CoDiNet: Path Distribution Modeling With Consistency and Diversity for Dynamic Routing

Huanyu Wang, Zequn Qin, Songyuan Li, and Xi Li

Abstract—Dynamic routing networks, aimed at finding the best routing paths in the networks, have achieved significant improvements to neural networks in terms of accuracy and efficiency. In this paper, we see dynamic routing networks in a fresh light, formulating a routing method as a mapping from a sample space to a routing space. From the perspective of space mapping, prevalent methods of dynamic routing did not take into account how routing paths would be distributed in the routing space. Thus, we propose a novel method, termed CoDiNet, to model the relationship between a sample space and a routing space by regularizing the distribution of routing paths with the properties of consistency and diversity. In principle, the routing paths for the self-supervised similar samples should be closely distributed in the routing space. Moreover, we design a customizable dynamic routing module, which can strike a balance between accuracy and efficiency. When deployed upon ResNet models, our method achieves higher performance and effectively reduces average computational cost on four widely used datasets.

Index Terms—Routing space mapping, distribution of routing paths, the consistency regularization, the diversity regularization, dynamic routing

1 INTRODUCTION

Dynamic routing is a sample-adaptive inference mechanism for neural networks. At inference time, only part of a dynamic routing network would be activated for each sample, which is aimed at reducing computational cost with little performance compromised [1], [2], [3], [4]. In essence, dynamic routing can be considered as a mapping from a sample space to a routing space. As shown in Fig. 1, when samples are presented to a dynamic routing model, they are mapped into a routing space. Each sample’s routing path in the routing space can be represented as a binary vector, consisting of a sequence of to-be-run and to-be-skipped blocks. The model walks through the to-be-run blocks.

From the perspective of space mapping, how inference paths should be distributed in the routing space remains relatively unexplored. Since dynamic routing is related to two spaces, i.e., a sample space and a routing space, we focus on this question: how do we model the relationship between the two spaces? We expect that the distance between similar samples should be close in the routing space. Otherwise, the distance should be far. Moreover, what kind of samples are similar? We utilize the self-supervised similarity and encourage the self-supervised augmentations walking through close routing paths. Since the images obtained by self-supervised augmentations remain semantically and visually unchanged, their routing paths should be close in the routing space. In comparison, images belong to the same semantic class might be significantly different in terms of similar background, object layout, and color. The feature extraction patterns of these samples are different so that it is infeasible to utilize close routing paths for such visually different images. As shown in Fig. 2, images a and a’, images b and b’ are similar to each other, while c and c’, d and d’ are dissimilar. Therefore, the routing paths of images a and a’, images b and b’ should be the same or similar, while routing paths of images c and c’, d and d’ should be different.

To this end, we explicitly model the relationship between the sample space and the routing space, therein establishing the connection between samples and routing paths. We propose a novel dynamic routing method, termed CoDiNet, to regularize the path distribution in a routing space with the properties of consistency and diversity, based on the aforementioned space mapping. First, the consistency regularization makes augmentations of the same sample have similar feature activations, thus forming a specific routing paths for specific samples. Parameters on the specific routing path are consistently stimulated by similar samples, which is favorable to parameter sharing among similar samples and robustness of the network. At the same time, the diversity regularization makes the routing paths generated by dynamic routing more diverse, which strengthens the exploration of the network. It is evident that the more routing paths are used, the more capacity of the network is utilized.

Computational cost is another important issue because the computational ability of different platforms varies considerably. For instance, the inference speed of ResNet-50 [5] on GTX 1080ti (30fps) is much faster than that on Maxwell TitanX (18fps) at a resolution of 224 × 224. The average cost of dynamic routing networks should be customizable, when the networks are applied to platforms with different...
computational budgets. To this end, we propose a differentiable computational cost loss to optimize the average computational cost of the model and make the computational budgets customizable.

The main contributions of CoDiNet can be summarized into three parts:

- We explicitly model the relationship between the sample space and the routing space, therein establishing the connection between samples and routing paths.
- We propose a novel consistency-and-diversity regularized optimizing method modeling the relationships between the two spaces, which makes routing paths optimizable.
- We design a customizable dynamic routing module and achieve state-of-the-art results in terms of computational cost reduction and performance.

2 RELATED WORKS

In this section, we revisit relevant methods and divide them into three categories: dynamic routing networks, early prediction networks, and model compression methods. Dynamic routing and early prediction are two typical approaches to dynamic inference. The former focus on skipping unnecessary units at inference time, while the latter is characterized by multiple exits. Model compression methods are also popular for cost reduction with static inference.

2.1 Dynamic Routing Networks

Layer dropping has long been used as a regularization technique in neural networks, e.g., DropConnection [6] and Dropout [7]. Veit et al. [8] found that only short paths of deep residual networks are needed. Motivated by this, dynamic routing networks have emerged as a promising technique to skip blocks or layers at inference time for acceleration [1], [2], [3], [4]. Specifically, ConvNet-AIG [1] proposed a convolutional network that adaptively defines its inference graph conditioned on the input images. It proposes a router to make the execution decision for each convolutional block. SkipNet [2] introduced a method with LSTM gate-ways to determine whether the current block would be skipped or not. Besides, BlockDrop [3] adopted an extra policy network to sample routing paths from the whole routing space to speed up ResNets’ inference. Slimmable Nets [9] intended to train a model to support multiple widths to fit different computational constraints. Recursive network [63] proposes to execute a convolutional layer multiple times. RNR [66] models the dynamic process as a Markov decision process and uses reinforcement learning for training. Spatial dynamic convolutions for fast inference were proposed in [10], [11], [12], [13]. Multi-scale networks were introduced in [14], [64]. They learn easy samples at low resolutions, while hard samples at high resolutions. Channel-based dynamic routing methods [16], [17] were introduced as well. Recently, various dynamic methods with different kinds of selection have been proposed. Multi-kernel methods [18], [19] select different CNN kernels according to the input images.
for better performance. Recursive network [63] are introduced to reuse the networks.

At training time, dynamic routing models are prone to early convergence to suboptimal states. To deal with the issue, SkipNet [2] uses multiple training stages, Blockdrop [3] uses curriculum learning, dynamic conv [10] uses annealing, sparsity network [12] uses a non-conditional pre-training. In comparison, we focus on how to learn proper paths. To this end, we model the relation between samples and their routing paths explicitly and optimize the routing paths directly, achieving a more stable dynamic routing model.

2.2 Early Prediction Networks

While a dynamic routing network has only one exit, an early prediction network is characterized by multiple exits. In an early prediction network, the network exits once the criterion for a certain sample is satisfied. Traditional methods [20], [21] applied heuristic and greedy algorithms to reduce the executed layers. BranchlyNet [22] proposed a multiple-branch framework by attaching fully connected layers to intermediate layers of the backbone. ACT [23] proposed a halting unit for a recurrent neural network (RNN) to realize early prediction. Following ACT, SACT [24] proposed a CNN-based early prediction network, adopting a stopping unit for each point on feature maps. Since then, early prediction frameworks have been widely used in classification for efficient inference.

Considering multi-scale inputs, MSDN [25] introduced early-exit branches based on DenseNet [26]. According to the allowed time budget, McIntosh et al. [27] proposed an RNN architecture to dynamically determine the exit. Li et al. [28] proposed a self-distillation mechanism to supervise inter-layer outputs with deeper layers. Instead of bypassing residual units, DCP [29] generated decisions to save the computational cost for channels. Hydranets [30] proposed to replace the last residual block with a Mixture-of-Experts layer. Recently, methods have been adopted to other applications, such as action recognition [31], [32] and object detection [33]. Our method belongs to dynamic routing networks. Thus, our method does not have multiple exits as early prediction networks do. We also compare our method with early prediction networks in Section 4.3.2.

2.3 Model Compression Methods

Compression methods are proposed for high-performance models on platforms with limited computational resources. Knowledge distillation [34], [35], [36], [37], low-rank factorization [38], [39], [40], and quantization [41], [42], [43] have been widely used to compress the structures and to prune the parameters of neural networks. Besides, recent researches tend to prune unimportant filters or features [44], [45], [46], [47], [48] to compress or speed-up the model. They identify ineffective channels or layers by examining the magnitude of the weight or activation. The relatively ineffective channels and layers are pruned from the model. Then the pruned model is finetuned to mitigate the accuracy loss. With the iteration of pruning unnecessary parts and then finetuning the model, computational cost and model size can reduce effectively. Perforated CNN [49] speeds up the inference by skipping the computations at fixed spatial locations. In addition, Neural Architecture Search provides other technique plans achieving low-cost models including MnasNet [50], ProxylessNAS [51], EfficientNet [52], and FbNet [53]. In contrast to this line of work where the same amount of computation is applied to all samples, we focus on efficient inference by dynamically choosing a series of blocks to be executed conditioned on the input.

3 Methods

In this section, we illustrate our method CoDiNet in detail. First, we formulate dynamic routing as a space mapping and introduce the basics of dynamic routing accordingly. Second, we show how to model the relationship between the two spaces with the regularization of consistency and diversity. Third, we design a training strategy to make our dynamic routing network adaptive to different computational budgets. Finally, we illustrate our process of training and inference. For convenience, Table 1 summarizes the notations.

### Table 1

| Notations | Description |
|-----------|-------------|
| \( S \) | a sample space |
| \( s_i \) | a sample |
| \( \mathcal{R} \) | a routing space |
| \( r_i \) | the routing path for \( s_i \) |
| \( \phi \) | an \( n \)-block network |
| \( F_k \) | the \( k \)th block of \( \phi \) |
| \( a_k \) | the input of the \((k+1)\)th block of \( \phi \) |
| \( U_k \) | the router for the \( k \)th block |
| \( u_k \) | the execution decision of the \( k \)th block |
| \( v_k \) | the relaxation of \( u_k \) |
| \( m_c \) | the margin for consistency |
| \( m_d \) | the margin for diversity |
| \( \mathcal{L}_{\text{con}} \) | the loss function for consistency |
| \( \mathcal{L}_{\text{div}} \) | the loss function for diversity |
| \( \mathcal{L}_{\text{cost}} \) | the loss function for customizable dynamic routing |
| \( \alpha \) | the hyper-parameter for \( \mathcal{L}_{\text{con}} \) |
| \( \beta \) | the hyper-parameter for \( \mathcal{L}_{\text{div}} \) |
| \( \gamma \) | the hyper-parameter for \( \mathcal{L}_{\text{cost}} \) |

3.1 Dynamic Routing as a Space Mapping

3.1.1 Routing Space

We see dynamic routing as a mapping from a sample space to a routing space. A sample space \( S \) is a set of samples, and a routing space \( \mathcal{R} \) is the set of all the possible routing paths of a dynamic routing network \( \phi \). In this way, dynamic routing can be considered as a mapping \( \phi : S \rightarrow \mathcal{R} \). That is, for each sample \( s_i \in S \), its routing path \( r_i \in \mathcal{R} \) in a dynamic routing network \( \phi \) is

\[
\text{(1)}
\]

\[ r_i = \phi(s_i). \]

A routing path consists of a sequence of to-be-skipped and to-be-run blocks. In this paper, we use the same block as in ResNet [5]. Let \( \phi \) be an \( n \)-block network and \( u_k \in \{0, 1\} \) be the execution decision for the \( k \)th block, where 0 stands for to-be-skipped and 1 stands for to-be-run. Then, a routing path for the sample \( s_i \) is a concatenation of decisions for all
the blocks, i.e., \( r_i = (u_1, u_2, \ldots, u_n) \). Hence, the routing space is \( \phi = \{0, 1\}^n \), which contains \( 2^n \) routing paths in total.

### 3.1.2 Routers

For each block in the network \( \phi \), there is a router used to decide whether the block should be executed for a specific sample. Let \( F_k \) be the \( k \)th block of \( \phi \), \( a_{k-1} \) be the input of \( F_k \), and \( U_k \) be the router for the \( k \)th block. Then, the dynamic routing result \( a_k \) of the \( k \)th block is defined as

\[
a_k = u_k \cdot F_k(a_{k-1}) + (1 - u_k) \cdot a_{k-1},
\]

where the value of \( u_k \) stands for the decision of \( U_k \).

Routers are supposed to find the correct path while incurring a low computational cost. To minimize the cost, we use a lightweight structure for each router, which only contains two fully connected layers, as shown in Fig. 3b. First, the router \( U_k \) gathers information from the block input \( a_{k-1} \in \mathbb{R}^{W_k \times H_k \times C_k} \) across channels by global average pooling, which is written as

\[
z_k = \text{gap}(a_{k-1}).
\]

Then, the fused features \( z_k \) are processed by two fully connected layers sequentially. Let \( W_1 \in \mathbb{R}^{d \times C_k} \) and \( W_2 \in \mathbb{R}^{2 \times d} \) be the weights of \( U_k \)'s first layer and second layer respectively, where \( d \) denotes the output dimension of the first layer. The router \( U_k \) is defined as

\[
U_k = W_2 \circ \sigma(W_1 \circ z_k),
\]

where \( \sigma(\cdot) \) is an activation function, and \( \circ \) denotes matrix multiplication. It is worth noting that \( U_k \) is a two-element vector as a result. \( U_k[0] \) is the first element of \( U_k \), and \( U_k[1] \) is the second element. In this way, the execution decision \( u_k \) of the \( k \)th block is calculated by

\[
u_k = \arg \max_j (U_k[j]),
\]

where \( u_k = 0 \) means to-be-skipped, and \( u_k = 1 \) means to-be-run as a result. The simple yet effective structure incurs less than one percent of the computational cost of a convolution block.

#### 3.1.3 Route Relaxation

To make the binary routing decisions \( u_1, u_2, \ldots, u_n \) optimizable in an end-to-end fashion, we utilize a continuous, differentiable relaxation function, Gumbel-Softmax [54], at the end of each router as shown in Fig. 3b. Gumbel-Softmax turns discrete values into continuous ones, enabling backpropagation. Let \( g \) be the noise samples from a Gumbel distribution, and let \( T \) be the temperature which is fixed to 1 in our experiments. The relaxation for \( u_k \), called \( v_k \), is calculated by

\[
v_k = \text{softmax}\left(\frac{\log(U_k) + g}{T}\right)[1],
\]

where \( \text{softmax}()[1] \) is the second element. In this way, the dynamic routing result of the \( i \)th block is relaxed by

\[
a_k = v_k \cdot F_k(a_{k-1}) + (1 - v_k) \cdot a_{k-1}.
\]

The routing path \( r_i \) for an \( n \)-block network is also relaxed by \( r_i = (v_1, v_2, \ldots, v_n) \), where each element is in \([0, 1] \). It is worth noting that a block in \( \phi \) is either run or skipped at inference time. We will describe it in Section 3.4.

Next, we propose to regularize routing paths in the routing space.

### 3.2 Consistency and Diversity Regularization

We expect that the routing paths of similar samples should be consistent in the routing space. Otherwise, the routing paths should be diverse. In real scenarios, the similarity of different images is difficult to measure. Therefore, we propose to regularize the routing paths of samples according to their self-supervised similarity, i.e., considering the augmentations of an image as its similar samples. To this end, we first generate similar images in Section 3.2.1. To realize consistency for similar samples and diversity for dissimilar samples, we directly regularize path distributions in Sections 3.2.2 and 3.2.3. An overview of the proposed optimization method is shown in Fig. 4.

#### 3.2.1 Similar Images Generation

To obtain similar samples, we randomly augment each original sample several times by random cropping and horizontal flipping. We treat the set of augmentations as similar inputs as shown in Figs. 5a and 5b. Suppose a training batch containing \( L \) samples, and \( M \) augmentations for each sample. Then, we have an augmented set \( \{s_{i,1}, s_{i,2}, \ldots, s_{i,M}\} \) for
a sample \( s_i \), where \( s_{i,j} \) stands for the \( j \)th augmentation for \( s_i \). Therefore, there are \( L \times M \) inputs in an iteration, i.e., \( \{ s_{1,1}, s_{1,2}, \ldots, s_{1,M} \}, \ldots, \{ s_{L,1}, s_{L,2}, \ldots, s_{L,M} \} \).

### 3.2.2 Consistency Regularization

As shown in Fig. 5c, consistency regularization is an attractive force in each group of similar samples to make them closer. Let \( r_{i,j} \) be the routing path of sample \( s_{i,j} \). \( \bar{r}_C \) be the mean of \( \{ r_{1,1}, r_{1,2}, \ldots, r_{1,M} \} \), which represents the routing path center of sample \( s_i \)’s augmentations. In this way, the optimization of consistency is written as

\[
L_{\text{con}} = \frac{1}{L} \frac{1}{M} \sum_{i=1}^{L} \sum_{j=1}^{M} \left( \left\| r_{i,j} - \bar{r}_C \right\|_2^2 - m_c \right)_+, \tag{8}
\]

where \( L \) is the batch size, \( M \) is the number of augmentations, \( \| \cdot \|_2 \) is the \( L^2 \) norm, \( m_c \) is the margin for consistency, and \( [x]_+ = \max(0, x) \) denotes the hinge.

The minimization of \( L_{\text{con}} \) is to narrow down the differences between all the routing paths and the mean routing path. To further illustrate the consistency, we visualize the path distribution of five classes from the CIFAR-10 test set in Fig. 6. In the first row of Fig. 6, the paths of the vanilla dynamic routing method scatter in the routing space randomly. With consistency regularization, paths cluster around the center in the routing space, becoming consistent as shown in the second row of Fig. 6.

### 3.2.3 Diversity Regularization

Conversely, as shown in Fig. 5d, diversity regularization is a repulsive force between each group of similar samples to
push them away. The optimization of diversity is defined as
\[
L_{\text{div}} = \frac{1}{L} \sum_{i \neq j} \sum_{k \neq i} \min(m_d - \|r_{i,z} - r_{j,z}\|, 0),
\] (9)
where \(m_d\) is the margin for diversity, and \(\sum_{i \neq j}\) means all samples in the batch except \(s_i\). Within a batch, the mean routing path is optimized to maximize the differences for different groups. In this way, the routing paths of different groups are dispersed and the diversity of routing paths can be guaranteed.

Another advantage of diversity regularization is that it helps the network explore more paths. Path distribution in the real scenario under diversity regularization is also shown in Fig. 6. Compared with the first two rows, samples in the third row cluster around several centers and different clusters keep distant from each other, which is in line with our expectations.

In our method, consistency and diversity are two facets of the problem. Although \(L_{\text{con}}\) makes the routing space compact, it also runs a risk of making the whole routing space collapse into a small space, impairing the diversity of routing paths. Thus, introducing \(L_{\text{div}}\) can compensate for the disadvantage of stand-alone \(L_{\text{con}}\) and make the routing paths diverse at the same time. Similarly, \(L_{\text{div}}\) can enhance the routing space exploration; however, making routing paths scattered without constraint can be harmful to the parameter sharing among routing paths. As a result, we propose to make use of \(L_{\text{con}}\) and \(L_{\text{div}}\) together.

### 3.3 Customizable Dynamic Routing

Dynamic routing is aimed at saving the cost of a network at inference time. How much cost dynamic routing should save for a network depends on the application scenario. Since the computational budget is different from device to device, it is better to make a dynamic routing network adaptive to devices.

We design a learning strategy to make a dynamic routing network adaptive to different computational budgets. Let \(c_k\) be the computational cost of the \(k\)th block. The total cost for an \(n\)-block network is defined as
\[
\text{cost}_\text{all} = \sum_{k=1}^{n} c_k \cdot u_k.
\] (10)

To make it learnable, we use the relaxed continuous routing variable, \(v_k\), similar to Eq. (7). After relaxation, the loss function for cost optimization \(L_{\text{cost}}\) becomes
\[
L_{\text{cost}} = \sum_{k=1}^{n} c_k \cdot v_k.
\] (11)

It is worth noting that we build a computational cost lookup table which records the floating-point operations (FLOPs) of each block. During optimization, each block \(F_k\) will be assigned a cost \(c_k\) given by the lookup table.

By putting all the losses together, the overall objective of our method is
\[
L_{\text{total}} = L_{\text{cls}} + \alpha \cdot L_{\text{con}} + \beta \cdot L_{\text{div}} + \gamma \cdot L_{\text{cost}},
\] (12)
in which \(L_{\text{cls}}\) is the cross entropy loss used for classification, \(L_{\text{con}}\) is the loss defined in Eq. (8), and \(L_{\text{div}}\) is the loss defined in Eq. (9). \(\alpha, \beta,\) and \(\gamma\) are the hyper-parameters for respective losses.
To make a dynamic routing network adaptive to different computational budgets, we tune the hyper-parameter $\gamma$ of $L_{\text{cost}}$. Therefore, we can customize the learned network with different computational costs and performances. That is, a smaller $\gamma$ encourages an expensive model, and a larger $\gamma$ encourages an inexpensive model.

3.4 Training and Inference

At training time, we optimize our network in two stages. First, we train the parameters of all the blocks and all the routers together by Eq. (12). Second, we finetune our network to narrow down the decisions’ difference between training and inference. We use continuous decisions $(v_1, v_2, \ldots, v_n)$ as a relaxation of binary decisions $(u_1, u_2, \ldots, u_n)$ to make the first training stage end-to-end, but training with the continuous relaxation $v_k$ of the binary decision $u_k$ will inevitably cause a gap between training and inference. Therefore, we finetune the network with the parameters in each router fixed.

In terms of inference, a block in the network is either run or skipped exclusively. That is

$$a_k = \begin{cases} F_k(a_{k-1}), & \text{if } v_k \geq 0.5; \\ a_{k-1}, & \text{otherwise.} \end{cases}$$

4 Experiments

In this section, we conduct a series of experiments to evaluate the performance of CoDiNet. First, we introduce the experimental setup. Second, we perform the ablation studies for the proposed regularization. Third, we compare our results with the state-of-the-art works and other related methods. Next, we show qualitative analysis on the proposed modules in our method. In the end, we compare different routing strategies and structures.

4.1 Experimental Setup

4.1.1 Datasets and Metrics

We evaluate our method on four widely used datasets, which are CIFAR-10 [55], CIFAR-100 [55], SVHN [56], and ImageNet [57] (ILSVRC2012). CIFAR-10/100 consists of 60,000 colored images, which are resized to $32 \times 32$. Out of the 60,000 images, 50,000 images are used for training, and the other 10,000 images are used for testing. SVHN includes 73,257 training images and 26,032 testing images. ImageNet contains 1,281,167 training images and 50,000 validation images that are annotated with 1,000 classes and resized to $224 \times 224$. We use classification accuracy (top-1) as an evaluation metric.

4.1.2 Implementation Details

For data augmentation, we follow the settings as in [1], [2]. Images from CIFAR-10/100 are padded with 4 pixels on each side. Images from all datasets except SVHN are randomly cropped and horizontally flipped with a probability of 0.5. Those from SVHN are randomly cropped only. On CIFAR-10/100 and SVHN, the lightweight ResNets [5] are adopted as the backbones, including ResNet-32, ResNet-74 and ResNet-110. On ImageNet, ResNet-50 and ResNet-101 are adopted as the backbones. Finally, the computational is measured in GMACCs, i.e., billions of multiply-accumulate operations as used in [1], [2], [3], [4].

During training, we use SGD as an optimizer with a momentum of 0.9 and a weight decay of $1e-4$. The initial learning rate is set to 0.1 and a multi-step scheduler is adopted. On CIFAR10/100, the step-wise learning rate...
decays by 0.1 at 150 and 200 epochs. As for ImageNet, it decays by 0.1 every 30 epochs. As for the loss hyper-parameters, $\alpha$ and $\beta$ are set to 0.2 respectively. To control the computational cost precisely, the hyper-parameter $\gamma$ for $L_{\text{cost}}$ is tuned adaptive. Besides, the margin for consistency $m_c$ and the margin for diversity $m_d$ are set to 0.2 and 0.5 respectively.

### 4.2 Ablation Study

In this part, we discuss the effectiveness of each module in CoDiNet. First, we perform the ablation studies on consistency and diversity. Then, we discuss the customizable dynamic routing module, which is proposed to strike the balance between computational cost reduction and accuracy.

#### 4.2.1 Effectiveness of Regularization

First, we conduct experiments on CIFAR-10 to show the ablation studies on each component based on ResNet-110. As shown in Table 3, the accuracy of the vanilla dynamic routing method is 93.66 percent with 113 paths. With consistency regularization, limited paths are utilized, and the performance decreases to 92.88 percent. When utilizing the consistency and diversity regularization at the same time, it achieves an accuracy of 94.47 percent with 276 paths.

#### 4.2.2 Effect of Customizable Dynamic Routing

Another benefit of our method is that we can optimize the computational cost explicitly. As we discussed in Section 3.3, the proposed $L_{\text{cost}}$ can balance the trade-off between accuracy and cost. And we only need to tune the weight of $L_{\text{cost}}$, i.e., $\gamma$ in Eq. (12), to obtain a desired model. Table 2 shows the trade-off between classification accuracy, GMACCs, and the average inference time on the CIFAR-10 dataset. With the increasing of the computational cost, the accuracy tends to be upward. In the extreme case, our method achieves even only 0.10 GMACCs with a comparable accuracy, which can meet the requirements on low power platforms.

### 4.3 Performance Comparison

In this part, we compare the results of CoDiNet with related methods. First, we compare CoDiNet with the results of ResNets on CIFAR-10/100. Next, we compare CoDiNet with the state-of-the-art dynamic routing networks, early prediction models, and related compression methods on CIFAR-10 and ImageNet.

#### 4.3.1 Comparison With ResNets

We make a comparison between CoDiNet and ResNets, w.r.t. accuracy and GMACCs. As shown in Table 5, our method achieves higher accuracy with less cost in all experimental settings. In particular, compared with ResNet-110 on the CIFAR-10 dataset, our method needs 60 percent cost (0.29 GMACCs) compared to the original network (0.51 GMACCs), and achieves 0.87 percent improvement on accuracy. Similarly, on CIFAR-100, our method achieves significant improvement compared with ResNets and vanilla dynamic routing networks (without $L_{\text{con}}$ and $L_{\text{div}}$). It achieves an accuracy of 72.9 percent with 0.24 GMACCs.

---

**Table 2**

| Settings      | GMACCs | Speedup (in GMACCs) | Inference Time GTX-1080 (ms) | Inference Time GTX-1080i (ms) | Inference Time RTX-2080ti (ms) | Acc. (%) |
|---------------|--------|---------------------|------------------------------|------------------------------|-------------------------------|---------|
| ResNet-110    | 0.51   | 1.0×                | 17.2                         | 16.4                         | 15.3                          | 93.60   |
| $\gamma = 0.01$ | 0.29   | 1.8×                | 9.43                         | 8.69                         | 8.56                          | 94.47   |
| $\gamma = 0.02$ | 0.27   | 1.9×                | 8.53                         | 8.41                         | 8.16                          | 94.30   |
| $\gamma = 0.04$ | 0.22   | 2.3×                | 7.69                         | 7.62                         | 7.37                          | 93.94   |
| $\gamma = 0.08$ | 0.20   | 2.6×                | 7.66                         | 7.18                         | 6.98                          | 93.71   |
| $\gamma = 0.10$ | 0.10   | 5.1×                | 3.85                         | 3.74                         | 3.29                          | 92.45   |

GMACCs refers to billions of multiply-accumulates. $\gamma$ is the weight for $L_{\text{cost}}$ defined in Eq. (12). The unit of inference time is millisecond.

---

**Table 3**

| Methods          | Routers | $L_{\text{con}}$ | $L_{\text{div}}$ | #Path | Acc. (%) |
|------------------|---------|------------------|------------------|-------|---------|
| ResNet-110       | —       | —                | —                | 1     | 93.60   |
| Vanilla          | √       | —                | —                | 113   | 93.66   |
| Vanilla + $L_{\text{con}}$ | √       | —                | —                | 20    | 92.88   |
| Vanilla + $L_{\text{div}}$ | √       | —                | √                | 1079  | 91.34   |
| CoDiNet          | √       | √                | √                | 276   | 94.47   |

Vanilla is the vanilla dynamic routing network. The number of activated routing paths and the accuracy under different constraints based on ResNet-110.

---

**Table 4**

| Networks          | ResNet-110 | Vanilla dynamic routing | CoDiNet |
|-------------------|------------|-------------------------|---------|
| Acc. (%)          | 94.19      | 93.15                   | 94.28   |
| GMACCs            | 0.51       | 0.38                    | 0.39    |

Results are based on ResNet-110. Vanilla dynamic routing is the method without $L_{\text{con}}$ and $L_{\text{div}}$.

1. The results of ACT and SACT are quoted from [3] because ACT and SACT did not report corresponding results.
Besides, the cost reduction on deep networks is much larger than cost reduction on shallow networks. It shows that deep networks are more redundant than shallow networks.

### 4.3.2 Comparison on CIFAR-10

We compare CoDiNet with other state-of-the-art dynamic routing methods, early prediction networks, and related compression methods. As shown in Table 6, we compare with the following methods: BlockDrop [3], SkipNet [2], Conv-AIG [1], ACT [23], SACT [24], CGAP [59], DDI [60], Iamm [58], DG-Net [61]. Following [3], PFE [44] and LCCL [62] are used for comparison. BlockDrop and SkipNet are prevalent methods, applying reinforcement learning and LSTM respectively to implement the dynamic routing. BlockDrop achieves an accuracy of 93.6 percent with 0.17 GMACCs on CIFAR-10 with ResNet-110. Conversely, SkipNet focuses more on computational cost reduction, obtaining an accuracy of 92.38 percent with 0.09 GMACCs. Conv-AIG achieves an accuracy of 94.24 percent with about 0.41 GMACCs. As shown in Fig. 8a, the CoDiNet outperforms other methods in most cases with a comparable computational cost. Our method achieves an accuracy of 94.47 percent with only 0.29 GMACCs. More importantly, our method does not conflict with compression methods, and it can be used along with compression methods for better performance.

#### TABLE 6

Comparison With State-of-the-Arts on CIFAR-10

| Methods       | Backbones | GMACCs | Acc. (%) |
|---------------|-----------|--------|----------|
| baseline      |           |        |          |
| ResNet-32     | —         | 0.14   | 92.40    |
| ResNet-110    | —         | 0.50   | 93.60    |
| dynamic routing |         |        |          |
| SkipNet [2]   | ResNet-74 | 0.09   | 92.38    |
| BlockDrop [3] | ResNet-110| 0.17   | 93.60    |
| Conv-AIG [1]  | ResNet-110| 0.41   | 94.24    |
| IammNN [58]   | ResNet-101| 1.10   | 94.60    |
| CGap [59]     | ResNet-110| 0.19   | 93.43    |
| early prediction |       |        |          |
| ACT [23]      | ResNet-110| 0.38   | 93.50    |
| SACT [24]     | ResNet-110| 0.31   | 93.40    |
| DDI [60]      | ResNet-74 | 0.14   | 93.88    |
| DG-Net [61]   | ResNet-101| 3.20   | 93.99    |
| DG-Net (light)| ResNet-101| 2.22   | 91.99    |
| ours          |           |        |          |
| CoDiNet-32    | ResNet-32 | 0.09   | 92.48    |
| CoDiNet-110   | ResNet-110| 0.29   | 94.47    |

The results of the others are the best results reported in their papers. GMACCs refer to billions of multiply-accumulates.

The accuracy against computational cost (GMACCs) of CoDiNet comparing to related methods on CIFAR-10 and ImageNet.

#### TABLE 7

Comparison With State-of-the-Arts on ImageNet

| Methods        | Backbones | GMACCs | Acc. (%) |
|---------------|-----------|--------|----------|
| baseline      |           |        |          |
| ResNet-50     | —         | 3.86   | 75.36    |
| ResNet-101    | —         | 7.63   | 76.45    |
| dynamic routing |         |        |          |
| Conv-AIG 50 [1] | ResNet-50 | 3.06   | 76.18    |
| Conv-AIG 101  | ResNet-101| 5.11   | 77.37    |
| SkipNet [2]   | ResNet-101| 6.70   | 77.40    |
| SkipNet (light)| ResNet-101| 3.60   | 75.22    |
| LC-Net [65]   | ResNet-101| 2.89   | 74.10    |
| BlockDrop [3] | ResNet-101| 7.32   | 76.80    |
| DG-Net [61]   | ResNet-101| 7.05   | 76.80    |
| DDI [60]      | DenseNet-201| 3.50  | 76.50    |
| early prediction |       |        |          |
| MSDN [25]     | DenseNets | 2.30   | 74.24    |
| RA-Net [64]   | DenseNets | 2.40   | 75.10    |
| IammNN [58]   | ResNet-101| 4.00   | 69.50    |
| ACT [23]      | ResNets   | 6.70   | 75.30    |
| SACT [24]     | ResNets   | 7.20   | 75.80    |
| ours          |           |        |          |
| CoDiNet-50    | ResNet-50 | 3.10   | 76.63    |
| CoDiNet-101   | ResNet-101| 5.02   | 77.85    |

The results of the others are the best results reported in their papers. GMACCs refer to billions of multiply-accumulates.
processes different samples in different
is 0.5, our method achieves the best performance on
resulting in under-fitting.
to qualitatively ana-
is similar as
is larger than 0.5,
paths clus-
is shown in Table 8. With a
is 3.10 GMACCs based on ResNet-50 and an accuracy of
is 2.40 GMACCs. In comparison, CoDiNet outperforms
are 76.18 and 77.37 percent with 3.06 and 5.11 GMACCs
reports results based on ResNet-50 and ResNet-101, which
are 76.18 and 77.37 percent with 3.06 and 5.11 GMACCs
skipNet [2] achieves an accuracy of 75.22 per-
cases, which achieves an accuracy of 75.10 percent
with 2.40 GMACCs. In comparison, CoDiNet outperforms
these methods, which achieves an accuracy of 76.63 percent
with 3.10 GMACCs based on ResNet-50 and an accuracy of
77.85 percent with 5.02 GMACCs based on ResNet-101.

4.4 Qualitative Analysis
In this part, we conduct experiments to qualitatively ana-
lyze our proposals. First, we analyze the effect of the consist-
tency regularization and the diversity regularization. Next,
we visualize the distribution of relaxed routing paths. Finally, we show the images sharing the same routing paths.

4.4.1 Analysis of Consistency
To illustrate the effectiveness of the consistency regularization, we adopt KL divergence as an indicator to measure the difference of paths between original test set and augmented test set. That is, a smaller KL divergence indicates a better consistency. Compared with the vanilla dynamic routing on the CIFAR-10 test set, we found that the consistency regularization can significantly enhance the consistency between routing paths of original and augmented images. It also improves the performance of the augmented test set. As shown in Fig. 9, the KL divergence between original and augmented images decreases considerably with $\mathcal{L}_{con}$.

Additionally, we design a qualitative experiment to show the effect of the consistency regularization. We adopt various augmentation methods including random cropping, horizontal flipping, vertical flipping, and rotation on the 10,000 images of the CIFAR-10 test set. For each original image, we compare its routing path with that of its augmentation under two models: CoDiNet and vanilla dynamic routing. As a result, 5,342 out of 10,000 image pairs have consistent routing paths with CoDiNet, i.e., the original image and its augmented image have the same routing path. With vanilla dynamic routing, only 2,265 images have consistent paths with their augmentation.

4.4.2 Analysis of Diversity
The number of distinct routing paths at inference time under different settings of $m_d$ is shown in Table 8. With a larger margin, more routing paths will be obtained. When $m_d$ is 0.5, our method achieves the best performance on CIFAR-10 based on ResNet-110. When $m_d$ is larger than 0.5, the performance drops. The reason for that might lie in too dispersed routing paths resulting in under-fitting.

Next, we visualize the routing path distribution on the CIFAR-10 test set to demonstrate the tendency of different margins of diversity $m_d$. In Fig. 7, paths are dispersed significantly with an increase of $m_d$. When $m_d$ is 0.25, paths cluster into two groups. When $m_d$ is 0.75, the scale of coordinate is similar as $m_d = 0.5$, but the distribution is more dispersed.

4.4.3 Illustration of Routing Paths for Similar Samples
In this section, we obtain groups of similar images by applying self-supervised augmentation methods (random cropping, horizontal flipping, vertical flipping, and rotation) on the CIFAR-10 test set ten times and then visualize the distribution of routing paths for self-supervised similar images to show the routing paths of “similar samples” clustering together by directly visualizing the distribution of routing paths for self-supervised similar images. We mark the augmented images from the same raw images with the same color. As shown in Fig. 10, points with the same color, i.e., augmentations of the same sample, tend to cluster together. As a result, the routing paths of self-supervised similar samples tend to cluster together in our method.

4.4.4 Correlation Between Samples Similarity and Routing Paths Similarity
In this section, we calculate the RCC (Pearson correlation coefficient) between sample feature similarity and sample routing path similarity. A higher correlation coefficient value indicates the sample feature similarity is more po-
ositively correlated to the path similarity, therefore, the routing paths of similar images are closer in the routing space. Spec-
ifically, we use the Cosine Similarity on every pair of routing paths and sample features as the routing paths similarity and sample similarity. To better represent samples, the sample features are extracted by a third-party
unsupervised model (an ImageNet pre-trained MoCo [67]). As shown in Table 9, the Pearson correlation coefficient for our CoDiNet is 0.581, while the one for the vanilla dynamic routing is 0.024. Thus, our method is about 24 times larger than the vanilla one. Besides, we plot the correlation diagrams for different sample pairs in Fig. 12. Clearly, our method is more likely to encourage the consistency between routing path similarity and sample similarity.

4.4.5 Visualization of Relaxed Routing Paths

As shown in Fig. 11, we visualize the relaxed routing paths of the vanilla dynamic routing network and CoDiNet by t-SNE. Different colors correspond to different mathematical expectations of the numbers of to-be-run blocks. Red refers to more to-be-run blocks, while blue refers to fewer to-be-run block. The path distribution of the method without $L_{con}$ or $L_{div}$ is shown in Fig. 11a, where the paths gather in a small space around the center. In comparison, the path distribution of CoDiNet are regularly distributed and scattered throughout a much larger space as shown in Fig. 11b.

| Methods                  | Vanilla Dynamic Routing | Our CoDiNet |
|--------------------------|-------------------------|-------------|
| RCC                      | 0.024                   | 0.581       |

RCC refers to the Pearson correlation coefficient. The RCC value ranges from -1 to 1. A RCC value of 0 implies that there is no linear correlation between the similarities. Experiments is on CIFAR-10.

Moreover, we provide three groups of samples to present the routing paths of the self-supervised similar images cluster together no matter they belong to the same category or not. As shown in Fig. 11, we provide three groups of images and mark out their routing paths. First, images in the green rectangle are augmentations from the same image. Then, images in the red rectangle are from the same category. Images in the blue rectangle are from different categories. As shown in Fig. 11a, the routing paths of images in each rectangle are scattered among the whole distribution without $L_{con}$ or $L_{div}$. In comparison, with $L_{con}$ and $L_{div}$ the routing paths of images in each group are respectively cluster together as shown in Fig. 11b. Therefore, the proposed consistency regularization term can effectively make the routing paths of similar images clustering together.

![Fig. 10. The visualization of the routing paths of multi-augmentations for samples. We randomly visualize 150 groups of augmentations from the CIFAR-10 test set. Dots in the same color stand for a group of augmentations from the same sample. Best viewed in color.](image1)

![Fig. 11. Visualization of continuous routing paths by t-SNE on the CIFAR-10 test set. The color of each dot is the mathematical expectation numbers of to-be-run blocks. Red refers to more to-be-run blocks, while blue to fewer. Images in the green rectangle are augmentations from the same image. Images in the red rectangle are from the same category. Images in the blue rectangle are from different categories. Best viewed in color.](image2)

![Fig. 12. The correlation between the similarity of sample features and the similarity of sample routing paths with our CoDiNet and the vanilla dynamic routing. We use Cosine Similarity to measure the similarity. It is worth noting that the closer the point to the dotted line, the more positively correlated the similarity of samples and the similarity of paths.](image3)

![Table 9: Illustration of the RCC (Pearson Correlation Coefficient) Between Sample Similarity and the Routing Path Similarity](image4)
5 Conclusion

In this paper, we see routing mechanisms from a novel perspective that regards a dynamic routing network as a mapping from a sample space to a routing space. From this view, path distribution in routing space is a fundamental problem in a dynamic routing network. We propose a novel framework CoDiNet to regularize path distribution with diversity and consistency. Moreover, we design a customizable dynamic routing module enabling the network to adapt to different computational budgets. We compare CoDiNet with state-of-the-art methods on four benchmark datasets, demonstrating that it can effectively reduce the computational cost without compromising performance.

Acknowledgments

This work was supported in part by National Key Research and Development Program of China under Grant 2020AAA0107400, National Natural Science Foundation of China under Grant U20A20222, Zhejiang Provincial Natural Science Foundation of China under Grant LR19F020004, and key scientific technological innovation research project by Ministry of Education.

References

[1] A. Veit and S. Belongie, “Convolutional networks with adaptive inference graphs,” in Proc. Eur. Conf. Comput. Vis., 2018, pp. 1–16.
[2] W. Xin, Y. Fisher, D. Zi-Yi, D. Trevor, and G. E. Joseph, “SkipNet: Learning dynamic routing in convolutional networks,” in Proc. Eur. Conf. Comput. Vis., 2018, pp. 409–424.
[3] Z. Wu et al., “BlockDrop: Dynamic inference paths in residual networks,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit., 2018, pp. 8817–8826.
[4] A. Almahairi, N. Ballas, T. Cooijmans, Y. Zheng, H. Larochelle, and A. Courville, “Dynamic capacity networks,” in Proc. Int. Conf. Mach. Learn., 2016, pp. 8817–8826.
[5] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit., 2016, pp. 770–778.
[6] L. Wan, M. Zeiler, S. Zhang, Y. Le Cun, and R. Fergus, “Regularization of neural networks using dropConnect,” in Proc. Int. Conf. Mach. Learn., 2013, pp. 1058–1066.
[7] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, “Dropout: A simple way to prevent neural networks from overfitting,” J. Mach. Learn. Res., 2014, pp. 1929–1958.
[8] A. Veit, M. Wilber, and S. Belongie, “Residual networks behave like ensembles of relatively shallow networks,” in Proc. Adv. Neural Inf. Process. Syst., 2016, pp. 550–558.
[9] J. Yu, L. Yang, N. Xu, J. Yang, and T. Huang, “Simmable neural networks,” in Proc. Int. Conf. Learn. Representations, 2018, pp. 1–12.
[10] T. Verelst and T. Tuylelaars, “Dynamic convolutions: Exploiting spatial sparsity for faster inference,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit., 2020, pp. 2317–2326.
[11] J. Yu and T. S. Huang, “Universally simmable networks and improved training techniques,” in Proc. IEEE Int. Conf. Comput. Vis., 2019, pp. 1803–1811.
[12] F. Sun, M. Qin, T. Zhang, L. Liu, Y.-K. Chen, and Y. Xie, “Computation on sparse neural networks: An inspiration for future hardware,” 2020, arXiv:2004.11946.
[13] Z. Xie, Z. Zhang, X. Zhu, G. Huang, and S. Lin, “Spatially adaptive inference with stochastic feature sampling and interpolation,” in Proc. Eur. Conf. Comput. Vis., 2020, pp. 531–548.
[14] G. Huang, D. Chen, T. Li, F. Wu, L. van der Maaten, and K. Weinberger, “Multi-scale dense networks for resource efficient image classification,” in Proc. Int. Conf. Learn. Representations, 2018, pp. 1–14.
[15] L. Yang, Y. Han, X. Chen, S. Song, J. Dai, and G. Huang, “Resolution adaptive networks for efficient inference,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit., 2020, pp. 2369–2378.
[16] Z. Su, L. Fang, W. Kang, D. Hu, M. Pietikäinen, and L. Liu, “Dynamic group convolution for accelerating convolutional neural networks,” in Proc. Eur. Conf. Comput. Vis., 2020, pp. 1–18.
[17] A. Jordao, M. Lie, and W. R. Schwartz, “Discriminative layer pruning for convolutional neural networks,” IEEE J. Selected Top. Signal Process., vol. 14, no. 4, pp. 828–837, May 2020.
[18] Y. Chen, X. Dai, M. Lin, D. Chen, L. Yuan, and Z. Liu, “Dynamic convolution: Attention over convolution kernels,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit., 2020, pp. 11030–11039.
[19] J. Chen, X. Wang, Z. Guo, X. Zhang, and J. Sun, “Dynamic region-aware convolution,” 2020, arXiv:2003.12243.
[20] L. Reyzin, “Boosting on a budget: Sampling for feature-efficient prediction,” in Proc. Int. Conf. Mach. Learn., 2011, pp. 529–536.
[21] H. Hu, A. Grubl, A. Milos, and M. Hubert, “Efficient feature group sequencing for anytime linear prediction,” in Proc. 32nd Conf. Uncer. Artif. Intell., 2017, pp. 279–288.
[22] S. Teerapittayanon, B. McDanel, and H.-T. Kung, “BranchyNet: Fast inference via early exiting from deep neural networks,” in Proc. Int. Conf. Pattern Recognit., 2016, pp. 2464–2469.
[23] A. Graves, “Adaptive computation time for recurrent neural networks,” 2016, arXiv:1603.09883.
[24] M. Figurnov et al., “Spatially adaptive computation time for residual networks,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit., 2017, pp. 1039–1048.
[25] 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,” in Proc. Int. Conf. Learn. Representations, 2019, pp. 1–14.
[26] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, “Densely connected convolutional networks,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit., 2017, pp. 2261–2269.
[27] L. McIntosh, N. Maheswaranathan, D. Sussillo, and J. Shlens, “Recurrent segmentation for variable computational budgets,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit., 2018, pp. 1761–1770.
[28] H. Li, H. Zhang, X. Qi, R. Yang, and G. Huang, “Improved techniques for training adaptive deep networks,” in Proc. IEEE Int. Conf. Comput. Vis., 2019, pp. 1891–1900.
[29] X. Gao, Y. Zhao, L. Dudziak, R. Mullins, and C.-z. Xu, “Dynamic channel pruning: Feature boosting and suppression,” in Proc. Int. Conf. Learn. Representations, 2019, pp. 1–14.
[30] R. Teja Mullapudi, W. R. Mark, N. Shazeer, and K. Fafahalain, “Hydranets: Specialized dynamic architectures for efficient inference,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit., 2018, pp. 8080–8089.
[31] N. Hussein, M. Jain, and B. E. Bejrondi, “TimeGate: Conditional gating of segments in long-range activities,” 2020, arXiv:2004.01808.
[32] Y. Meng et al., “AR-Net: Adaptive frame resolution for efficient action recognition,” in Proc. Eur. Conf. Comput. Vis., 2020, pp. 86–104.
[33] P. Zhang, Y. Zhong, and Y. Li, “SlimYOLOv3: Narrower, faster and better for real-time UAV applications,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit., 2019, pp. 37–45.
[34] G. Hinton, O. Vinyals, and J. Dean, “Distilling the knowledge in a neural network,” in Proc. Adv. Neural Inf. Process. Syst., 2015, pp. 1–6.
[35] G. Chen, W. Choi, X. Yu, T. Han, and M. Chandraker, “Deep residual learning for image recognition,” in Proc. Adv. Neural Inf. Process. Syst., 2017, pp. 770–778.
[36] Z. Chen, L. Zhang, Z. Cao, and J. Guo, “Distilling the knowledge from handcrafted features for human activity recognition,” in IEEE Trans. Inf. Ind., vol. 14, no. 10, pp. 4334–4342, Oct. 2018.
[37] R. Yu et al., “NISP: Pruning networks using neuron importance score propagation,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit., 2018, pp. 9194–9203.
[38] Y. Ioannou, D. Robertson, J. Shotton, R. Cipolla, and A. Criminisi, “Training CNNs with low-rank filters for efficient image classification,” in Proc. Int. Conf. Learn. Representations, 2016, pp. 1–14.
[39] M. McIntosh, N. Maheswaranathan, N. Sussillo, and J. Shlens, “Convolutional neural networks with low rank expansions,” in Proc. Int. Conf. Learn. Representations, 2016, pp. 1–14.
[40] M. Jaderberg, A. Vedaldi, and A. Zisserman, “Speeding up convolutional networks with state-of-the-art methods on four benchmark datasets, demonstrating that it can effectively reduce the computational cost without compromising performance.”
Huanyu Wang received the master's degree in 2017 from the National University of Singapore, Singapore, where he worked on problems in machine learning and massive data. He is currently working toward the PhD degree with Zhejiang University. His research interests include dynamic routing, semantic segmentation, and neural architecture search.

Zequn Qin received the master's degree in 2019 from Northwestern Polytechnical University, China, where he worked on problems in computer vision and pattern recognition. He is currently working toward the PhD degree with Zhejiang University. His research interests include autonomous vehicles, dynamic routing, and semantic segmentation.

Xi Li received the PhD degree in 2009 from the National Laboratory of Pattern Recognition, Chinese Academy of Sciences, Beijing, China. From 2009 to 2010, he was a post-doctoral researcher with CNRS Telecom ParisTech, France. He was a senior researcher with the University of Adelaide, Australia. He is currently a full professor with Zhejiang University. His research interests include visual tracking, compact learning, motion analysis, face recognition, data mining, and image retrieval.

For more information on this or any other computing topic, please visit our Digital Library at www.computer.org/csdl.