SP-Net: Slowly Progressing Dynamic Inference Networks

Huanyu Wang[1*], Wenhu Zhang[2⋆], Shihao Su[1], Hui Wang[1], Zhenwei Miao[3], Xin Zhan[3], and Xi Li[1,4,5⋆⋆]

[1] College of Computer Science and Technology, Zhejiang University
[2] Polytechnic Institute, Zhejiang University
[3] Alibaba Group, HangZhou, Zhejiang, China
[4] Shanghai Institute for Advanced Study, Zhejiang University
[5] Shanghai AI Laboratory
{huanyuhello, wenhuzhang, shihaocs, wanghui_17}@zju.edu.cn, {zhenwei.mzw, zhanxin.zx}@alibaba-inc.com, xilizju@zju.edu.cn

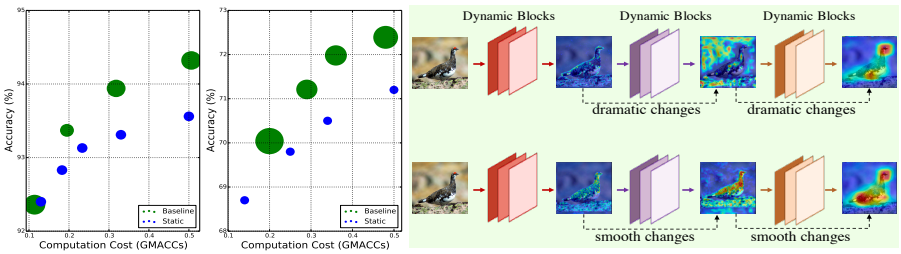
Abstract. Dynamic inference networks improve computational efficiency by executing a subset of network components, i.e., executing path, conditioned on input sample. Prevalent methods typically assign routers to computational blocks so that a computational block can be skipped or executed. However, such inference mechanisms are prone to suffer instability in the optimization of dynamic inference networks. First, a dynamic inference network is more sensitive to its routers than its computational blocks. Second, the components executed by the network vary with samples, resulting in unstable feature evolution throughout the network. To alleviate the problems above, we propose SP-Nets to slow down the progress from two aspects. First, we design a dynamic auxiliary module to slow down the progress in routers from the perspective of historical information. Moreover, we regularize the feature evolution directions across the network to smoothen the feature extraction in the aspect of information flow. As a result, we conduct extensive experiments on three widely used benchmarks and show that our proposed SP-Nets achieve state-of-the-art performance in terms of efficiency and accuracy.

Keywords: Dynamic Inference, Slowly Progressing, Executing Path Regularization, Feature Evolution Regularization.

1 Introduction

Recent years have witnessed a significant development in deep neural networks. The excellent performance of these networks is ascribed not only to the sophisticated design of network modules but also the increasing depth of the network. However, the merit comes with the price that deep neural networks are highly dependent on computational resources to ensure both efficiency and accuracy,

⋆ H. Wang and W. Zhang make equal contribution to this work.
⋆⋆ Corresponding author



(a) CIFAR-10 (b) CIFAR-100 (c) Feature Evolution

Fig. 1. (a) The stability of static networks (ResNets [12]) and vanilla dynamic inference networks on CIFAR-10. The size of dots is the variance of results. (b) The stability of static networks and vanilla dynamic inference networks on CIFAR-100. The size of dots is the variance of results. (c) Upper: The features evolve unstable throughout the network. Lower: The features evolve stable throughout the network.

limiting the deployment of these powerful models in real-world scenarios. Hence, to improve the inference efficiency, extensive efforts have been devoted to model compression methods *e.g.*, weight quantization [11,37,25], knowledge distillation [13,3,45], and low-rank approximation [16,21,17]. More often than not, these methods reduce the computational budgets at the expense of a slight drop on performance. Being different from previous streams of methods, dynamic inference methods save computational resources by dynamically assigning adequate computation conditioned on input samples to prevent performance drop. Specifically, dynamic inference networks adaptively execute part of network components and skip the rest when inferring a given sample.

A number of dynamic inference methods have been developed by assigning a router for each convolutional block. At inference time, each router makes a binary executing decision for the current block. Essentially, the binary decision alters the block between a convolution function and an identity mapping. To optimize the discrete executing decisions in an end-to-end fashion, relaxation functions, *e.g.*, Softmax and Gumbel-Softmax, are introduced in the aforementioned methods. However, such an inference mechanism makes the optimization of the dynamic inference network unstable. As shown in Fig. 1(a) and Fig. 1(b), the vanilla dynamic inference methods show larger variance than static inference methods. First, a dynamic inference network is more sensitive to its routers than its convolutional blocks. In essence, the relaxed executing decision in the training process is a coefficient in the linear combination of a convolution function and an identity mapping. Thus, a slight difference in the executing decision results in changes of each element in the output feature map while changes in the input feature impose less effect on the output. Second, since the information flow of the network is calculated in a chain, selectively executing a subset of the network components would cause the interruption of the flow. As a result, features usually evolve unstably across the whole network. For an instance, compared

with the features that evolve stable in the lower figure of Fig. 1(c), features in the upper one of Fig. 1(c) show unstable evolution throughout the network.

In this paper, we propose a slowly progressing dynamic inference network to stabilize the optimization via slowing down the progress from the following two aspects. First, we slow down the progress in routers. Considering the unbalance of sensitivity between the parameters in convolutional blocks and routers, we take advantage of the historical information by introducing a dynamic auxiliary module to provide a guidance. Specifically, the auxiliary module is implemented as a momentum-based moving average of the dynamic inference network and gives out pseudo executing paths. Guided by the pseudo executing paths, the routers are more stable against instant changes in paths, and the dynamic inference network reduces gradient variance, thus stabilizing the training procedure. Moreover, we slow down the feature evolution between blocks to ease the interruption of information flow brought by dynamic inference. In this way, the feature evolves smoothly throughout the information flow, and skipping some of the blocks brings in less drastic changes in feature maps. The most straightforward solution to the interruption brought by the varying executing components is minimizing the changes between blocks. However, such a solution cost severe harm to the performance of networks. Thus, we take an alternative strategy that restricts the direction of feature evolution to remedy the interruption of information flow in dynamic inference.

The contributions of this paper are summarized as follows:

- We propose a slowly progressing dynamic inference network, which effectively stabilizes the optimization of dynamic inference networks.
- We slow down the progress in routers by taking advantage of the information from historical iterations to solve the unbalance of sensitivity between parameters in convolutional blocks and routers.
- We slow down the feature evolution by regularizing the direction of the feature evolution, making the feature evolves smoothly throughout the network.
- We conduct experiments on three widely used benchmarks and show that our method obtains state-of-the-art results in terms of performance and efficiency.

2 Relates Work

Dynamic Inference Networks. Dynamic inference networks have emerged as a promising technique to skip blocks or layers at inference time for acceleration [32,41,38,28]. Specifically, ConvNet-AIG [32] 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 [41] introduced a method with LSTM gate-ways to determine whether the current block would be skipped or not. Besides, BlockDrop [38] adopted an extra policy network to sample executing paths from the whole routing space to speed up the inference of ResNet. Spatial dynamic convolutions for fast inference were proposed in [33,44,29,40]. Multi-scale networks were introduced in [15,43].

They learn easy samples at low resolutions, while hard samples at high resolutions. Channel-based dynamic inference methods [28] were introduced as well. Recently, various dynamic methods with different kinds of selection have been proposed. Multi-kernel methods [2] select different CNN kernels for better performance. Recursive network [10] are introduced to reuse the networks.

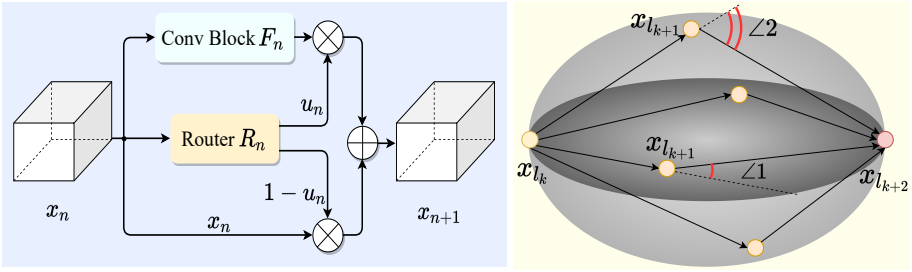
Different from these method who mainly focus on designing dynamic inference mechanisms, we pay attention to the training problem of the dynamic inference network and stabilize the dynamic inference network via slow down the process in router and smoothen feature evolution throughout the network.

Early Prediction Networks. While a dynamic inference 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. SACT [5] proposed a halting unit for a recurrent neural network (RNN) to realize early prediction, 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 [15] introduced early-exit branches. According to the allowed time budget, McIntosh et al. [22] proposed an RNN architecture to dynamically determine the exit. Li *et al.* [20] proposed a self-distillation mechanism to supervise inter-layer outputs with deeper layers. Instead of bypassing residual units, DCP [8] generated decisions to save the computational cost for channels. Hydranets [31] 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 [9,23] and object detection [48].

Our method belongs to dynamic inference networks and does not have multiple exits as early prediction networks do at inference time. However, at training time, we insert several classifiers at different stages of the network.

Knowledge Distillation Methods. Knowledge distillation proposed a concept of distillation, where a lightweight student model is trained to learn the outputs logits of an over-parameterized teacher model. Initially, knowledge distillation [13] is applied to image classification by utilizing the probabilities of each class, generated from the teacher model as soft labels for training a student model [1,14] or learning the intermediate feature maps [26,46]. Moreover, self-distillation is an extension of knowledge distillation, which leverages the network itself as a teacher. Specifically, Born-Again [7] proposes a well-trained model as a teacher to guide a student model from scratch. Furthermore, self ensemble is another way to take advantage of the knowledge distillation. For example, mean teacher [30] proposes an exponential moving average of the model weights. Besides, ensemble in temporal is introduced in [18] to regularized consistent output by a sample wised moving average on the predictions. Recently, self ensemble is also introduced in domain adaptation [42], unsupervised domain adaptation task [6], and medical images segmentation [24].

Different from these methods who conduct knowledge distillation to transfer knowledge, we introduce it for stabilizing the training of the dynamic execution



(a) Dynamic Residual Block. (b) Feature Evolution.

decisions in dynamic inference network. To our best knowledge, this is the first work to introduce the distillation in dynamic inference.

3 Method

In this section, we introduce our slowly progressing dynamic inference network (SP-Net) in detail. First, we illustrate the preliminaries of dynamic inference and overview our proposed SP-Net. Second, we introduce the dynamic auxiliary module to provide guidance. Next, we explain the feature evolution regularization. Finally, we summarizes the optimization of the proposed model.

3.1 Preliminaries and Overview

Dynamic Inference Network. Given a N-block dynamic inference network $\mathcal{F}^d$, we term the n-th convolution block F_n, where $n \in \{1, \cdots, N\}$. Generally, a dynamic inference network assigns a router to each convolutional block to decide the execution state, as shown in Fig. 2(a). R_n, the router of F_n, makes a binary executing decision u_n for the current block F_n, where u_n is either 0 or 1, representing skipping the block or executing the block respectively. Combining the execution state, the output feature x_{n+1} of the current block F_n is calculated as

$$\begin{aligned} x_{n+1} &= u_n \cdot F_n(x_n) + (1 - u_n) \cdot x_n \\ &= R_n(x_n) \cdot F_n(x_n) + (1 - R_n(x_n)) \cdot x_n. \end{aligned} \tag{1}$$

Let R_n be the router of the n-th convolutional block. To obtain a binary execution decision, R_n usually applies functions that are not continuously differentiable, *e.g.*, argmax, rounding, or sampling. Utilizing such functions hampers the backpropagation of training the network in an end-to-end fashion. Thus, during training, u_n is usually relaxed into a continuous form v_n. Specifically, $v_n \in [0, 1]$ stands for the probability of executing the convolutional block and it is obtained by some relaxation functions involving operations such as Softmax, Gumbel-Softmax, *etc.* Collecting all executing decisions for respective blocks forms an executing path for a given sample $(u_1, u_2, \cdots, u_N)$. Based on the relaxation function, a continuous executing path $(v_1, v_2, \cdots, v_N)$ is obtained accordingly.

The Proposed Framework. In our method, we improve the optimization of the dynamic inference network from two aspects: utilizing the historical information and regularizing the direction of feature evolution. The whole framework of our method is shown in Fig. 2. Our proposed framework contains a dynamic inference network, $\mathcal{F}^d$, a dynamic auxiliary module, $\mathcal{F}^t$, and several attached classifiers.

To analyze the unbalance of sensitivity, we define the parameters of routers R_n as H_n, and the parameters of convolutional blocks F_n as W_n, respectively. In this way, the Eq. (1) is reformulated as,

$$x_{n+1} = R_n(x_n, H_n) \cdot F_n(x_n, W_n) + (1 - R_n(x_n, H_n)) \cdot x_n. \tag{2}$$

The gradient of output feature x_{n+1} with respect to parameters in routers, *i.e.*, H_n and convolutional block, *i.e.*, W_n are computed as,

$$\frac{\partial x_{n+1}}{\partial H_n} = \frac{\partial R_n(\cdot)}{\partial H_n}[F_n(x_n, W_n) - x_n], \tag{3}$$

$$\frac{\partial x_{n+1}}{\partial W_n} = \frac{\partial F_n(\cdot)}{\partial W_n} R_n(x_n, H_n). \tag{4}$$

As shown in Eq. (3), the gradient with respect to router parameters is proportional to $F_n(x_n, W_n) - x_n$, which is the difference between the input feature and the output of convolutional block. Convolutional blocks are able to extract features with large variations from input, resulting in an intense fluctuation in the gradient of routers. In contrast, the gradients with respect to parameters in convolutional blocks are proportional to $R_n(x_n, H_n) \in [0, 1]$, which are a more restricted fluctuation interval. Therefore, the difference between the gradients of routers and convolutional blocks shows that they are of unbalanced sensitivity.

In order to alleviate the unbalance of sensitivity, we introduce a dynamic auxiliary module, of which the network structure is the same as the dynamic inference model. We denote the parameters of $\mathcal{F}^d$ as θ^d and those of $\mathcal{F}^t$ as θ^t. During training, the parameters in the dynamic auxiliary module are progressively updated from the dynamic inference network, in every step by

$$\theta^t = m \cdot \theta^t + (1 - m) \cdot \theta^d, \tag{5}$$

where $m \in [0, 1)$ is a momentum coefficient. In this way, the dynamic auxiliary module $\mathcal{F}^t$ provides historical information including executing decisions and predicted logits. A knowledge distillation loss is employed to transfer the historical information to $\mathcal{F}^d$. It is worth noting that, only the parameters of the dynamic inference network, *i.e.*, θ^d, are updated by back-propagation and the dynamic auxiliary module is not involved in the workflow at inference time.

Second, as defined in Eq. (1), when u_n is zero, the whole information flow would degrade into a skip connection. In this way, the information flow from x_n to x_{n+1} are interrupted, resulting in unstable feature extraction. To solve this problem, we propose to regularize the features of consecutive stages evolve in the same direction. Specifically, we attach several classifiers at stages where down-sampling is applied [47] as shown in Fig. 2. Each classifier is trained with

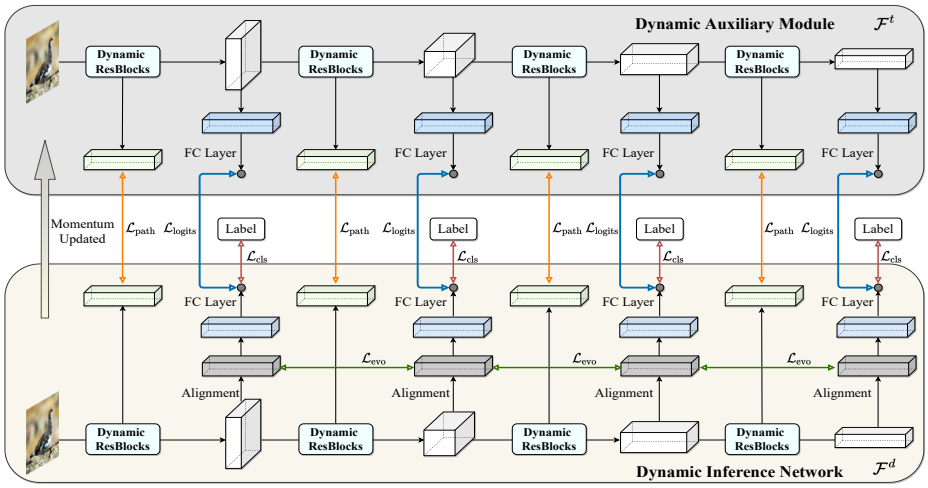


Fig. 2. Illustration of the proposed Slowly Progressing Dynamic Inference Network. The proposed framework consists of a dynamic inference network and a dynamic auxiliary module. We attach several classifiers at different stages of the network.

semantic labels of input. As a result, these classifiers capture the features of the given sample at different resolutions across the network. Then, we regularize the feature evolution angles among the features input to classifiers. These classifiers except the one at the last stage are only utilized in the training period, so there is no additional computation and parameters at inference time.

3.2 Dynamic Auxiliary Module

For convenience, we omit the superscript for dynamic inference network $\mathcal{F}^d$ and the dynamic auxiliary module $\mathcal{F}^t$ in the definition. The set of feature maps extracted by the network is denoted as $\{x_1, x_2, \cdots, x_N\}$ and the feature maps fed to the classifiers is denoted as $\{x_{l_1}, x_{l_2}, \cdots, x_{l_K}\} \subseteq \{x_1, x_2, \cdots, x_N\}$. To be more specific, x_{l_k} is the feature map at the end of the k-th stage of the network. In this way, the output classification results of the dynamic inference network and dynamic auxiliary module are

$$y_1, \cdots, y_K = c_1(x_{l_1}), \cdots, c_K(x_{l_K}), \tag{6}$$

where c_k is the k-th classifier and y_k is the logits output by the k-th classifier. As defined in Section 3.1, we term the executing path $p = (v_1, \cdots, v_N)$.

Executing Path Distillation. With the obtained executing path in the dynamic auxiliary module p^t and the executing path in the dynamic inference network p^d, we conduct an executing path distillation, defined as

$$\mathcal{L}_{\text{path}} = \|p^t - p^d\|_2^2 = \sum_{n=1}^{N} (v_n^t - v_n^d)^2, \tag{7}$$

where v_n^t and v_n^d are the executing decision of the n-th dynamic residual block in the dynamic auxiliary module and dynamic inference network respectively.

Logits Distillation. We transfer the knowledge from the outputs logits $\{y_1^t, \cdots, y_K^t\}$ of dynamic auxiliary module to the dynamic inference network by,

$$\mathcal{L}_{\text{logits}} = \sum_{k=1}^{K} \text{KL}(y_k^t \| y_k^d), \tag{8}$$

where $\text{KL}(\cdot\|\cdot)$ is the Kullback-Leibler divergence loss and K is the total number of classifiers in dynamic inference network.

3.3 Feature Evolution Direction

As defined in Eq. (1), feature evolution in a N-block network starts from the input image x_1 to the feature for classification x_N. To make feature evolution from input image to classification feature in a stable process, we propose to regularize the feature evolution in the same direction by minimizing the angles between features from consecutive stages.

Feature Evolution Angle. In most situations, the features from different stages are of different resolutions. Therefore, we introduce a fully connected layer to align them into the same size. Let a_k be the alignment layer before the k-th classifier and x_{l_k} be the input feature to the alignment layer. In this way, the feature evolution angles are calculated by the first-order difference between the aligned features of consecutive stages as follows,

$$\begin{aligned} \overrightarrow{x_{l_1}, x_{l_2}} &= a_2(x_{l_2}) - a_1(x_{l_1}), \\ &\cdots \\ \overrightarrow{x_{l_{K-1}}, x_{l_K}} &= a_K(x_{l_K}) - a_{K-1}(x_{l_{K-1}}). \end{aligned} \tag{9}$$

Then, we regularize the cosine similarity on the first-order difference of features, *i.e.*, the evolution angles, making the feature evolution in the same direction as

$$\mathcal{L}_{\text{evo}} = \frac{1}{2} \cdot \sum_{m=1}^{M-2} (1 - \cos(\overrightarrow{x_{m+2}, x_{m+1}}, \overrightarrow{x_{m+1}, x_m})), \tag{10}$$

where $\cos(\cdot, \cdot)$ is the cosine similarity. As shown in Fig. 2(b), with $\mathcal{L}_{\text{evo}}$, the feature evolution angle throughout the network would be smaller. Thus, the optimization space is limited and the feature variance throughout the network is effectively smoothened. Therefore, although part of components is skipped at inference time, the interruption brought by dynamic inference is eased.

Multi-Stage Classification. Next, to capture the semantic information along feature evolution, all classifiers are supervised by the semantic labels as

$$\mathcal{L}_{\text{cls}} = \sum_{k=1}^{K} \text{CE}(y_k^d, Y), \tag{11}$$

where $\text{CE}(\cdot, \cdot)$ is the Cross-Entropy loss and Y is the semantic label.

3.4 Optimization

Finally, we put all loss together and optimize the dynamic inference network in an end-to-end fashion, the objective function is written as

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{cls}} + \alpha \cdot \mathcal{L}_{\text{path}} + \beta \cdot \mathcal{L}_{\text{logits}} + \gamma \cdot \mathcal{L}_{\text{evo}}, \tag{12}$$

where $\mathcal{L}_{\text{cls}}$ is defined in Eq. (11), $\mathcal{L}_{\text{path}}$ is defined in Eq. (7), $\mathcal{L}_{\text{logits}}$ is defined in Eq. (8), and $\mathcal{L}_{\text{evo}}$ is defined in Eq. (10).

4 Experiment

In this section, we first introduce the experimental settings and implementation details. Second, we compare the results of our proposed method with state-of-the-arts on three benchmarks in terms of efficiency and accuracy. Next, we conduct ablation studies to validate the effectiveness of our design. Finally, we present a qualitative analysis of different designs. Our code is publicly available at https://github.com/huanyuhello/SP-Net.

4.1 Experimental Settings

Datasets and Models. We evaluate the proposed method on three popular classification benchmarks: CIFAR-10, CIFAR-100, and ImageNet. CIFAR-10/100 consists of 50,000 training images and 10,000 testing images with a resolution of 32×32 and annotated by 10/100 classes. ImageNet consists of 1,281,167 training images and 50,000 testing images with a resolution of 224×224 and annotated by 1,000 classes. We conduct extensive experiments based on ResNets, *i.e.*, ResNet-32/110 for CIFAR-10/100 and ResNet-50/101 for ImageNet. For evaluation, we utilize the top-1 metric to measure the classification accuracy and GMACCs (billions of multiply-accumulate operations) to measure the computational cost.

Implementation Details. At training time, we train the whole network for 320 epochs with a batch size of 256 on CIFAR-10/100, and 120 epochs with a batch size of 128 on ImageNet. The initial learning rate of is 0.1 with different schedules. It decreases the learning rate to its 10% at each milestone. Milestones of CIFAR-10/100 are set at epochs 150 and 250, while milestones of ImageNet are set at epochs 30, 60, and 90. Moreover, we employ stochastic gradient descent (SGD) with a momentum of 0.9 and a weight decay of 1e-4 in our method. Besides, the hyper-parameters α, β, and γ, are set to 1, 0.5, and 1e-4 in Eq. (12).

4.2 Performance Comparison

In this section, we compare the performance of SP-Net with state-of-the-arts on CIFAR and ImageNet datasets w.r.t accuracy and computational cost.

Table 1. Performance comparison with state-of-the-arts on CIFAR-10.

Methods	Backbones	GMACCs	Acc. (%)
ResNet-32 [12]	—	0.14	92.40
ResNet-110 [12]	—	0.50	93.60
dynamic inference			
SkipNet [41]	ResNet-74	0.09	92.38
BlockDrop [38]	ResNet-110	0.17	93.60
ConvAIG [32]	ResNet-110	0.41	94.24
IamNN [19]	ResNet-101	1.10	94.60
CGap [4]	ResNet-110	0.19	93.43
CoDiNet [35]	ResNet-110	0.29	94.47
RDI-Net [34]	ResNet-110	0.38	95.10
early prediction			
ACT [5]	ResNet-110	0.38	93.50
SACT [5]	ResNet-110	0.31	93.40
DDI [36]	ResNet-74	0.14	93.88
DG-Net [27]	ResNet-101	3.20	93.99
DG-Net (light)	ResNet-101	2.22	91.99
SP-Net	ResNet-110	0.46	95.22
SP-Net (light)	ResNet-110	0.13	93.79

Comparison on CIFAR-10. In this section, we compare our method with related method on CIFAR-10. We compare with baselines, dynamic inference methods including: SkipNet [41], BlockDrop [38], ConvAIG [32], CoDiNet [35], IamNN [19], CGap [4], and RDI-Net [34], and early prediction methods including: ACT [5], SACT [5], DDI [36], and DG-Net [27]. Specifically, dynamic inference methods concentrate on skipping the unnecessary blocks with a trainable routing module. Among these methods, RDI-Net achieves 95.10% accuracy with 0.38 GMACCs. CoDiNet achieves 94.47% accuracy with 0.29 GMACCs. Differently, early prediction methods define multiple classifiers and adaptively exit the network. Among these methods, ACT achieves 93.50% with 0.38 GMACCs, while SACT achieves 93.40% with 0.31 GMACCs. In comparison, we provide two versions of model. The light-weight version achieves 93.79% with 0.13 GMACCs, while the normal version achieves 95.22% accuracy with 0.46 GMACCs.

Comparison on ImageNet. In this section, we compare our method with related method on ImageNet including: ConvAIG [32], SkipNet [41], LCNet [39], BlockDrop [38], DG-Net [27], CoDiNet [35], RDI-Net [34], MSDN [15], RA-Net [43], IamNN [19], ACT [5], and SACT [5]. As shown in Table 2, the backbone network ResNet-50 achieves 75.36% with 7.72 GMACCs. In dynamic inference methods, ConvAIG achieves 76.18% with 6.12 GMACCs. SkipNet achieves 75.22% with 7.20 GMACCs. Recently, CoDiNet proposes to regularize the consistency and diversity among the executing paths, which achieves 76.63% with 6.20 GMACCs. Similarly, the RDI-Net proposes to ranking the similarity among input samples

Table 2. Performance comparison with state-of-the-arts on ImageNet.

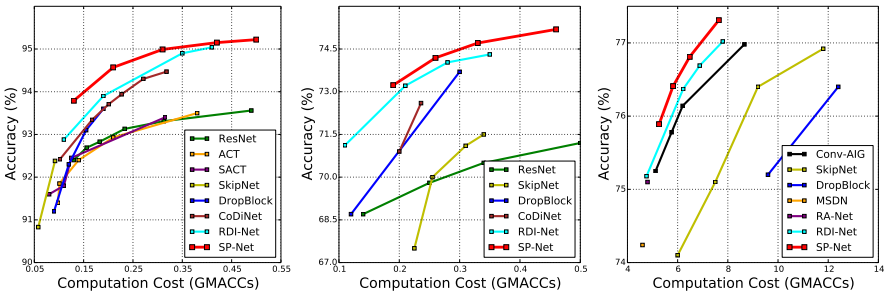
Methods	Backbones	GMACCs	Acc. (%)
ResNet-50 [12]	—	7.72	75.36
ResNet-101 [12]	—	15.26	76.45
dynamic inference			
ConvAIG [32]	ResNet-50	6.12	76.18
SkipNet [41]	ResNet-101	13.40	77.40
SkipNet (light)	ResNet-101	7.20	75.22
LCNet [39]	ResNet-50	5.78	74.10
BlockDrop [38]	ResNet-101	14.64	76.80
DG-Net [27]	ResNet-101	14.10	76.80
CoDiNet [35]	ResNet-50	6.20	76.63
RDI-Net [34]	ResNet-50	7.42	76.96
early prediction			
MSDN [15]	DenseNets	4.60	74.24
RA-Net [43]	DenseNets	4.80	75.10
IamNN [19]	ResNet-101	8.00	69.50
ACT [5]	ResNets	13.40	75.30
SACT [5]	ResNets	14.40	75.80
SP-Net	ResNet-50	7.24	77.21
SP-Net (light)	ResNet-50	5.62	76.41

and their executing paths, which achieves 76.96% with 7.42 GMACCs. In comparison, SP-Net achieves 77.21% with 7.24 GMACCs, which improves 1.75% with 0.32 GMACCs reduction than the static one.

4.3 Ablation Studies

In this part, we discuss the effectiveness of each module in the proposed method. First, we perform the ablation studies on different proposals. Then, we discuss the customizable dynamic routing module to strike the balance.

Evaluation of Proposals. As shown in Table 3, we conduct the ablation studies on our proposed modules. We conduct our method based on ResNet-110 and show a series of experiments. The baseline refers to the vanilla dynamic inference network. Component A refers to dynamic auxiliary module. Component B refers to multi-task regularization. And Component C refers to feature evolution direction regularization. Specifically, the baseline method achieves an accuracy of 93.83% with 0.47% GMACCs and 70.17% with 0.46 GMACCs on CIFAR-10 and CIFAR-100, respectively. With the dynamic auxiliary module, it increases to 94.40% under similar computational cost on CIFAR-10. Moreover, when applying the multi-stage loss, the performance improves by 0.56% on CIFAR-10 and 1.28% on CIFAR-100. Finally, putting all the components together, our method achieves 1.61% improvements with 0.05 GMACCs cost reduction on CIFAR-10 and 3.9% improvements with 0.05 GMACCs cost reduction on CIFAR-100.



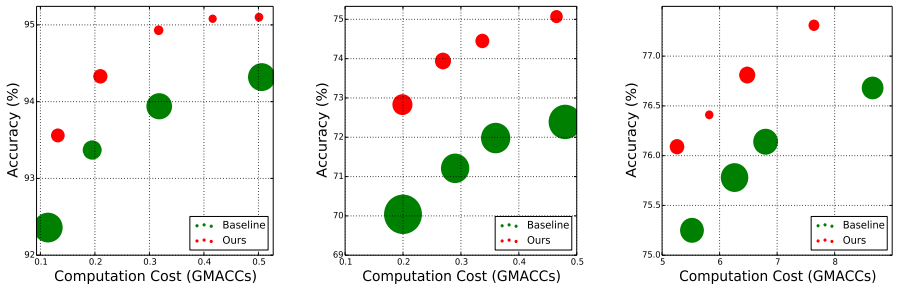
(a) CIFAR-10 (b) CIFAR-100 (c) ImageNet

Fig. 3. The accuracy against computation comparison with state-of-the-art methods.

Table 3. Ablation Studies of our method on CIFAR-10 / 100. Baseline refers to vanilla dynamic inference network. A refers to the dynamic auxiliary module. B refers to multi-stage classification. C refers to the feature evolution direction regularization.

Method	Components			CIFAR-10		CIFAR-100	
	A	B	C	Acc. (%)	GMACC	Acc. (%)	GMACC
ResNet110	—	—	—	93.61	0.51	71.24	0.51
Baseline	—	—	—	93.83	0.47	70.17	0.46
EXP-1	✓	—	—	94.40	0.48	73.63	0.47
EXP-2	✓	✓	—	95.04	0.47	74.91	0.46
EXP-3	✓	✓	✓	95.22	0.46	75.14	0.46

Customizable Computation Cost. As shown in Fig. 3, we show the performance of our method compared with the baseline under different computational costs. To enable the computational cost controllable, we apply a cost loss to adjust the executing rates of each block as $\mathcal{L}_{\text{cost}} = \sum_{n=1}^{N} \|v_n^d - t\|_2^2$, where v_n^d is the executing rates, *i.e.*, relaxed executing decision, of the n-th convolution block. t is a hyper-parameter, standing for the target executing rate of the dynamic inference network. We set t to 0.2, 0.4, 0.6, 0.8, and 0.9 on our method and the baseline method. With different target executing rates t, the accuracy varies with the computational cost accordingly. On CIFAR-10, when t is set to 0.2, our method achieves 93.75% at the computational cost of 0.14 GMACCs compared with the baseline method of 90.48% with 0.11 GMACCs. When t is set to 0.4, our method achieves 94.47% accuracy with 0.21 GMACCs, while the baseline method achieves 93.52% with 0.19 GMACCs. Similarly, on CIFAR-100, when t is set to 0.8, our method achieves 75.14% accuracy with 0.46 GMACCs compared with the baseline method of 70.17% accuracy with 0.46 GMACCs. Besides, we train each setting multiple times and demonstrate the variance of the performance as the size of the dots in Fig. 4 on CIFAR-10, CIFAR-100, and ImageNet. The variances of our SP-Nets are smaller than baseline methods, which indicates that



(a) CIFAR-10 (b) CIFAR-100 (c) ImageNet

Fig. 4. The stability and accuracy against computational cost of our method comparing to vanilla dynamic inference networks (Baselines) on CIFAR-10, CIFAR-100, and ImageNet. It is worth noting that the size of dots is the variance of results.

Table 4. Ablation Studies on the momentum coefficient, *i.e.*, m, of SP-Net on CIFAR-10 / 100. We set momentum from 0.5 to 0.8 under the same setting.

Method	Momentum	CIFAR-10		CIFAR-100	
	Coefficient	Acc. (%)	GMACCs	Acc. (%)	GMACCs
EXP-1	m=0.5	94.97	0.47	74.79	0.46
EXP-2	m=0.6	95.03	0.46	75.14	0.46
EXP-3	m=0.7	95.22	0.46	74.86	0.45
EXP-4	m=0.8	94.82	0.47	74.53	0.45

our proposed SP-Net not only improves the performance of dynamic inference but also stabilizes the training process accordingly.

4.4 Qualitative Analysis

In this section, we present a qualitative analysis of the key design of our proposed method. First, we conduct a case study on the momentum coefficient m to update the parameter in the dynamic auxiliary module. Second, we evaluate different strategies to regularize the feature evolution.

Study on Momentum Coefficient. The performance of different momentum coefficient under a similar computational cost is shown in Table 4. We perform the momentum coefficients m from 0.5 to 0.9. The accuracy increases from 94.97% with 0.47 GMACCs to 95.22% with 0.46 GMACCs with the increasing of the coefficient from 0.5 to 0.7 and then decreases on CIFAR-10. Specifically, on CIFAR-10, when m is set to 0.7, our method achieves the best performance of 95.15% with 0.46 GMACCs. Differently, on CIFAR-100, when m is set to 0.6, our method achieves the best performance of 75.14% with 0.46 GMACCs. Then performance increases from 74.79% with 0.46 GMACCs to 75.14% with the increasing of the coefficient from 0.5 to 0.6 and then decrease to 74.53% with 0.45

Table 5. Ablation Studies on different feature evolution regularization strategies on CIFAR-10 / 100. EXP-1 refers to regularization on each adjacent convolutional blocks across the whole network; EXP-2 refers to regularization on each adjacent convolutional blocks within the same resolutions of features; EXP-3 refers to regularization on different resolutions of features across the whole network.

Method	CIFAR-10		CIFAR-100	
	Acc. (%)	GMACCs	Acc. (%)	GMACCs
EXP-1	94.43	0.48	73.57	0.50
EXP-2	94.66	0.49	74.05	0.49
EXP-3	95.22	0.46	74.86	0.46

GMACCs. Empirically, the coefficient is usually set to a high value in traditional knowledge distillation. However, in the dynamic inference scenarios, since the parameters of routers are much more sensitive than those of convolutional blocks, a high momentum coefficient would lead the optimization collapse.

Study on Feature Evolution Regularization. We study different strategies of feature evolution regularization as shown in Table 5. We design three methods to regularize the feature evolution direction. EXP-1 refers to regularizing on each adjacent convolutional block across the whole network, achieving 94.43% with 0.48 GMACCs on CIFAR-10. EXP-2 refers to regularizing on each adjacent convolutional blocks within the same resolutions of features, achieving 94.66% with 0.49 GMACCs on CIFAR-10. EXP-3 refers to regularizing on different resolutions of features across the whole network, achieving 95.22% with 0.46 GMACCs on CIFAR-10. As a result, regularizing on different resolutions of features achieves the highest accuracy on both CIFAR-10 and CIFAR-100.

5 Conclusion

In this paper, we concentrate on researching an under-explored field in dynamic inference method. We propose a novel strategy called slowly progressing dynamic inference networks via stabilizing executing path and regularizing feature evolution. First, we design a dynamic auxiliary module to solve the imbalance sensitivity between parameters in routers and networks. Besides, we invite a feature evolution regularization making the features evolve smooth throughout the network. As a result, our method achieves state-of-the-art performance on three widely used benchmark in term of accuracy and computational cost reduction.

Acknowledgement

This work was supported by Alibaba Innovative Research (AIR) Program, Alibaba Research Intern Program, National Key Research and Development Program of China under Grant 2020 AAA0107400, Zhejiang Provincial Natural Science Foundation of China under Grant LR19F020004, and National Natural Science Foundation of China under Grant U20A20222.

References

1. Ba, J., Caruana, R.: Do deep nets really need to be deep? In: Proc. Advances Neural Inf. Process. Syst. pp. 2654–2662. Curran Associates, Inc. (2014)
2. Chen, Y., Dai, X., Liu, M., Chen, D., Yuan, L., Liu, Z.: Dynamic convolution: Attention over convolution kernels. In: Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (June 2020)
3. Chen, Z., Zhang, L., Cao, Z., Guo, J.: Distilling the knowledge from handcrafted features for human activity recognition. In: IEEE Trans. Indust Info. (2018)
4. Du, X., Li, Z., Ma, Y., Cao, Y.: Efficient network construction through structural plasticity. In: IEEE J. Emerging Selected Topics Circ. Syst. (2019)
5. Figurnov, M., Collins, M.D., Zhu, Y., Zhang, L., Huang, J., Vetrov, D., Salakhutdinov, R.: Spatially adaptive computation time for residual networks. In: Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (2017)
6. French, G., Mackiewicz, M., Fisher, M.H.: Self-ensembling for visual domain adaptation. In: Proc. Int. Conf. Learn. Representations (2018)
7. Furlanello, T., Lipton, Z.C., Tschannen, M., Itti, L., Anandkumar, A.: Born-again neural networks. In: Proc. Int. Conf. Mach. Learn. pp. 1602–1611 (2018)
8. Gao, X., Zhao, Y., Dudziak, L., Mullins, R., Xu, C.z.: Dynamic channel pruning: Feature boosting and suppression. In: Proc. Int. Conf. Learn. Representations (2019)
9. Ghodrati, A., Bejnordi, B.E., Habibian, A.: Frameexit: Conditional early exiting for efficient video recognition. In: Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (2021)
10. Guo, Qiushan, Z.Y.Y.W.D.L.H.Q., Yan, J.: Dynamic recursive neural network. In: Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (2019)
11. Han, S., Mao, H., Dally, W.J.: Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. In: Proc. Int. Conf. Learn. Representations (2016)
12. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (2016)
13. Hinton, G., Vinyals, O., Dean, J.: Distilling the knowledge in a neural network. In: Proc. Advances Neural Inf. Process. Syst. (2015)
14. Hinton, G., Vinyals, O., Dean, J.: Distilling the knowledge in a neural network. In: Proc. Advances Neural Inf. Process. Syst. (2015)
15. Huang, G., Chen, D., Li, T., Wu, F., van der Maaten, L., Weinberger, K.: Multi-scale dense networks for resource efficient image classification. In: Proc. Int. Conf. Learn. Representations (2018)
16. Ioannou, Y., Robertson, D., Shotton, J., Cipolla, R., Criminisi, A.: Training cnns with low-rank filters for efficient image classification. In: Proc. Int. Conf. Learn. Representations (2016)
17. Jaderberg, M., Vedaldi, A., Zisserman, A.: Speeding up convolutional neural networks with low rank expansions. In: Proc. British Mach. Vis. Conf. (2014)
18. Laine, S., Aila, T.: Temporal ensembling for semi-supervised learning. In: Proc. Int. Conf. Learn. Representations (2017)
19. Leroux, S., Molchanov, P., Simoens, P., Dhoedt, B., Breuel, T., Kautz, J.: Iamnn: Iterative and adaptive mobile neural network for efficient image classification. In: arXiv:1804.10123 (2018)
20. Li, H., Zhang, H., Qi, X., Yang, R., Huang, G.: Improved techniques for training adaptive deep networks. In: Proc. IEEE Int. Conf. Comput. Vis. (2019)

21. McIntosh, L., Maheswaranathan, N., Sussillo, D., Shlens, J.: Convolutional neural networks with low-rank regularization. In: Proc. Int. Conf. Learn. Representations (2016)
22. McIntosh, L., Maheswaranathan, N., Sussillo, D., Shlens, J.: Recurrent segmentation for variable computational budgets. In: Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (2018)
23. Meng, Y., Lin, C.C., Panda, R., Sattigeri, P., Karlinsky, L., Oliva, A., Saenko, K., Feris, R.: Ar-net: Adaptive frame resolution for efficient action recognition. In: Proc. Eur. Conf. Comput. Vis. (2020)
24. Perone, C.S., Ballester, P.L., Barros, R.C., Cohen-Adad, J.: Unsupervised domain adaptation for medical imaging segmentation with self-ensembling. In: arXiv preprint arXiv:1811.06042 (2018)
25. Polino, A., Pascanu, R., Alistarh, D.: Model compression via distillation and quantization. In: Proc. Int. Conf. Learn. Representations (2018)
26. Romero, A., Ballas, N., Kahou, S.E., Chassang, A., Gatta, C., Bengio, Y.: Fitnets: Hints for thin deep nets. In: Proc. Int. Conf. Learn. Representations (2015)
27. Shafiee, M.S., Shafiee, M.J., Wong, A.: Dynamic representations toward efficient inference on deep neural networks by decision gates. In: Proc. CVPR Workshop (2019)
28. Su, Z., Fang, L., Kang, W., Hu, D., Pietikäinen, M., Liu, L.: Dynamic group convolution for accelerating convolutional neural networks. In: Proc. Eur. Conf. Comput. Vis. (2020)
29. Sun, F., Qin, M., Zhang, T., Liu, L., Chen, Y.K., Xie, Y.: Computation on sparse neural networks: an inspiration for future hardware. In: arXiv:2004.11946 (2020)
30. Tarvainen, A., Valpola, H.: Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results. In: Proc. Advances Neural Inf. Process. Syst. pp. 1195–1204 (2017)
31. Teja Mullapudi, R., Mark, W.R., Shazeer, N., Fatahalian, K.: Hydranets: Specialized dynamic architectures for efficient inference. In: Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (2018)
32. Veit, A., Belongie, S.: Convolutional networks with adaptive inference graphs. In: Proc. Eur. Conf. Comput. Vis. (2018)
33. Verelst, T., Tuytelaars, T.: Dynamic convolutions: Exploiting spatial sparsity for faster inference. In: Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (2020)
34. Wang, H., Li, S., Su, S., Qin, Z., Li, X.: Rdi-net: Relational dynamic inference networks. In: Proc. IEEE Int. Conf. Comput. Vis. (2021)
35. Wang, H., Qin, Z., Li, S., Li, X.: Codinet: Path distribution modeling with consistency and diversity for dynamic routing. In: IEEE Trans. Pattern Anal. Mach. Intell. (2021)
36. Wang, Y., Shen, J., Hu, T.K., Xu, P., Nguyen, T., Baraniuk, R.G., Wang, Z., Lin, Y.: Dual dynamic inference: Enabling more efficient, adaptive and controllable deep inference. In: IEEE J. of Selected Topics Signal Process. (2020)
37. Wu, J., Leng, C., Wang, Y., Hu, Q., Cheng, J.: Quantized convolutional neural networks for mobile devices. In: Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (2016)
38. Wu, Z., Nagarajan, T., Kumar, A., Rennie, S., Davis, L.S., Grauman, K., Feris, R.: Blockdrop: Dynamic inference paths in residual networks. In: Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (2018)
39. Xia, Wenhan, H.Y.X.D., Jha, N.K.: Fully dynamic inference with deep neural networks. In: arXiv:2007.15151 (2020)

40. Xie, Z., Zhang, Z., Zhu, X., Huang, G., Lin, S.: Spatially adaptive inference with stochastic feature sampling and interpolation. In: Proc. Eur. Conf. Comput. Vis. (2020)
41. Xin, W., Fisher, Y., Zi-Yi, D., Trevor, D., Joseph, E.G.: Skipnet: Learning dynamic routing in convolutional networks. In: Proc. Eur. Conf. Comput. Vis. (2018)
42. Xu, Y., Du, B., Zhang, L., Zhang, Q., Wang, G., Zhang, L.: Self-ensembling attention networks: Addressing domain shift for semantic segmentation. In: Proc. AAAI Conf. Artif. Intell. pp. 5581–5588 (2019)
43. Yang, L., Han, Y., Chen, X., Song, S., Dai, J., Huang, G.: Resolution adaptive networks for efficient inference. In: Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (2020)
44. Yu, J., Huang, T.S.: Universally slimmable networks and improved training techniques. In: Proc. IEEE Int. Conf. Comput. Vis. (2019)
45. Yu, R., Li, A., Chen, C.F., Lai, J.H., Morariu, V.I., Han, X., Gao, M., Lin, C.Y., Davis, L.S.: Nisp: Pruning networks using neuron importance score propagation. In: Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (2018)
46. Zagoruyko, S., Komodakis, N.: Paying more attention to attention: Improving the performance of convolutional neural networks via attention transfer. In: Proc. Int. Conf. Learn. Representations (2017)
47. Zhang, L., Shi, Y., Shi, Z., Ma, K., Bao, C.: Task-oriented feature distillation. In: Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M.F., Lin, H. (eds.) Proc. Advances Neural Inf. Process. Syst. pp. 14759–14771 (2020)
48. Zhang, P., Zhong, Y., Li, X.: Slimyolov3: Narrower, faster and better for real-time uav applications. In: Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (2019)