,” in *ECCV*, 2018.
- [35] A. Veit and S. Belongie, “Convolutional networks with adaptive inference graphs,” in *ECCV*, 2018.
- [36] W. Hua *et al.*, “Channel gating neural networks,” *NeurIPS*, 2019.
- [37] J. Lin *et al.*, “Runtime neural pruning,” *NeurIPS*, 2017.
- [38] X. Gao *et al.*, “Dynamic channel pruning: Feature boosting and suppression,” in *ICLR*, 2019.
- [39] C. Herrmann *et al.*, “Channel selection using gumbel softmax,” in *ECCV*, 2020.
- [40] L. Yang *et al.*, “Resolution adaptive networks for efficient inference,” in *CVPR*, 2020.
- [41] K. He *et al.*, “Deep residual learning for image recognition,” in *CVPR*, 2016.

- [42] Y. Li *et al.*, “Tiny-dsod: Lightweight object detection for resource-restricted usages,” in *BMVC*, 2018.
- [43] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” in *ICLR*, 2015.
- [44] Z. Li and S. Arora, “An exponential learning rate schedule for deep learning,” in *ICLR*, 2020.
- [45] S. Laskaridis *et al.*, “Hapi: Hardware-aware progressive inference,” in *ICCAD*, 2020.
- [46] Y. Kaya *et al.*, “Shallow-deep networks: Understanding and mitigating network overthinking,” in *ICML*, 2019.
- [47] K. Simonyan and A. Zisserman, “Very deep convolutional networks for large-scale image recognition,” *arXiv preprint arXiv:1409.1556*, 2014.
- [48] C. Szegedy *et al.*, “Rethinking the inception architecture for computer vision,” in *CVPR*, 2016.
- [49] S. Zagoruyko and N. Komodakis, “Wide residual networks,” in *BMVC*, 2016.
- [50] Z. Huang and N. Wang, “Data-driven sparse structure selection for deep neural networks,” in *ECCV*, 2018.
- [51] Y. Wang *et al.*, “Learning versatile filters for efficient convolutional neural networks,” in *NeurIPS*, 2018.
- [52] L. Liebenwein *et al.*, “Provable filter pruning for efficient neural networks,” in *ICLR*, 2019.
- [53] X. Ding *et al.*, “Centripetal sgd for pruning very deep convolutional networks with complicated structure,” in *CVPR*, 2019.
- [54] S. Lin *et al.*, “Towards optimal structured cnn pruning via generative adversarial learning,” in *CVPR*, 2019.
- [55] J.-H. Luo and J. Wu, “Autopruner: An end-to-end trainable filter pruning method for efficient deep model inference,” *Pattern Recognition*, 2020.
- [56] G. Huang *et al.*, “Densely connected convolutional networks,” in *CVPR*, 2017.
- [57] S. Ioffe and C. Szegedy, “Batch normalization: Accelerating deep network training by reducing internal covariate shift,” in *ICML*, 2015.
- [58] C. Zhao *et al.*, “Variational convolutional neural network pruning,” in *CVPR*, 2019.



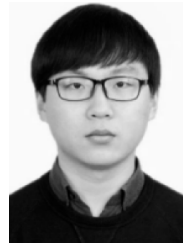
Hao Kong received the B.E. degree from University of Electronic Science and Technology of China, China, in 2019. He is currently a Ph.D. student at the School of Computer Science and Engineering, Nanyang Technological University, Singapore. His research interests include edge AI acceleration, model compression, and adaptive deep learning.



Di Liu is an Associate Professor at Department of Computer Science, Norwegian University of Science and Technology (NTNU), Norway. In prior to that, he was a research fellow at Nanyang Technological University, Singapore and a faculty member at Yunnan University, China. He received his PhD degree from Leiden University, The Netherlands, and both his MEng and BEng degrees from Northwestern Polytechnical University, China.



Shuo Huai received the B.E. degree from the School of Computer Science at Shandong University, China, in 2019. He is currently a Ph.D. student at the School of Computer Science and Engineering at Nanyang Technological University, Singapore. His research interests are efficient deep learning algorithms, embedded intelligence, and in-memory computing.



Xiangzhong Luo received the B.E. degree from Shanghai Jiao Tong University, Shanghai, China, in 2019. He is currently pursuing the Ph.D degree from the School of Computer Science and Engineering at Nanyang Technological University, Singapore. His current research interests include edge intelligence, hardware-aware neural architecture search, and model compression.



Ravi Subramaniam is a Distinguished Technologist at HP Inc working on Machine Learning algorithms and system architectures for ML at the edge and for low power and embedded systems. He has led solution development and standards work in IOT, P2P systems, distributed systems, Cloud computing, Operating systems over the last 28 years.



co-chair of IEEE Young Professionals for the IEEE Princeton/Central Jersey section.

Christian Makaya is currently a Principal Research Scientist at HP Labs, HP inc., Palo Alto, CA. He received the Ph.D. degree and the M.Sc. degree from University of Montreal, Quebec, Canada in 2007 and 2003, respectively. The focus of his current research interests is on artificial intelligence, machine learning, deep learning, distributed computing systems, edge computing, security and privacy, and network intelligence. He is a Senior Member of IEEE and serves on the Industry Outreach Board of IEEE Communication Society (ComSoc). He served as the



Science and Technology (IS&T) in 2012, and Outstanding Electrical Engineer by the School of Electrical and Computer Engineering of Purdue University in 2013. Dr. Lin received SWE Achievement Award in 2021.

Qian Lin is a HP Fellow and Chief Technologist for Machine Learning and Vision Systems in Personal Systems Software (PSSW). She is also an adjunct professor at Purdue University, supervising PhD students in their deep learning research. Dr. Lin joined Hewlett-Packard Company in 1992. She received her BS from Xi'an Jiaotong University in China, her MSEE from Purdue University, and her Ph.D. in Electrical Engineering from Stanford University. Dr. Lin is inventor/co-inventor for 45 issued patents. She was awarded Fellowship by the Society of Imaging



processor systems, and machine learning accelerators.

Weichen Liu is a Nanyang Assistant Professor at the School of Computer Science and Engineering, Nanyang Technological University, Singapore. He received the Ph.D. degree from the Hong Kong University of Science and Technology in 2011, and the B.Eng. and M.Eng. degrees from Harbin Institute of Technology, China, in 2004 and 2006, respectively. Dr. Liu authored and co-authored more than 100 research papers in peer-reviewed journals, conferences, and books. His research interests include embedded and real-time systems, multiprocessor systems, and machine learning accelerators.