RAPNet: Resolution-Adaptive and Predictive Early Exit Network for Efficient Image Recognition

Youbing Hu, *Student Member, IEEE,* Yun Cheng, *Member, IEEE,* Zimu Zhou, *Member, IEEE,* Zhiqiang Cao, *Student Member, IEEE,* Anqi Lu, *Member, IEEE,* Jie Liu, *Fellow, IEEE,* Min Zhang, *Member, IEEE* and Zhijun Li, *Member, IEEE*

***Abstract*—Deploying compute-intensive deep neural networks (DNNs) on resource-constrained end devices has become a prominent trend, enabling localized intelligence. However, efficiently deploying these DNNs at scale poses challenges. To address this, extensive research has focused on the early exit architecture based on convolutional neural networks (CNNs), which dynamically adapt network depth to reduce inference computation. Nevertheless, the sequential execution of all internal classifiers (ICs) and subsequent termination based on an exit criterion is inefficient. Motivated by these insights, we introduce a resolution-adaptive prediction network (RAPNet) architecture. RAPNet comprises a lightweight prediction network that captures global image features and an inference network integrated with an early exit architecture. The prediction network accurately determines the optimal IC position conditioned on the input images for efficient image classification. Additionally, we incorporate resolution-adaptive inference and feature fusion mechanisms by computational reuse, to effectively mitigate image spatial redundancy and improve the accuracy of ICs. We conduct extensive experiments across various datasets and architectures to demonstrate that RAPNet achieves a significantly better accuracy vs. computational trade-off than other recently proposed early exit methods. For instance, when using MobileNet as the base network, RAPNet achieves significant accuracy improvements of 12% and 5.7% on the Tiny Imagenet and CIFAR-100 datasets respectively, surpassing other early exit methods with similar computational constraints.**

***Index Terms*—Deep learning, early exit network, resolution adaptive inference, predictive early exit**

I. INTRODUCTION

INTELLIGENT applications driven by deep neural networks (DNNs) are experiencing a remarkable surge in popularity [1]. These applications are extensively utilized on wearable devices like smartphones and smartwatches, providing intelligent services across various domains [2], [3], [4]. With advancements in hardware platforms for end devices [9], [17] and the exceptional feature extraction capabilities of convolutional neural networks (CNNs) [18], DNNs have achieved unprecedented success in tasks such as image classification [6], object detection [7], [10], [11] and semantic segmentation [15], [16]. However, the resource constraints of most end devices, due to their compact size and lightweight nature, pose significant challenges in efficient DNN deployment [50], [76]. Furthermore, as DNN models directly interact with users on these devices, it is crucial to ensure high-quality user experience (QoE) by imposing strict limitations on the inference efficiency of these models [19]. The advent of dynamic neural network technology [20] has effectively addressed these challenges by dynamically allocating computational resources among samples in an uneven manner.

Efficient inference of DNNs on resource-constrained devices has been a focus of extensive research, leading to various technical approaches [12], [13]. These approaches encompass static models like knowledge distillation [21], network pruning [22], [23], and network architecture search (NAS) [24], [27], as well as dynamic models such as early exit architectures [14], [25], [28]. Unlike static models that allocate uniform computational resources to all samples, dynamic models have gained significant attention for their adaptive and efficient allocation of computational resources to individual samples. Among dynamic models, early exit neural networks have emerged as a simpler yet highly efficient approach, captivating researchers in the pursuit of efficient neural network architectures.

The early exit architecture improves a convolutional neural network (CNN) by attaching internal classifiers (ICs) into its hidden layer, which are jointly trained using a weighted optimization objective function. During inference, the ICs are executed sequentially until a convincing inference result is obtained, determined by comparing the confidence with a fixed threshold set as the exit criterion. Recent advancements in optimizing early exit architectures include techniques like training early exit networks using knowledge distillation [28] and enabling better collaboration among ICs by integrating the outputs of preceding ICs into the decisions of subsequent ICs through computational reuse [29], [30]. These methods have demonstrated outstanding recognition performance, as exemplified by the state-of-the-art approach, ZTW [29].

Although progress has been made in early exit architectures, existing approaches primarily focus on reducing computational overhead stemming from model redundancy. However, they tend to overlook the computational cost associated with the ICs themselves and the spatial redundancy of samples. In current implementations, all ICs are executed sequentially at a fixed resolution during inference, resulting in suboptimal computational efficiency. Additionally, the shallow ICs lack

Youbing Hu, Zhiqiang Cao, Anqi Lu, Jie Liu, Min Zhang and Zhijun Li are with the Faculty of Computing, Harbin Institute of Technology, Harbin 150000, China (e-mail: youbing@stu.hit.edu.cn; zhiqiang_cao@stu.hit.edu.cn; luanqi@stu.hit.edu.cn; jieliu@hit.edu.cn; zhangmin2021@hit.edu.cn; lizhijun_os@hit.edu.cn).

Yun Cheng is with the Swiss Data Science Center, ETH Zurich and EPFL, 8092 Zurich, Switzerland (e-mail: yun.cheng@sdsc.ethz.ch).

Zimu Zhou is with the School of Data Science, City University of Hong Kong, Hong Kong, China (e-mail: zimuzhou@cityu.edu.hk).

a global receptive field, leading to a significant decrease in accuracy. To achieve efficient inference of deep neural networks (DNNs) on resource-constrained end devices without sacrificing accuracy, a dynamic approach is needed to address both the computational cost arising from spatial redundancies in samples and model redundancies. However, it has the following challenges:

1) Effectively addressing the challenge of sequentially executing all ICs and dynamically selecting the optimal IC based on the input images for efficient image recognition remains a formidable task. This challenge is exacerbated by the varying accuracy and computational cost associated with each IC in the early exit network.
2) Designing an adaptive inference mechanism within the early exit network to address the spatial redundancy of samples without incurring additional computational cost is a significant challenge. This challenge arises from the fixed size of input images in the early exit network, which hinders the consideration of spatial redundancy and necessitates the development of a solution capable of supporting arbitrary resolutions.
3) Enabling shallow ICs to learn global feature representations of samples without introducing additional computational overhead poses a significant challenge. This is because shallow ICs inherently lack the capability to capture comprehensive global features, leading to a notable decline in their accuracy. Overcoming this challenge requires finding a solution that can provide global feature representation to shallow ICs while maintaining computational efficiency.

In this paper, we proposed a Resolution-Adaptive Prediction Network (RAPNet) architecture, which comprises a lightweight prediction network capable of learning the global feature representation of images and an early exit inference network. RAPNet's prediction network can precisely predict the optimal early exit position of input images in the early exit inference network, avoiding the high computational cost introduced by sequentially executing all ICs. Meanwhile, we designed resolution adaptive inference and feature fusion modules by computational reuse. The former is achieved through a specialized internal classifier structure that enables the processing of input images with arbitrary resolutions, thereby reducing computational costs associated with spatial redundancy. The latter involves fusing the global features learned by the predictive network with the local features extracted by the CNN. This fusion enhances the semantic information of the input IC, resulting in a significant improvement in accuracy without incurring additional computational costs. This effect is particularly pronounced for ICs located at CNN shallow layers. In summary, the contributions of this work are summarized as follows:

1) We proposed the Resolution-Adaptive Prediction Network (RAPNet), comprising a lightweight prediction network and an early exit inference network. The prediction network efficiently learns global image features and accurately predicts the optimal early exit position within the inference network, avoiding the need for sequential execution of all ICs and minimizing computational costs.
2) We have developed a dedicated internal classifier within the inference network to support input images of any resolution, effectively reducing computational costs associated with spatial redundancy in the image.
3) We introduced a feature fusion mechanism that leverages the global features extracted from the prediction network, effectively addressing the limited global feature representation in the ICs of the early exit network. This fusion mechanism greatly improves the accuracy of the internal classifiers within the inference network.
4) We conducted extensive experimental evaluations for the proposed methods on image classification tasks. The results demonstrated that RAPNet achieves a significantly better accuracy-computational trade-off than other recently proposed early exit methods. For instance, when leveraging MobileNet as the base network, RAPNet improved the accuracy of the Tiny Imagenet and CIFAR-100 datasets by 12% and 5.7%, respectively, compared to other early exit methods with similar computational constraints.

In the rest of the paper, we introduce the related work in Sec. II and elaborate on the RAPNet design in Sec. III. Finally, we evaluate the performance of RAPNet in Sec. IV and give concluding remarks in Sec. V.

II. Related Work

A. Vision Transformer

Inspired by the remarkable success of Transformer in NLP tasks [35], researchers developed Vision Transformer [36] for image recognition. The Vision Transformer divides input images into sequences of image blocks and transforms them into tokens as input. The backbone structure of the Transformer consists of stacked building blocks, each of which contains self-attention layers and feedforward networks for processing these tokens. Vision Transformer builds on the self-attention mechanism to efficiently capture remote dependencies between patches from the input images and is widely adopted in the computer vision community [34], [37], [38]. However, the lack of induction bias [36] requires ViT models to be pre-trained on very large datasets such as JFT-300M [39] in order to pursue the desired performance. Extensive research efforts have been dedicated to enhancing the efficiency and minimizing the computational burden of the visual transformer [40], [41], [42], [43], [44]. Swin Transformer [45] develops multi-stage network architectures with down-sampling and obtains better inference efficiency. DeiT [43] introduces an extra token for knowledge distillation. LV-ViT [44] leverages all tokens to compute the training loss, and the location-specific supervision label of each patch token is generated by a machine annotator.

Our work leverages the ability of self-attention to capture long-range dependencies between patches and to extract global features from the input images. However, compared to existing efficient ViT models, our work is significantly different in several aspects. Firstly, we focus on CNN models with early exit structure and merely use the self-attention mechanism to extract global features from the input images. Furthermore, when applying the self-attention mechanism, we do not require a fine-grained partition of the image. Instead, we extract coarse-grained global features. As a result, the model can maintain a smaller depth and a shorter sequence length of tokens.

B. Conditional Computation and Multi-Scale Features

Conditional computation based on deep neural networks was first proposed in [59], [60], and since then more sophisticated and efficient methods have been proposed in this field, including dynamic routing [61], cascading with multiple networks [26], [62], [63] and skipping intermediate layers [66] or channels [64], [65]. Besides the above, early exit architectures [25] are widely studied in the field of dynamic neural networks [20] by adding ICs to the network allowing easy samples to exit earlier. BranchyNet [25] is the first to add an IC to the hidden layer of a CNN and then utilize a confidence-based early exit strategy to allow samples with sufficient confidence to an early exit. Shallow-Deep Networks (SDN) [32] is a conceptually simple but efficient method that uses the top-1 accuracy of the softmax output compared to a fixed-threshold confidence score as an early exit policy, significantly improving the efficiency of network inference while enabling researchers to further understand the phenomenon of network overthinking. Patience-Based Early Exit (PBEE) [31] combines the outputs of the previous ICs and terminates the inference only when the outputs of the successive τ ICs are consistent, and it outperforms SDN on a number of NLP tasks. Zero Time Waste (ZTW) [30] further reuses the computational results of the previous classifier, adding direct connections between ICs and combining previous outputs in an ensemble-like manner, its performance has been further improved. The Exit Predictor (EP) [33] method achieves computational savings by introducing a prediction exit mechanism to allow easy samples to bypass the execution of certain ICs.

CNN's extract global features by increasing depth to continuously enlarge the perceptual field, which uses pooling operations or convolution operations with steps for downsampling [49], [67], which may limit the network's ability to recognize objects at arbitrary scales. Recent studies have proposed to utilize both coarse global features and fine local features in the network, which has significantly improved the performance of the network in many vision tasks, including image classification [5], object detection [7], semantic segmentation [8], and pose estimation [71]. DS-Net [75] proposed an Intra-scale Propagation module to process two different resolutions in each block and an Inter-Scale Alignment module to perform information interaction across features at dual scales. Moreover, the multi-scale structure [63] shows a promising ability in adaptive inference and memory-efficient network [72], [73]. RANet [26] designs a resolution-adaptive network structure by sequentially maintaining multiple network layers from low to high resolution, which exists simultaneously throughout the network with multiple scale feature maps that are used sequentially from low to high resolution for object recognition, reducing the computational overhead from the spatial redundancy of samples.

Our approach is based on the early exit architecture and the idea of a predictive exit mechanism [33], but it differs significantly. First, our method predicts precisely which IC to use for inference, while the EP method only predicts which ICs can be skipped. Secondly, our mechanism uses the global feature representation of input samples, considering spatial redundancy, whereas the EP method relies on local features and ignores spatial redundancy. Finally, our method improves IC accuracy through computational reuse without wasting extra computations, unlike the EP method, which introduces extra computations for coarse-grained IC prediction.

III. Method

The early exit network incorporates ICs into the hidden layer of the CNN. It selectively allocates computational resources to each input image based on an exit criterion, determining whether to terminate the execution of the current IC. However, executing multiple ICs sequentially to identify challenging images and achieve high computational accuracy can lead to wasteful computations for ICs that do not meet the exit criterion. Furthermore, ICs attached to shallow layers of the CNN lack access to global features of the input image due to limited receptive fields. Consequently, the accuracy of these shallow ICs significantly decreases. Inspired by these observations, we introduce a resolution-adaptive predictive early exit network, aiming to improve the computational efficiency of early exit CNNs. Our approach focuses on accurately predicting the optimal IC for each image within the early exit network, thereby improving computational efficiency. Additionally, we seek to significantly improve the accuracy of shallow ICs and reduce the spatial redundancy of images by leveraging computational reuse techniques.

Specifically, we propose a Resolution-Adaptive Predictive Network, which consists of a lightweight prediction network capable of extracting global features of the input image and an early exit CNN network. The prediction network directly predicts the optimal IC within the early exit CNN by extracting global features from the input image. This approach avoids the computational waste associated with activating all ICs in sequence. Additionally, to address the issue of reduced accuracy in shallow ICs due to limited global receptive fields, we introduce a feature fusion mechanism that significantly improves the accuracy of these classifiers. To mitigate the computational cost from spatial redundancy in input images, we design ICs for the early exit CNN network that can support arbitrary resolutions. This customization is achieved without sacrificing accuracy, as demonstrated in Fig. 5.

A. Overview

Inference. We start by describing the inference procedure of RAPNet, which is shown in Fig. 1. For each test sample $x \in R^{C \times H \times W}$ (C, H, and W represent channel, height, and width respectively), it is first partitioned x into the tokens $t \in R^{K \times E}$, K being the total number of tokens and E the embedding dimension of each token. Then the global features of the input image are extracted by stacking N transformer blocks comprising multi-head self-attention and FFN. Finally,

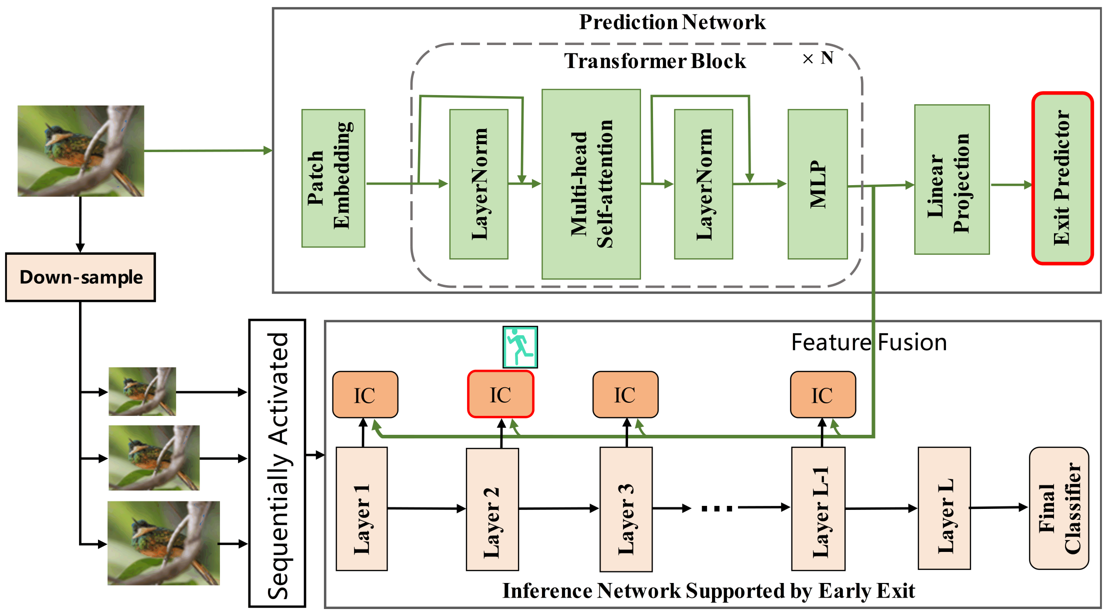


Fig. 1. An overview of RAPNet. The Internal Classifier (IC) is a dedicated internal classifier designed to support input images of any resolution. RAPNet consists of a lightweight prediction network that utilizes global features learned from the images and an early exit inference network. The prediction network uses the multi-head self-attention mechanism to captures the global features of the image and predicts the optimal IC within the inference network. Subsequently, the input images are sequentially fed into the inference network, starting from low to high resolution. At test time, different resolutions of the images are sequentially activated until a convincing prediction (e.g. sufficiently confident) has been obtained or the final classifier has been inferred.

these global features are computed by an exit predictor to obtain $L-1$ output values $\mathbf{s}$, where each value $s_i \in (0,1)$, L is the number of layers of the early exit CNN network. The exit predictor will be introduced in Sec. III-B. Once the value of s_i is greater than η of a pre-set adjustable threshold, the i-th IC of the early exit CNN is selected as the final executive classifier. Otherwise the l-th classifier of the base network is selected as the final executive classifier. When the inference network starts performing task inference, the resolution of the input image is activated sequentially from low to high resolution until a convincing prediction (greater than threshold γ) has been obtained or the final classifier has been inferred. The resolution adaptive inference will be described in detail in Sec. III-C. It is worth noting that if there is no s_i greater than η, then the l-th classifier of the base network is chosen as the final executing classifier, using only the original input resolution, because we do not modify the structure of the base network, and therefore it does not support other resolutions of the input images.

Training. To ensure RAPNet is trained properly, we propose a 3-stage training scheme, where the first two stages are indispensable, and the third stage is designed to further improve the performance.

Stage-1: Training the prediction network backbone. When designing the prediction network, we employ a linear projection to align the dimensionality of the extracted global features with the number P of classes in the classification task. During training, we exclude the exit predictor and treat the linear projection as a standalone classifier. Subsequently, we minimise the loss $\mathcal{L}_{pred}$ (Eq. 1) to train the backbone of the prediction network.

$$\mathcal{L}_{pred} = CE(\mathbf{p}_{pred}; \mathbf{y}) \tag{1}$$

where $\mathbf{p}_{pred}$ is the output of the prediction network, $\mathbf{y}$ is the ground true label, $CE(.;.)$ represent the cross entropy loss.

Stage-2: Training early exit inference networks supporting arbitrary input resolution. In this stage, we fuse the global features obtained from the prediction network's backbone, trained in stage-1, into each IC of the inference network. These fused features are utilized during the training of the inference network, as described in Sec. III-D. We trained the early exit inference network by utilizing images of varying input resolutions, aiming to minimize the classification loss $\mathcal{L}_{cls}$ (Eq. 2).

$$\mathcal{L}_{cls} = CE(\mathbf{p}_f; \mathbf{y}) + \sum_{r=m}^{M} \sum_{i=1}^{L-1} KL(\mathbf{p}_{i,m}; \mathbf{p}_f) \tag{2}$$

where $KL(.;.)$ represent the Kullback-Leibler divergence. $\mathbf{p}_f$ is the predicted output of the base network. m is the resolution of the input image, e.g. for the ImageNet dataset we set $m \in M$={112×112, 162×162, 224×224}. $\mathbf{p}_i$ is the predicted output of the i-th IC. It is worth noting that the output $\mathbf{p}_f$ of the final classifier is computed only once based on the original input resolution.

Stage-3: Finally, we fine-tune the exit predictor of the prediction network. In this stage, we freeze the backbone of the prediction network and then fine-tune the exit predictor only. We added on the exit predictor and then let RAPNet's inference network serve as the network that generates supervised data to fine-tune the exit predictor for a given set of images $\{\mathbf{x}_b, \mathbf{y}_b\}_{b=1}^{B}$. Specifically, for each mini-batch of images, a resolution m is randomly chosen from the set of resolutions

M. The images are then resized to the resolution $m \times m$ and fed into the inference network, resulting in $L-1$ Softmax top-1 scores $c_{i,b}$. These scores are compared with a predetermined threshold η (during training, we always set η to 0.5), and if $c_{i,b}$ is greater than η, $\mu_{i,b}$=1, otherwise $\mu_{i,b}$=0. Finally, we minimize the loss function of Eq. 3 to update exit preditor.

$$\mathcal{L}_{pred} = \frac{1}{B}\sum_{b=1}^{B}\sum_{i=1}^{L-1} BCE(s_{i,b};\mu_{i,b}) \tag{3}$$

where $s_{i,b} \in (0,1)$ is the output distribution calculated after the Sigmoid function of the exit predictor, $BCE(.;.)$ is the binary cross entropy loss.

B. Prediction Network

The primary role of the RAPNet prediction network is to effectively extract the global feature representation from the input image and utilize these features to predict the optimal IC within the inference network. At test time, the prediction network is executed first for each input image to determine its optimal IC within the inference network. Therefore, it is crucial for the prediction network to be lightweight in order to ensure efficient execution. In Fig. 1, the backbone of the RAPNet prediction network utilizes a multi-head self-attention mechanism to efficiently capture the global features of the input image. It comprises three main components: patch embedding, N transformer blocks, and linear projection. The patch embedding step divides and embeds the input image into $t \in R^{K\times E}$ tokens. These tokens are then processed by N transformer blocks, which extract the global feature representation of the image. Each transformer block consists of multi-head self-attention and MLP layers, with LayerNorm used for residual concatenation. Finally, the extracted global feature representation is mapped to the same dimension as the classification task (e.g. 100 for the cifar100 dataset) through a linear transformation. This dimension mapping aids in the training of the prediction network. To achieve a lightweight and efficient prediction network for RAPNet, we adopt the Swin Transformer [45] as the backbone. Specifically, we empirically set the network depth N to 2 and the embedding dimension E to 48, as indicated in Table IV. Consistent with the original paper, the window size is set to $M = 7$ by default. This choice ensures that the prediction network can effectively extract the global features of the input image while maintaining its lightweight nature.

The exit predictor of the prediction network utilizes linearly transformed global feature inputs to determine the optimal IC in the inference network. This enables the prediction network to exit the processing and provide the best IC for the image within the inference network. The exit predictor in RAPNet operates in a similar manner to the early exit architecture, leveraging a predictive network to effectively learn global feature representations of images. Subsequently, these global features are used to calculate a probability s_i for each IC. When the probability s_i for the i-th IC surpasses a predefined adjustable threshold η, the inference network selects the i-th IC as the final classifier for execution. This approach eliminates

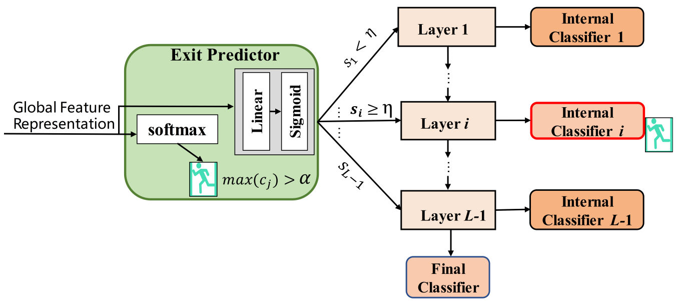


Fig. 2. The structure and execution process of RAPNet's exit predictor. At test time, we introduce two thresholds α and η to realize a trade-off between performance and computational. The value of α controls the proportion of images that can be directly recognized using the global features extracted by the prediction network. The value of η is used to select the optimal IC for the image within the inference network.

the need for sequential execution of all ICs, resulting in improved efficiency.

Fig. 2 depicts the structure and execution process of RAPNet's exit predictor. It consists of a Softmax-based exit criterion and a linear transformation module followed by a Sigmoid function. The linear transformation module maps the input global features to the same dimension $L-1$ as the hidden layer of the inference network. The Sigmoid function is then employed to transform the values between 0 and 1, representing the probability of selecting each IC. We introduce two thresholds α and η to realize a trade-off between performance and computational. The threshold α controls the percentage of input images that can be directly classified using the global features extracted by the prediction network, resulting in minimal computational cost. In our implementation, if Softmax's top-1 output $max(c_j) > \alpha$, the inference terminates immediately and the attribute input to category j. Otherwise, the input image undergoes further processing by the optimal IC predicted by the exit predictor. During the training process, we set the value of α to 1, which indicates that all images are included in the training of the RAPNet inference network. If $s_i > \eta$, the i-th classifier is selected as the final classifier for inference. When performing inference, the classifiers before the selected IC are skipped, and any subsequent ones are no longer executed, which significantly saves the computational overhead introduced by the sequence execution of all ICs. If all s_i are less than η, the L-th classifier of the base network is used to infer and no ICs are needed to execute the inference.

C. IC Supporting Resolution-Adaptive Inference

Fig. 3 provides a detailed overview of the IC in RAPNet, which consists of an initialization layer, a feature fusion module, and an output layer. The initialization layer applies **M** consecutive convolution operations with step sizes based on the input image resolution. This efficiently reduces the feature map size and decreases the computational cost of the IC. For instance, in this paper, **M** is set to 2 if the input image resolution is greater than 112, 1 if the resolution is greater than 64, and 0 if the resolution is less than 64. The feature fusion module is explained in detail in Sec. III-D. The output layer consists of a feature reduction (FR) layer and a

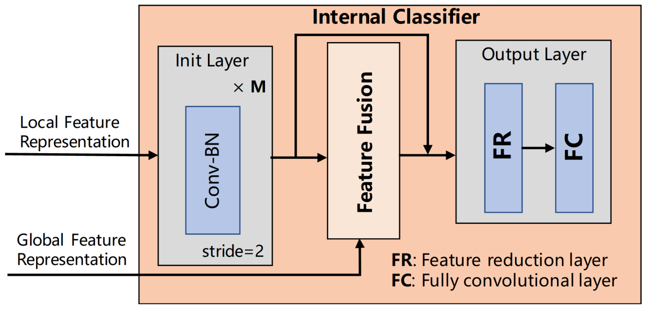

Internal Classifier
Init Layer
× M
Conv-BN
stride=2
Feature Fusion
Output Layer
FR
FC
Local Feature Representation
Global Feature Representation
FR: Feature reduction layer
FC: Fully convolutional layer

Fig. 3. The IC details of RAPNet. The ICs in RAPNet include initialization layers, a feature fusion module, and an output layer. The initialization layers use consecutive convolution layers with a stride of 2 and batch normalization to quickly reduce the feature map size and computational cost. The feature fusion module addresses the lack of global features in ICs. The output layer is a fully convolutional network, allowing for input images of any resolution.

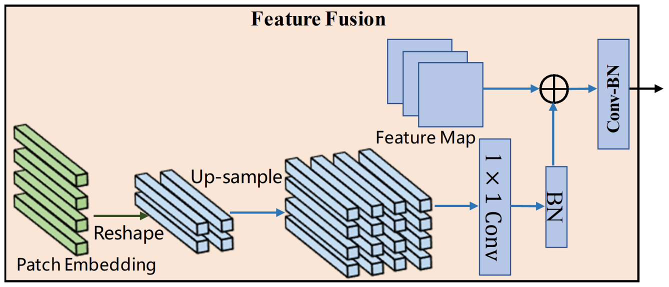


Fig. 4. The feature fusion module of RAPNet. Reshape and up-sampling operations are employed to align the spatial dimension of global features represented as tokens and local features represented as feature maps. The 1×1 convolution is applied to align their channel dimensions. The BN layers are utilized to alleviate the semantic gap between different feature representations.

fully convolutional (FC) layer. In the case of high-resolution images, despite the downsampling operations performed in the initialization layer to reduce the feature map size, the resulting feature maps can still be quite large. To further decrease the computational cost and reduce the feature map size, RAPNet incorporates the AdaptiveMaxPool2d operation for feature reduction. Following the approach in SDN [32], we choose the pooling size such that any feature map larger than 4x4 is pooled into a size of 4×4, while smaller sizes remain unchanged.

The ICs of RAPNet utilize fully convolutional layers instead of fully connected layers as the final classifiers, allowing for the flexibility to handle input images of any resolution. The number of input channels in the fully convolutional layer corresponds to the number of output channels obtained after fusion through the feature fusion module. The number of output channels corresponds to the number of classes in the classification task, while the size of the convolutional kernel matches the size of the feature map outputted by the FR layer.

In Fig. 5, we introduced six early exits in Wideresnet and Resnet, placed at 15%, 30%, . . .90% of the network's total FLOPs. The converted networks are denoted as EE-Wideresnet and EE-Resnet, with the suffix -Conv indicating the use of fully convolutional layers for the ICs. As can be seen from the figure, replacing the fully-connected layer with a fully-convolutional layer results in a significant improvement in the accuracy of the earlier ICs, while the later ICs show only a slight decrease in accuracy. For example, the accuracy of the first IC in EE-Restnet-Conv and EE-WideResNet-Conv improves by more than 4%, and most of the other ICs show higher accuracy compared to their performance before the replacement. Only the fifth IC in EE-Restnet has slightly lower accuracy than EE-Resnet-Conv. Thus, using a fully-convolutional layer not only avoids increasing the computational effort, but also improves the accuracy of the ICs compared to using a fully-connected layer.

By designing the ICs of RAPNet as fully convolutional networks, it supports input images of arbitrary resolutions. Therefore, during the training of RAPNet, we use input images with different down-sampled resolutions and train the early exit inference network using the loss function in Eq. 2. It is worth noting that all input images of different resolutions share the same set of parameters. At test time, each input image first passes through the prediction network to determine its optimal IC in the inference network. Subsequently, the input image is sequentially fed into the inference network from low to high resolutions until a convincing prediction (greater than the threshold value γ) is obtained or the final classifier completes the inference. Here, we use a confidence-based exit criterion and set the threshold value as γ.

D. Feature Fusion

Fig. 4 presents the details of the feature fusion module. The feature map dimension extracted by CNN is $C\times H\times W$, where C, H, and W are the channel, height, and width of the feature map, respectively. While the patch embedding of the global features extracted by the prediction network is $K \times E$, where K is the number of image patches and E is the embedding dimension. The feature fusion module of RAPNet first reshapes the patch embeddings, then adapts its spatial dimension with the CNN feature map alignment using the up-sampling operation, and finally adapts the channel dimension with the CNN feature map alignment using the 1×1 convolution. By aligning the patch embeddings with the feature map in both spatial dimension and channel dimension, there is still a huge semantic gap between the patch embeddings and the feature map. To alleviate this semantic gap, we normalize the patch embeddings using BN.

IV. Evaluation

This section discusses the performance results of RAPNet. We present four datasets and four baselines for evaluating the accuracy achieved by RAPNet. Then, we discuss the impact of different components of RAPNet on the overall performance separately. Finally, we reported the energy consumption of RAPNet when performing inference on the Nvidia Jetson Nano device.

A. Experimental Setup

The RAPNet is implemented with Pytorch[1] 1.11.0. We used four popular open-source datasets on classification tasks to

[1]https://pytorch.org/

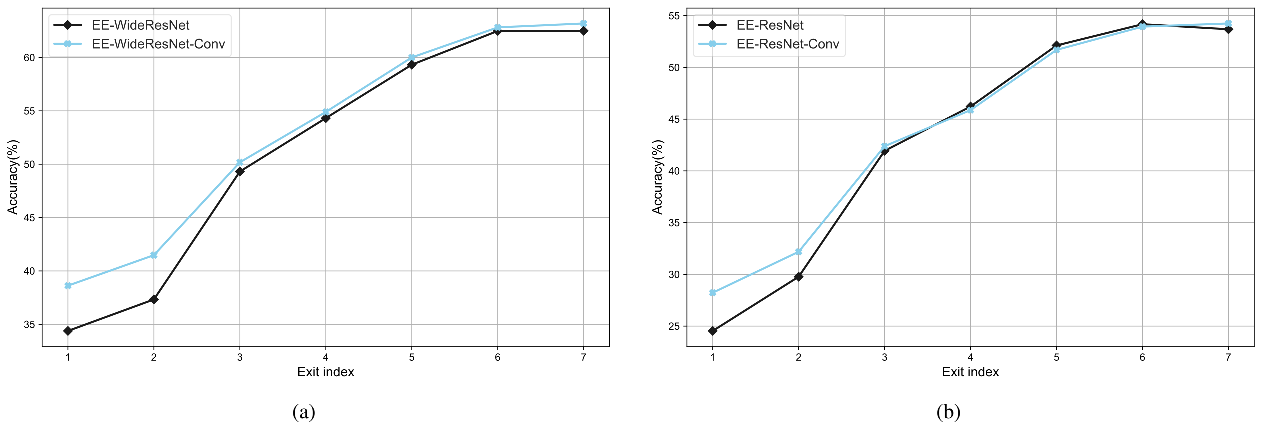


Fig. 5. Change in the accuracy of the IC on the Tiny Imagenet test set after replacing the fully-connected layer with a fully-convolutional layer, denoted by the suffix "-Conv". The utilization of full-convolutional layers not only reduces computational but also improves the accuracy of the IC. (a) WideResNet. (b) ResNet.

evaluate the proposed RAPNet: CIFAR-10 [46], CIFAR-100 [46], Tiny Imagenet [48] and ImageNet [47] datasets, and four common CNN architectures ResNet-56 [49], MobileNet [51], WideResNet [52] and VGG-16BN [53] as the base networks. To evaluate the efficiency of the models, we calculated the average number of FLOPs required to perform forward pass on a single sample and use it as a hardware-independent measure of inference cost. The top-1 accuracy commonly used for classification tasks was used as a performance metric. During the evaluation of RAPNet's energy consumption, we conducted experiments using an Nvidia Jetson Nano (Nano) equipped with an ARM A57 CPU, 4GB of memory, and a 128-core Maxwell GPU as the testing device.

Both the CIFAR-10 and CIFAR-100 datasets contain 50,000 training and 10,000 test images of resolution 32×32, corresponding to 10 and 100 classes, respectively. We hold out 5,000 images in the training set as a validation set to search the optimal confidence threshold for resolution adaptive inference. Tiny Imagenet contains 100,000 training and 10,000 test images of resolution 64×64. For resolution adaptive inference, we randomly hold out 10, 000 images in the training set as the validation set to search threshold for resolution adaptive inference. The ImageNet dataset contains 1.2 million images of 1,000 classes for training, and 50,000 images for validation. For adaptive inference tasks, we use the original validation set for testing, and hold out 50,000 images from the training set as a validation set.

We apply standard data augmentation schemes on the CIFAR, Tiny ImageNet and Imagenet datasets. On the two CIFAR datasets, images are randomly cropped to samples with 32×32 pixels after zero-padding 4 pixels on each side. Furthermore, images are horizontally flipped with probability 0.5 and RGB channels are normalized by subtracting the corresponding channel mean and divided by their standard deviation. On Tiny ImageNet dateset, images are randomly cropped to samples with 64×64 pixels after zero-padding 8 pixels on each side. Then, images are horizontally flipped with probability 0.5, and brightness, contrast and saturation are modified with probability 0.2. The RGB channels are normalized by subtracting the corresponding channel averages and dividing by their standard deviations. On ImageNet, images are horizontally flipped with probability 0.5 and RGB channels are normalized by subtracting the corresponding channel mean and divided by their standard deviation for training, and apply a 224 × 224 center crop to images at test time.

We train the proposed models using stochastic gradient descent (SGD) with a multi-step learning rate policy. To enable resolution-adaptive inference in RAPNet, we train the model using different input resolutions simultaneously. For the CIFAR dataset, we use an input resolution set M of $\{16\times16$, 28×28, $32\times32\}$. For the TinyImageNet, the resolution set M is set to $\{32\times32$, 48×48, $64\times64\}$. Finally, for ImageNet, the resolution set M is set to $\{112\times112$, 192×192, $224\times224\}$. The batch size of both CIFAR and Tiny ImageNet datasets is set to 128. The batch size of the ImageNet dataset is set to 64. In all datasets and models, we use a momentum of 0.9. For VGG-16BN and WideResNet we set the weight decay to 0.0005 and the other models to 0.0001. Moreover, for the CIFAR and Tiny ImageNet datasets, the models are trained from scratch for 100 epochs with an initial learning rate of 0.06, which is divided by a factor of after 35, 60 and 85 epochs. The same training scheme is applied to the ImageNet dataset. And we train the model from scratch using an initial learning rate of 0.01 for 90 epochs and the initial learning rate decreases after 30 and 75 epochs. Throughout the experiment, we maintained a fixed value of η at 0.5 for RAPNet, ensuring consistency with the training phase.

For the prediction network, we consistently used the following settings in all experiments: the attention head count was set to 6, the embedding dimension E was set to 48, and the depth N of the transformer block was set to 2.

We compared RAPNet to the following recent state-of-the-art approaches for early exit architectures described in the literature.

1) SDN [32]: Shallow-Deep networks (SDN) is a conceptually simple and effective method in which a comparison of top-1 execution based on the softmax output with a fixed threshold is used as an exit criterion. Internal classifiers are attached to layers selected according to the number of computational operations required to reach

them.

2) PBEE [31]: Patience-Based Early Exit (PBEE) architecture, which terminates inference after τ consecutive unchanging answers. In this paper, we set τ=2, meaning that if the answer of the current IC is the same as the answers of the previous two ICs, we terminate the execution and return that answer, otherwise we continue the computation.
3) ZTW [30]: Zero Time Waste (ZTW), early exit architecture by computational reuse and integration. The feature representation is augmented by directly connecting the output of the previous IC with the output of the subsequent classifier via computational reuse, while the prediction results of the previous classifier are combined with the prediction results of the subsequent classifier using arithmetic integration.
4) EP [33]: Exit Predictor (EP), a predictive early exit mechanism based on deeply separable convolution [58] was designed to guide some apparently hard samples to bypass some early exit computation.

B. Performance Comparison of Different Methods

1) RAPNet Performance on Different Computational: We examined the highest accuracy of the different methods given the same computational resources (relative to the percentage of the base network computational operations). To ensure a fair comparison, we re-executed the open-source code provided in the papers for SDN, PBEE, and ZTW, considering the given resource constraint. We selected the configuration that yielded the highest precision in our experiments. As for the EP method, we implemented it based on the description provided in the paper and fine-tuned the hyper-parameters to achieve the maximum accuracy. For RAPNet, we adjusted the threshold values α and γ to maximize accuracy within the given resource constraint. The results are presented in Table I.

Based on the table provided, several observations can be made regarding the different methods. Firstly, PBEE consistently achieves the lowest accuracy among all methods, especially when operating under low computational settings (less than 50%). This is primarily because PBEE lacks flexibility in adapting to computational constraints, as it requires consistent prediction results for a fixed number of τ consecutive ICs. On the other hand, SDN employs a fixed confidence threshold as the exit criterion and dynamically allocates computational resources to different samples, effectively managing computations within the specified computational limits. For example, when utilizing MobileNet as the base network with a computational limit of 10%, SDN successfully computes easy samples using the first IC and challenging samples using the base network. This approach results in a significant reduction of 90% in computational cost while sacrificing only 10% of accuracy on the C10 dataset. In contrast, ZTW introduces direct connections between individual classifiers through computational reuse. This allows the predictions from previous classifiers to contribute to the decisions made by subsequent classifiers, resulting in improved IC performance and higher accuracy compared to SDN, EP, and PBEE across all four models. The EP method enables certain distinct hard samples to bypass the execution of multiple ICs through a prediction early exit mechanism. However, the introduction of a prediction network inherently introduces additional computational overhead, and the accuracy of the prediction network predictions critically impacts the overall accuracy. Consequently, to meet the constraints of low computational, only the previous classifiers are utilized for prediction, leading to poorer end-to-end accuracy. For instance, when the computational power is set to 10%, the VGG model sacrifices 20% accuracy on the C10 dataset, representing a 10% reduction compared to SDN.

Compared to the aforementioned methods, RAPNet consistently achieves the highest accuracy across various computational resources, demonstrating minimal accuracy degradation. This can be attributed to RAPNet's efficient reduction of computational overhead, addressing both model redundancy and sample space redundancy through the predictive exit mechanism and resolution-adaptive inference. Additionally, RAPNet effectively leverages global features of input samples to enhance the accuracy of ICs, resulting in significantly higher accuracy compared to the mentioned methods, especially when computational power is limited to less than 25%. For instance, when utilizing ResNet as the base network, RAPNet exhibits a maximum accuracy improvement of approximately 11% compared to the state-of-the-art ZTW method on the T-IM dataset. In contrast to EP's predictive early exit mechanism, which introduces additional computational overhead merely to bypass unnecessary classifier execution, RAPNet's predictive early exit mechanism is more precise and enhances IC accuracy through computational reuse. While PBEE incorporates information reuse from previous layers to determine computation termination, this approach alone is insufficient to effectively reduce computational waste in the network. Similarly, although ZTW introduces integration among classifiers through computational reuse, the limitation of lacking global feature extraction from samples restricts the extent of computational reduction. Overall, RAPNet's combination of an accurate predictive early exit mechanism and feature extraction capabilities enables superior performance in terms of accuracy and computational efficiency compared to the mentioned methods.

The evaluation results of the 60 models mentioned above demonstrate that RAPNet consistently outperforms other baseline methods in terms of computational efficiency while maintaining high accuracy. This superior performance is observed across all datasets, models, and architecture combinations. To further validate our observations on larger datasets, we conducted experiments on ImageNet using a pre-trained ResNet-50 model from the torchvision package. The results presented in Table II demonstrate that RAPNet exhibits significant improvements over the four tested baselines even in this more challenging scenario.

2) Quantitative Analysis: Fig. 6 shows the number of correctly identified images by RAPNet at different stages on the CIFAR-100 dataset, using WideResNet as the base network. The value of α is fixed at 0.9, while γ is adjusted accordingly. The figure also demonstrates the accuracy variation of RAPNet across different computational settings. By adjusting the

TABLE I
RESULTS ON FOUR DIFFERENT ARCHITECTURES AND THREE DATASETS: CIFAR-10 (C10), CIFAR-100 (C100) AND TINY IMAGENET (T-IM). TEST ACCURACY PERCENTAGE (%) FOR RESOURCE BUDGETS: 10%, 15%, 25%, 50%, 75%, 100% OF THE BASE NETWORK. THE FIRST COLUMN SHOWS TEST ACCURACY OF THE BASE NETWORK.

WideResNet								**VGG**							
Data	**Algo**	**10%**	**15%**	**25%**	**50%**	**75%**	**100%**	**Data**	**Algo**	**10%**	**15%**	**25%**	**50%**	**75%**	**100%**
T-IM 59.9	SDN	30.8	33.3	36.8	46.0	54.7	59.4	**T-IM** 58.8	SDN	30.7	37.1	40.1	50.5	57.4	59.6
	PBEE	24.6	24.6	29.9	37.8	52.7	58.5		PBEE	21.5	21.5	30.4	45.2	55.2	**60.1**
	ZTW	32.4	37.9	40.0	50.1	57.5	**60.2**		ZTW	31.7	39.6	41.4	52.3	59.3	**60.1**
	EP	25.2	27.7	34.4	47.4	56.0	59.6		EP	22.9	29.3	37.5	48.8	56.5	**60.1**
	RAPNet	**42.3**	**43.0**	**44.0**	**51.9**	**59.2**	**60.3**		RAPNet	**44.6**	**45.5**	**47.1**	**55.4**	**60.3**	**60.3**
C100 75.3	SDN	51.9	53.3	55.9	65.1	71.6	75.0	**C100** 70.6	SDN	50.1	56.2	58.5	67.2	70.6	71.4
	PBEE	41.3	41.3	46.7	57.2	66.0	73.2		PBEE	37.8	37.8	51.2	65.3	65.3	70.9
	ZTW	52.1	53.8	59.5	69.1	**74.5**	**76.2**		ZTW	52.0	57.3	60.2	69.3	**72.6**	**73.5**
	EP	46.2	48.8	50.1	61.2	72.4	75.3		EP	39.4	43.6	56.2	66.7	71.5	71.7
	RAPNet	**53.1**	**54.6**	**60.3**	**70.8**	**74.9**	**76.2**		RAPNet	**56.7**	**58.9**	**61.5**	**70.2**	**72.6**	**73.5**
C10 94.1	SDN	79.8	81.7	83.8	91.7	**94.1**	**94.4**	**C10** 92.9	SDN	78.1	83.4	85.4	**92.1**	**93.0**	**93.0**
	PBEE	72.2	72.2	78.0	84.0	90.3	93.8		PBEE	62.3	62.3	75.0	86.0	91.0	**93.1**
	ZTW	81.7	83.5	86.7	92.9	**94.5**	**94.7**		ZTW	81.9	84.8	87.1	**92.5**	**93.2**	**93.2**
	EP	74.5	79.6	83.2	89.6	92.2	**94.3**		EP	65.2	68.7	76.1	88.1	92.4	**93.0**
	RAPNet	**84.4**	**86.0**	**90.1**	**94.2**	**94.8**	**94.8**		RAPNet	**85.0**	**86.2**	**90.5**	**92.6**	**93.2**	**93.2**
ResNet-56								**MobileNet**							
Data	**Algo**	**10%**	**15%**	**25%**	**50%**	**75%**	**100%**	**Data**	**Algo**	**10%**	**15%**	**25%**	**50%**	**75%**	**100%**
T-IM 53.8	SDN	23.8	27.8	31.2	41.2	49.9	54.5	**T-IM** 59.6	SDN	28.2	32.1	35.6	47.1	55.3	58.9
	PBEE	21.2	21.2	29.0	37.6	48.2	53.4		PBEE	20.8	20.8	26.7	38.4	50.3	55.6
	ZTW	25.4	31.7	35.2	46.2	53.7	**56.3**		ZTW	30.1	34.3	37.3	49.5	56.7	59.7
	EP	22.4	25.7	34.4	41.8	52.1	55.0		EP	22.5	28.9	37.0	47.0	55.8	57.7
	RAPNet	**44.0**	**44.1**	**46.3**	**51.3**	**54.2**	**56.4**		RAPNet	**45.6**	**46.3**	**47.2**	**51.1**	**57.5**	**60.1**
C100 69.0	SDN	42.3	45.1	47.1	57.2	64.7	69.0	**C100** 65.1	SDN	49.0	52.1	54.3	63.5	66.8	67.8
	PBEE	38.6	38.6	45.2	53.5	60.1	67.0		PBEE	39.1	39.1	47.1	61.6	61.6	67.0
	ZTW	46.2	49.7	51.3	62.1	68.4	**70.7**		ZTW	50.5	53.7	54.5	65.2	**68.4**	**69.0**
	EP	41.6	48.2	50.5	58.5	67.7	69.1		EP	45.6	50.9	54.7	66.4	**68.1**	68.1
	RAPNet	**54.7**	**56.2**	**60.4**	**64.9**	**69.4**	**70.8**		RAPNet	**56.9**	**59.4**	**61.7**	**67.6**	**68.4**	**69.0**
C10 92.9	SDN	74.3	76.4	77.7	87.3	91.1	**92.0**	**C10** 90.7	SDN	80.3	84.7	86.1	90.5	90.8	90.8
	PBEE	62.7	62.7	69.8	81.8	87.5	91.0		PBEE	66.7	66.7	76.3	85.9	89.7	90.9
	ZTW	75.5	78.1	80.3	88.7	91.5	**92.1**		ZTW	82.4	85.2	86.7	90.9	**91.4**	**91.4**
	EP	64.6	71.1	75.4	86.2	89.7	91.9		EP	69.6	75.3	78.4	**91.2**	**91.4**	**91.4**
	RAPNet	**83.7**	**83.9**	**85.2**	**90.8**	**92.1**	**92.1**		RAPNet	**84.1**	**87.8**	**90.7**	**91.3**	**91.5**	**91.5**

TABLE II
REPORT ON THE ACCURACY(%) OF RAPNET SCALED UP TO LARGE-SCALE DATASET IMAGENET (IMN).

Data	**Algo**	**10%**	**15%**	**25%**	**50%**	**75%**	**100%**
IMN 76.1	SDN	20.3	26.7	33.8	53.8	69.7	75.8
	PBEE	18.7	18.7	28.3	28.3	62.9	73.3
	ZTW	26.8	31.4	34.9	54.9	70.6	**76.3**
	EP	20.1	25.9	34.2	54.1	69.8	75.0
	RAPNet	**36.2**	**40.1**	**46.9**	**58.7**	**71.5**	**76.3**

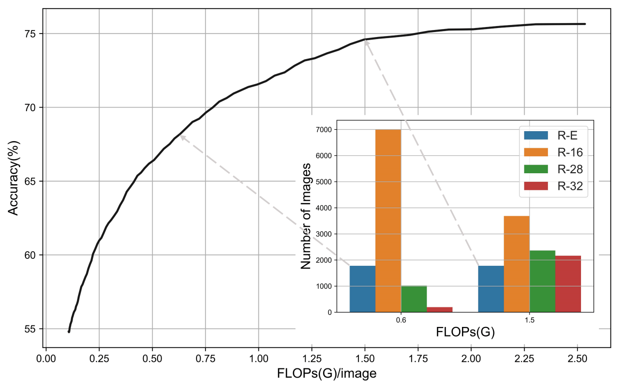


Fig. 6. No. of images correctly classified at different stages. R-E represents the number of images recognized by the prediction network, while R-m ($m \in \{16, 28, 32\}$) represents the number of images recognized at different resolutions.

threshold γ, RAPNet models can be obtained with different computational budgets. With a larger γ, more images will be recognized using a higher resolution. For instance, when the computational budget is set to 0.6 GFLOPs, approximately 70% of the total images are recognized at a low resolution of 16×16. However, as the computational budget (γ) increases, more images require recognition at higher resolutions. When the computational budget is set to 1.5 GFLOPs, the number of images recognized at a resolution of 32×32 increases from around 0.2% to 20%, while the percentage of images recognized at 16×16 resolution decreases to 38%. Therefore, increasing the value of γ incurs higher computational cost, while also increasing the number of correctly classified images at higher resolutions, resulting in a gradual improvement in accuracy. Furthermore, we have also found that there is no change in the number of images that are terminated during execution by the predictor, as the value of α remains fixed.

3) The Effect of Thresholds on RAPNet Performance: Table III presents the computational cost and accuracy results of RAPNet using the ResNet model on the T-IM dataset, considering different values of α and γ trade-offs. The exit rate (ER) denotes the rate at which execution is terminated based on α, and we emphasize the values of α and γ that yield the maximum accuracy.

From the table, we can observe that the impact of thresholds on RAPNet performance is crucial, and setting inappropriate

TABLE III
THE TRADE-OFF BETWEEN THE ACCURACY AND COMPUTATIONAL COST OF RESNET ON DIFFERENT THRESHOLDS α AND γ ON THE TINY IMAGENET DATASET.

α	ER(%)	γ	FLOPs	Acc(%)	α	ER(%)	γ	FLOPs	Acc(%)	α	ER(%)	γ	FLOPs	Acc(%)
		0.2	0.028G	44.0			0.2	0.033G	43.5			0.2	0.040G	42.7
		0.3	0.028G	44.0			0.3	0.037G	43.9			0.3	0.048G	43.4
0.1	97.4	**0.4**	0.028G	**44.0**	**0.2**	84.9	0.4	0.047G	44.9	0.3	68.4	0.4	0.071G	45.8
		0.5	0.029G	44.1			0.5	0.056G	45.9			0.5	0.089G	47.9
		0.6	0.029G	**44.1**			**0.6**	0.062G	**46.3**			0.6	0.107G	49.3
		0.2	0.046G	41.3			0.2	0.051G	40.0			0.2	0.055G	38.8
		0.3	0.046G	41.8			0.3	0.055G	41.8			0.3	0.070G	40.2
0.4	53.8	0.4	0.055G	45.9	0.5	42.0	0.4	0.087G	45.9	**0.6**	33.1	0.4	0.115G	45.7
		0.5	0.087G	49.8			0.5	0.120G	49.8			0.5	0.167G	52.0
		0.6	0.120G	**51.3**			0.6	0.144G	51.3			**0.6**	0.189G	**54.2**
		0.2	0.058G	37.6			0.2	0.062G	36.4			0.2	0.064G	35.6
		0.3	0.075G	39.0			0.3	0.082G	38.6			0.3	0.079G	37.0
0.7	25.6	0.4	0.131G	46.0	0.8	18.6	0.4	0.134G	44.7	0.9	12.2	0.4	0.142G	44.4
		0.5	0.174G	51.2			0.5	0.191G	51.3			0.5	0.204G	51.4
		0.6	0.216G	54.3			0.6	0.239G	54.2			0.6	0.249G	54.3

thresholds can lead to a very large drop in accuracy. For example, when the accuracy is approximately 55.2%, setting the appropriate α = 0.4 and γ = 0.6 makes the computational cost of RAPNet 0.120 GFLOPs. Setting the inappropriate α = 0.7 and γ = 0.5 results in a 1.45$\times$ increase in the computational cost to 0.174 GFLOPs. With the gradual increase of α, the value of ER becomes increasingly decreasing, which means that the prediction network fails to have high confidence in more samples to be recognized. Consequently, more samples will be inferred using the inference network, leading to a gradual increase in accuracy. For example, when the value of γ is set to the constant 0.6, the accuracy of RAPNet increases from 44.1% to 54.3% as α gradually increases from 0.1 to 0.9. Meanwhile, the number of images whose execution is terminated by the predicted network decreases from 97.4% to 12.2%. When the value of α is a constant, those hard samples that are not terminated by the prediction network for inference, will execute the early exit classifier predicted by the prediction network and then exit. With the increase of γ, more samples will be distributed by the prediction network to be executed by the ICs with higher accuracy in the back, and therefore the recognition accuracy is increasingly higher. For example, when α = 0.5 and γ = 0.2, 58.0% (1 - ER) of the samples will be executed by the inference network towards the front classifier and return the result, while with γ increases to 0.6, more samples will be executed by the classifier towards the back and return the result. Therefore, the accuracy increases from 40.0% to 51.3% while the computational cost increases from 0.051 to 0.144 GFLOPs.

C. Ablation Study

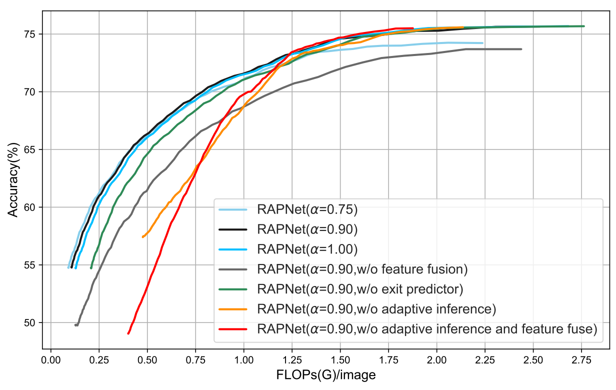
In this section, we use WideResNet as the base network on the CIFAR-100 dataset to study the effects of different designs and training methods on RAPNet performance, if not otherwise specified.

1) Impact of Different Designs on RAPNet Performance: Fig. 7 (a) presents a detailed analysis of RAPNet's performance under different combinations. Firstly, we examine the impact of different α values on RAPNet's performance. As shown in the figure, an increase in α results in more images being processed by RAPNet's early exit inference network, leading to a slight increase in computational cost. This is because a lower α value leads to more images being recognized directly by RAPNet's prediction network extracting global features, which has the least computational cost. We set α = 0.90 as default for its optimal performance.

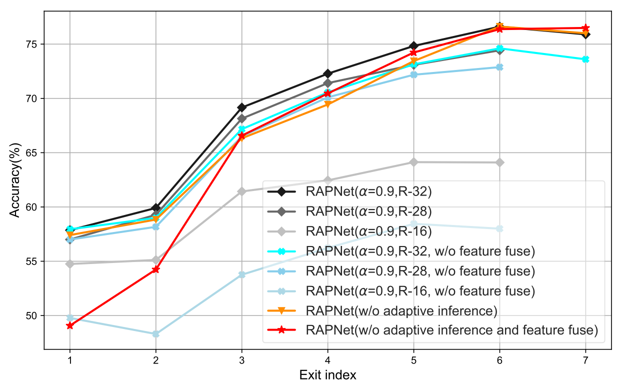
Secondly, we fix α at 0.90 and analyze the performance without the feature fusion module of RAPNet (light black curve). The results indicate that the feature fusion module has a significant influence on RAPNet's performance, particularly at lower computational budgets (below 0.5 GFLOPs), where it contributes to a performance gain of over 5%. Fig. 8 (b) further illustrates the impact of different designs on the accuracy of RAPNet's internal classifiers. We observe a similar trend to Fig. 7 (a), which remains consistent across various resolutions, with a more pronounced effect at lower resolutions. For instance, in the case of the second internal classifier using 16$\times$16 resolution (RAPNet (α=0.9, R-16) vs. RAPNet (α=0.9, R-16, w/o feature fusion)), the accuracy of RAPNet improves by 8%. These results well demonstrate the effectiveness of our design of feature fuse.

Third, we removed the exit predictor of RAPNet and evaluated its impact on RAPNet performance (green curve). We discovered that the removal of the exit predictor module results in an increase in computational cost. This is because RAPNet's exit predictor efficiently saves computation by directly and accurately predicting the optimal internal classifier for each image in the inference network. In contrast, without the exit predictor, the inference process becomes sequential, requiring the execution of all internal classifiers until a convincing prediction (e.g. sufficiently confident) has been obtained or the final classifier has been inferred. The above results effectively demonstrate that our designed exit predictor efficiently reduces computational waste by avoiding sequential execution of internal classifiers.

Fourth, we conducted an analysis on the effect of RAPNet's resolution-adaptive inference design by removing it (orange curve). We observed a significant increase in computational cost, particularly when the computational budget is below 1.25 GFLOPs. This is attributed to the presence of substantial spatial redundancy in the images, wherein using a 32$\times$32

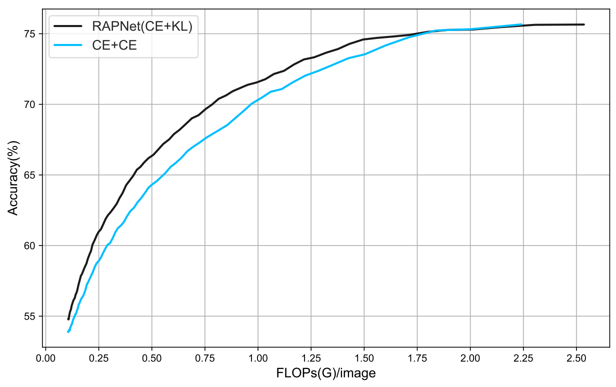


(a)

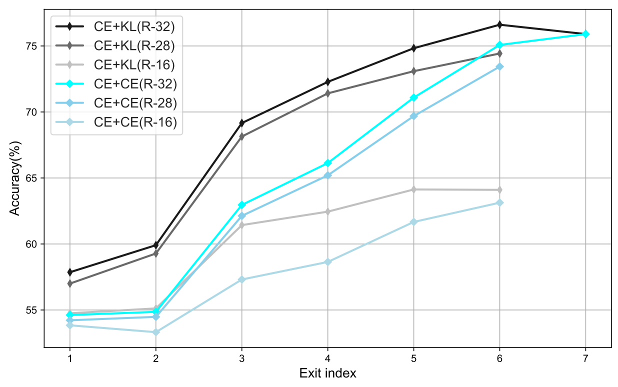


(b)

Fig. 7. Performance analysis of removing each of the RAPNet designs. (a) Accuracy comparison of different RAPNet designs. (b) Internal classifier accuracy comparison of different RAPNet designs. R-m represents the accuracy of each internal classifier within RAPNet at the corresponding resolution m. They are trained simultaneously and share the same model parameters.



(a)



(b)

Fig. 8. Performance analysis of RAPNet under different training strategies. (a) Performance comparison of RAPNet under different training strategies. (b) Internal classifier accuracy comparison of different RAPNet training strategies. R-m represents the accuracy of each internal classifier within RAPNet at the corresponding resolution m. They are trained simultaneously and share the same model parameters.

resolution to recognize a portion of easy images becomes computationally inefficient. These results strongly demonstrate that our resolution-adaptation design effectively reduces the spatial redundancy in the images.

Finally, we analyzed the impact of simultaneously removing RAPNet's resolution adaptive inference and feature fusion designs on its performance (red curve). We found that these two designs played a crucial role in improving the performance of RAPNet. When both designs were removed simultaneously, the performance of RAPNet decreased by more than 15%. Additionally, in Fig. 8 (b), we observed a significant decline in the accuracy of the shallow-layer internal classifier of the inference network when both designs were removed. For instance, after removing these two designs, the accuracy of the first internal classifier drops to 48%, which is 10% lower compared to the accuracy of the first internal classifier in RAPNet, which is 58%.

Through the above ablation study, we highlight the importance of each component in the RAPNet design. The collaborative inference enabled by RAPNet's prediction exit mechanism effectively reduces computational requirements during task inference, including both model complexity and spatial redundancy of images.

2) Impact of Training Strategy on RAPNet Performance: As shown in Eq. (2), we use $CE(,;,)$ to make the output of the final classifier of the base network fit the truth label, and $KL(,;,)$ to make the output of internal classifiers at different resolutions fit the output of the final classifier of the base network. We also try to make the output of internal classifiers at different resolutions and the output of the final classifier of the base network both fit the truth label:

$$\hat{\mathcal{L}_{cls}} = CE(\mathbf{p}_f; \mathbf{y}) + \sum_{r=m}^{M}\sum_{i=1}^{L-1} CE(\mathbf{p}_{i,m}; \mathbf{y}) \tag{4}$$

Fig. 8 (a) illustrates that the implementation of $CE + CE$ (Eq. 4) leads to a decrease in accuracy compared to the original $CE + KL$ (Eq. 2). Additionally, Fig. 8 (b) provides further evidence that the decrease in accuracy observed in the $CE + CE$ training strategy is due to lower accuracies of each internal classifier compared to $CE + KL$ at different resolutions. We choose Eq. 2 as the loss function because of the significant benefits in resolution adaptive inference stage.

3) Impact of Prediction Network Complexity on RAPNet Performance: Since for each inference, RAPNet's prediction

TABLE IV
THE PREDICTION NETWORK OF RAPNET HAS THE ACCURACY AND COMPUTATIONAL COST WHEN VARYING TRANS BLOCK DEPTH N AND EMBEDDING DIMENSION E.

N	E	**10%**	**15%**	**25%**	**50%**	**75%**	**100%**
1	24	25.0	28.2	30.5	37.1	49.9	53.5
	48	30.3	31.4	34.5	42.1	53.9	53.9
	96	-	-	34.6	42.6	48.4	53.1
2	24	37.5	37.6	40.3	46.6	50.8	54.3
	48	44.0	44.1	46.3	51.3	54.2	54.3
	96	-	-	-	45.5	47.1	50.8
3	24	43.2	43.2	45.8	49.0	53.0	54.2
	48	-	-	31.1	37.0	44.7	54.4
	96	-	-	-	-	25.8	36.9

network has to execute, its complexity is crucial for the whole architecture. We selected the ResNet-56 model on the T-IM dataset to test the impact of RAPNet's prediction network on the overall architecture performance at different complexity levels, and the reported presents in Table IV, where we use a placeholder to represent that the computational power constraint is not met.

Due to the RAPNet prediction network using an efficient VIT model like Swin Transformer, we set the head of self-attention to 8 throughout the prediction network, thus its computational cost is controlled by the depth N of the Trans Block and the embedding dimension E. These two parameters can be adapted to obtain a prediction network with different computational costs. From the table, we can observe that both embedding dimension and Trans Block are highly computationally intensive, and their variations significantly affect the accuracy and may also result in insufficient computational power. For example, when the depth of trans block N = 1 and the embedding dimension E = 96, the computational cost of the RAPNet prediction network has exceeded the constraint with low computational power constraint (less than 15%), and obviously cannot get the prediction result. Similarly, when the embedding dimension is constant, increasing the depth N, although it can greatly improve the accuracy, may also lead to the computational power not meeting the constraint. For example, when E = 48 and N are increased from 1 to 2, the average improvement in accuracy is 8.6% and the maximum up to 13.9%. However, when N is increased from 2 to 3, the computational power of less than 15% will not meet the constraint. Moreover, we also found that more complex prediction network even decreases the performance of RAPNet. Consequently, we argue that an appropriately sized prediction network is crucial for RAPNet. In this paper, through a comprehensive analysis of experimental data, we then set the depth of trans block N = 2 and embedding dimension E = 48, which can achieve the highest accuracy while the computational effort is moderate.

D. Energy Consumption on RAPNet

We choose VGG as the base model and deployed RAPNet to perform inference on the Nano device, and then used jetson_stats to measure its energy consumption at runtime. We measured the energy consumption over 100 inferences from the Tiny ImageNet test set and calculated their average as the reported energy consumption, as shown in Figure 9.

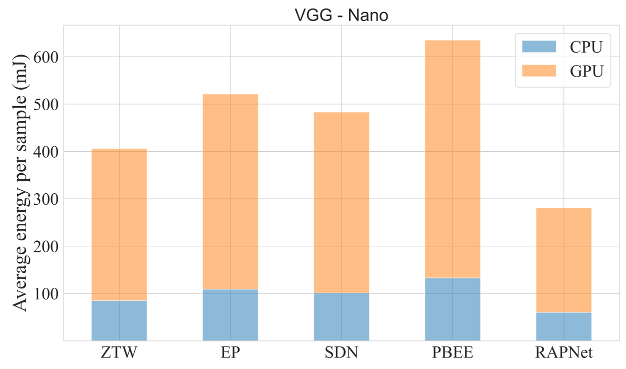


Fig. 9. The energy consumption of RAPNet vs. baselines.

From the Fig. 9, we found that RAPNet is the most energy-efficient because it precisely predicts the exit position by the predictive early exit mechanism and substantially improves the accuracy of the IC by the feature fusion mechanism, thus using the least computational cost and obtaining the lowest energy consumption is 281mJ. For the PBEE, we found that its energy consumption is 635mJ and is the most energy-consuming method due to the fact that it needs to perform at least τ (τ=2) ICs for each inference, so it performs more computational operations resulting in more energy consumption. For the EP, although it bypasses some unnecessary execution of ICs by the prediction early exit mechanism, the accuracy of the prediction network and the additional computational cost introduced by itself also impose energy consumption. With the SDN, it does not introduce any additional computational cost, so its energy consumption is lower than both EP and PBEE, which is 483mJ. Similarly, ZTW does not introduce additional computational cost, but it improves the accuracy of ICs by computational reuse so that more images will exit from the front IC when the same exit criterion is used, which saves computational operations resulting in lower energy consumption. Compared to the most energy-efficient ZTW among baselines, RAPNet achieved 1.43× the energy savings.

V. CONCLUSION

In this paper, we propose a Resolution-Adaptive Prediction Network (RAPNet) architecture, a dynamic network architecture that adaptively adapts the computational cost for each image. RAPNet is composed of a lightweight prediction network with a global feature representation of the learned input image and an early exit inference network. The prediction network can accurately predict the optimal exit position of that sample on the inference network. Furthermore, we further design resolution-adaptive inference and feature fusion modules for RAPNet by computational reuse. The former is achieved through a specialized internal classifier structure that enables the processing of input images with arbitrary resolutions, thereby reducing computational costs associated with spatial redundancy. The latter fuses the global features with the local features learned by the early exit network to augment the semantic information of the features input to the ICs, which substantially improves the accuracy of the ICs. We conduct extensive experiments across various datasets and architectures to demonstrate that RAPNet achieves a significantly better accuracy-computational trade-off than other recently proposed early exit methods. For instance, when leveraging MobileNet as the base network, RAPNet improved the accuracy of the Tiny Imagenet and CIFAR-100 datasets by 12% and 5.7% respectively, compared to the current state-of-the-art early exit methods with similar computational resources.

References

[1] A. Canziani, A. Paszke, and E. Culurciello, "An analysis of deep neural network models for practical applications," *arXiv preprint arXiv:1605.07678*, 2016.

[2] C. Shen, Y. Chen, Y. Liu and X. Guan, "Adaptive Human–Machine Interactive Behavior Analysis With Wrist-Worn Devices for Password Inference," in IEEE Transactions on Neural Networks and Learning Systems, vol. 29, no. 12, pp. 6292-6302, Dec. 2018, doi: 10.1109/TNNLS.2018.2829223.

[3] H. Kwon, C. Tong, H. Haresamudram, Y. Gao, G. D. Abowd, N. D. Lane, and T. Ploetz, "Imutube: Automatic extraction of virtual on-body accelerometry from video for human activity recognition," *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, vol. 4, no. 3, pp. 1–29, 2020.

[4] L. Chen, Y. Zhang, and L. Peng, "Metier: A deep multi-task learning based activity and user recognition model using wearable sensors," *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, vol. 4, no. 1, pp. 1–18, 2020.

[5] S. Laskaridis, S. I. Venieris, M. Almeida, I. Leontiadis, and N. D. Lane, "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.

[6] J. Cheng, J. Wu, C. Leng, Y. Wang and Q. Hu, "Quantized CNN: A Unified Approach to Accelerate and Compress Convolutional Networks," in IEEE Transactions on Neural Networks and Learning Systems, vol. 29, no. 10, pp. 4730-4743, Oct. 2018, doi: 10.1109/TNNLS.2017.2774288.

[7] S. Jiang, Z. Lin, Y. Li, Y. Shu, and Y. Liu, "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.

[8] F. Sultana, A. Sufian, and P. Dutta, "Evolution of image segmentation using deep convolutional neural network: a survey," *Knowledge-Based Systems*, vol. 201, p. 106062, 2020.

[9] M. Almeida, S. Laskaridis, I. Leontiadis, S. I. Venieris, and N. D. Lane, "Embench: Quantifying performance variations of deep neural networks across modern commodity devices," in *The 3rd international workshop on deep learning for mobile systems and applications*, 2019, pp. 1–6.

[10] Y. Hu, Z. Li, Y. Chen, Y. Cheng, Z. Cao, and J. Liu, "Content-aware adaptive device–cloud collaborative inference for object detection," *IEEE Internet of Things Journal*, vol. 10, no. 21, pp. 19 087–19 101, 2023.

[11] Y. Zhang, J.-H. Liu, C.-Y. Wang, and H.-Y. Wei, "Decomposable intelligence on cloud-edge iot framework for live video analytics," *IEEE Internet of Things Journal*, vol. 7, no. 9, pp. 8860–8873, 2020.

[12] Y.-T. Yang and H.-Y. Wei, "Edge–iot computing and networking resource allocation for decomposable deep learning inference," *IEEE Internet of Things Journal*, vol. 10, no. 6, pp. 5178–5193, 2023.

[13] Y. Chen, T. Zhao, P. Cheng, M. Ding, and C. W. Chen, "Joint front–edge–cloud iovt analytics: Resource-effective design and scheduling," *IEEE Internet of Things Journal*, vol. 9, no. 23, pp. 23 941–23 953, 2022.

[14] Q. Wang, W. Fang, and N. N. Xiong, "Tlee: Temporal-wise and layer-wise early exiting network for efficient video recognition on edge devices," *IEEE Internet of Things Journal*, pp. 1–1, 2023.

[15] L. Weng, K. Pang, M. Xia, H. Lin, M. Qian, and C. Zhu, "Sgformer: A local and global features coupling network for semantic segmentation of land cover," *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 16, pp. 6812–6824, 2023.

[16] C. Chen, C. Wang, B. Liu, C. He, L. Cong, and S. Wan, "Edge intelligence empowered vehicle detection and image segmentation for autonomous vehicles," *IEEE Transactions on Intelligent Transportation Systems*, vol. 24, no. 11, pp. 13 023–13 034, 2023.

[17] S. Wang, A. Pathania, and T. Mitra, "Neural network inference on mobile socs," *IEEE Design & Test*, vol. 37, no. 5, pp. 50–57, 2020.

[18] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Communications of the ACM*, vol. 60, no. 6, pp. 84–90, 2017.

[19] A. Cartas, M. Kocour, A. Raman, I. Leontiadis, J. Luque, N. Sastry, J. Nuñez-Martinez, D. Perino, and C. Segura, "A reality check on inference at mobile networks edge," in *Proceedings of the 2nd International Workshop on Edge Systems, Analytics and Networking*, 2019, pp. 54–59.

[20] Y. Han, G. Huang, S. Song, L. Yang, H. Wang, and Y. Wang, "Dynamic neural networks: A survey," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2021.

[21] G. Hinton, O. Vinyals, J. Dean *et al.*, "Distilling the knowledge in a neural network," *arXiv preprint arXiv:1503.02531*, vol. 2, no. 7, 2015.

[22] S. Lin, R. Ji, Y. Li, C. Deng and X. Li, "Toward Compact ConvNets via Structure-Sparsity Regularized Filter Pruning," in IEEE Transactions on Neural Networks and Learning Systems, vol. 31, no. 2, pp. 574-588, Feb. 2020, doi: 10.1109/TNNLS.2019.2906563.

[23] Z. Liu, J. Li, Z. Shen, G. Huang, S. Yan, and C. Zhang, "Learning efficient convolutional networks through network slimming," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 2736–2744.

[24] Y. Liu, Y. Sun, B. Xue, M. Zhang, G. G. Yen, and K. C. Tan, "A survey on evolutionary neural architecture search," *IEEE transactions on neural networks and learning systems*, 2021.

[25] S. Teerapittayanon, B. McDanel, and H.-T. Kung, "Branchynet: Fast inference via early exiting from deep neural networks," in *2016 23rd International Conference on Pattern Recognition (ICPR)*. IEEE, 2016, pp. 2464–2469.

[26] L. Yang, Y. Han, X. Chen, S. Song, J. Dai, and G. Huang, "Resolution adaptive networks for efficient inference," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 2369–2378.

[27] Y. Li, M. Dong, Y. Wang, and C. Xu, "Neural architecture search via proxy validation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 45, no. 6, pp. 7595–7610, 2023.

[28] S. Scardapane, M. Scarpiniti, E. Baccarelli, and A. Uncini, "Why should we add early exits to neural networks?" *Cognitive Computation*, vol. 12, no. 5, pp. 954–966, 2020.

[29] H. Li, H. Zhang, X. Qi, R. Yang, and G. Huang, "Improved techniques for training adaptive deep networks," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 1891–1900.

[30] M. Wołczyk, B. Wójcik, K. Bałazy, I. T. Podolak, J. Tabor, M. Śmieja, and T. Trzcinski, "Zero time waste: Recycling predictions in early exit neural networks," *Advances in Neural Information Processing Systems*, vol. 34, pp. 2516–2528, 2021.

[31] W. Zhou, C. Xu, T. Ge, J. McAuley, K. Xu, and F. Wei, "Bert loses patience: Fast and robust inference with early exit," *Advances in Neural Information Processing Systems*, vol. 33, pp. 18 330–18 341, 2020.

[32] Y. Kaya, S. Hong, and T. Dumitras, "Shallow-deep networks: Understanding and mitigating network overthinking," in *International conference on machine learning*. PMLR, 2019, pp. 3301–3310.

[33] R. Dong, Y. Mao, and J. Zhang, "Resource-constrained edge ai with early exit prediction," *Journal of Communications and Information Networks*, vol. 7, no. 2, pp. 122–134, 2022.

[34] K. Han, Y. Wang, H. Chen, X. Chen, J. Guo, Z. Liu, Y. Tang, A. Xiao, C. Xu, Y. Xu, Z. Yang, Y. Zhang, and D. Tao, "A survey on vision transformer," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 45, no. 1, pp. 87–110, 2023.

[35] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017.

[36] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly *et al.*, "An image is worth 16x16 words: Transformers for image recognition at scale," *arXiv preprint arXiv:2010.11929*, 2020.

[37] X. Zhu, W. Su, L. Lu, B. Li, X. Wang, and J. Dai, "Deformable detr: Deformable transformers for end-to-end object detection," *arXiv preprint arXiv:2010.04159*, 2020.

[38] B. Cheng, A. Schwing, and A. Kirillov, "Per-pixel classification is not all you need for semantic segmentation," *Advances in Neural Information Processing Systems*, vol. 34, pp. 17 864–17 875, 2021.

[39] C. Sun, A. Shrivastava, S. Singh, and A. Gupta, "Revisiting unreasonable effectiveness of data in deep learning era," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 843–852.

[40] M. Chen, M. Lin, K. Li, Y. Shen, Y. Wu, F. Chao, and R. Ji, "Coarse-to-fine vision transformer," *arXiv preprint arXiv:2203.03821*, 2022.

[41] Y. Liang, C. Ge, Z. Tong, Y. Song, J. Wang, and P. Xie, "Not all patches are what you need: Expediting vision transformers via token reorganizations," *arXiv preprint arXiv:2202.07800*, 2022.

[42] L. Li, D. Thorsley, and J. Hassoun, "Sait: Sparse vision transformers through adaptive token pruning," *arXiv preprint arXiv:2210.05832*, 2022.

[43] H. Touvron, M. Cord, M. Douze, F. Massa, A. Sablayrolles, and H. Jégou, "Training data-efficient image transformers & distillation through attention," in *International conference on machine learning*. PMLR, 2021, pp. 10 347–10 357.

[44] Z.-H. Jiang, Q. Hou, L. Yuan, D. Zhou, Y. Shi, X. Jin, A. Wang, and J. Feng, "All tokens matter: Token labeling for training better vision transformers," *Advances in neural information processing systems*, vol. 34, pp. 18 590–18 602, 2021.

[45] Z. Liu, Y. Lin, Y. Cao, H. Hu, Y. Wei, Z. Zhang, S. Lin, and B. Guo, "Swin transformer: Hierarchical vision transformer using shifted windows," in *Proceedings of the IEEE/CVF international conference on computer vision*, 2021, pp. 10 012–10 022.

[46] A. Krizhevsky, G. Hinton *et al.*, "Learning multiple layers of features from tiny images," 2009.

[47] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *2009 IEEE conference on computer vision and pattern recognition*. Ieee, 2009, pp. 248–255.

[48] Z. Abai and N. Rajmalwar, "Densenet models for tiny imagenet classification," *arXiv preprint arXiv:1904.10429*, 2019.

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

[50] X. Chen et al., "SmartDeal: Remodeling Deep Network Weights for Efficient Inference and Training," in IEEE Transactions on Neural Networks and Learning Systems, doi: 10.1109/TNNLS.2021.3138056.

[51] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," *arXiv preprint arXiv:1704.04861*, 2017.

[52] S. Zagoruyko and N. Komodakis, "Wide residual networks," *arXiv preprint arXiv:1605.07146*, 2016.

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

[54] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly *et al.*, "An image is worth 16x16 words: Transformers for image recognition at scale," *arXiv preprint arXiv:2010.11929*, 2020.

[55] Z. Liu, Y. Lin, Y. Cao, H. Hu, Y. Wei, Z. Zhang, S. Lin, and B. Guo, "Swin transformer: Hierarchical vision transformer using shifted windows," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 10 012–10 022.

[56] K. M. Abdul Kadhar and G. Anand, "Introduction to the raspberry pi," in *Data Science with Raspberry Pi*. Springer, 2021, pp. 49–78.

[57] A. A. Süzen, B. Duman, and B. Şen, "Benchmark analysis of jetson tx2, jetson nano and raspberry pi using deep-cnn," in *2020 International Congress on Human-Computer Interaction, Optimization and Robotic Applications (HORA)*. IEEE, 2020, pp. 1–5.

[58] L. Sifre and S. Mallat, "Rigid-motion scattering for texture classification," *arXiv preprint arXiv:1403.1687*, 2014.

[59] Y. Bengio, N. Léonard, and A. Courville, "Estimating or propagating gradients through stochastic neurons for conditional computation," *arXiv preprint arXiv:1308.3432*, 2013.

[60] A. Davis and I. Arel, "Low-rank approximations for conditional feedforward computation in deep neural networks," *arXiv preprint arXiv:1312.4461*, 2013.

[61] M. McGill and P. Perona, "Deciding how to decide: Dynamic routing in artificial neural networks," in *International Conference on Machine Learning*. PMLR, 2017, pp. 2363–2372.

[62] X. Wang, Y. Luo, D. Crankshaw, A. Tumanov, F. Yu, and J. E. Gonzalez, "Idk cascades: Fast deep learning by learning not to overthink," *arXiv preprint arXiv:1706.00885*, 2017.

[63] G. Huang, D. Chen, T. Li, F. Wu, L. Van Der Maaten, and K. Q. Weinberger, "Multi-scale dense networks for resource efficient image classification," *arXiv preprint arXiv:1703.09844*, 2017.

[64] J. Lin, Y. Rao, J. Lu, and J. Zhou, "Runtime neural pruning," *Advances in neural information processing systems*, vol. 30, 2017.

[65] B. E. Bejnordi, T. Blankevoort, and M. Welling, "Batch-shaping for learning conditional channel gated networks," *arXiv preprint arXiv:1907.06627*, 2019.

[66] A. Graves, "Adaptive computation time for recurrent neural networks," *arXiv preprint arXiv:1603.08983*, 2016.

[67] G. Huang, Z. Liu, G. Pleiss, L. Van Der Maaten, and K. Weinberger, "Convolutional networks with dense connectivity," *IEEE transactions on pattern analysis and machine intelligence*, 2019.

[68] T.-W. Ke, M. Maire, and S. X. Yu, "Multigrid neural architectures," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 6665–6673.

[69] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie, "Feature pyramid networks for object detection," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 2117–2125.

[70] H. Zhao, X. Qi, X. Shen, J. Shi, and J. Jia, "Icnet for real-time semantic segmentation on high-resolution images," in *Proceedings of the European conference on computer vision (ECCV)*, 2018, pp. 405–420.

[71] K. Sun, B. Xiao, D. Liu, and J. Wang, "Deep high-resolution representation learning for human pose estimation," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2019, pp. 5693–5703.

[72] S. Liu, B. Guo, K. Ma, Z. Yu, and J. Du, "Adaspring: Context-adaptive and runtime-evolutionary deep model compression for mobile applications," *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, vol. 5, no. 1, pp. 1–22, 2021.

[73] H. Wang, B. Guo, J. Liu, S. Liu, Y. Wu, and Z. Yu, "Context-aware adaptive surgery: A fast and effective framework for adaptative model partition," *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, vol. 5, no. 3, pp. 1–22, 2021.

[74] Z. Peng, W. Huang, S. Gu, L. Xie, Y. Wang, J. Jiao, and Q. Ye, "Conformer: Local features coupling global representations for visual recognition," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 367–376.

[75] M. Mao, R. Zhang, H. Zheng, T. Ma, Y. Peng, E. Ding, B. Zhang, S. Han *et al.*, "Dual-stream network for visual recognition," *Advances in Neural Information Processing Systems*, vol. 34, pp. 25 346–25 358, 2021.

[76] S. Wiedemann, K. -R. Müller and W. Samek, "Compact and Computationally Efficient Representation of Deep Neural Networks," in IEEE Transactions on Neural Networks and Learning Systems, vol. 31, no. 3, pp. 772-785, March 2020, doi: 10.1109/TNNLS.2019.2910073.