RDI-Net: Relational Dynamic Inference Networks

Huanyu Wang Songyuan Li Shihao Su Zequn Qin Xi Li*
Zhejiang University
{huanyuhello, leizungjyun, shihaocs}@zju.edu.cn
zequnqin@gmail.com, xilizju@zju.edu.cn *

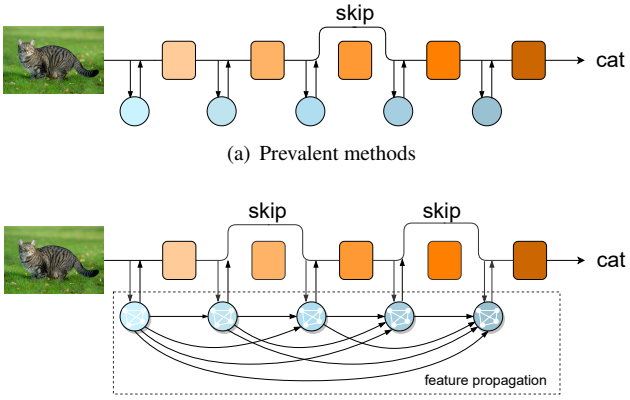
Abstract

Dynamic inference networks, aimed at promoting computational efficiency, go along an adaptive executing path for a given sample. Prevalent methods typically assign a router for each convolutional block and sequentially make block-by-block executing decisions, without considering the relations during the dynamic inference. In this paper, we model the relations for dynamic inference from two aspects: the routers and the samples. We design a novel type of router called the relational router to model the relations among routers for a given sample. In principle, the current relational router aggregates the contextual features of preceding routers by graph convolution and propagates its router features to subsequent ones, making the executing decision for the current block in a long-range manner. Furthermore, we model the relation between samples by introducing a Sample Relation Module (SRM), encouraging correlated samples to go along correlated executing paths. As a whole, we call our method the Relational Dynamic Inference Network (RDI-Net). Extensive experiments on CIFAR-10/100 and ImageNet show that RDI-Net achieves state-of-the-art performance and computational cost reduction.

1. Introduction

Recent years have witnessed a growing research interest in dynamic inference networks, which have been used in a wide range of applications, *e.g.*, image classification [40, 51, 48, 1, 41], action recognition [21, 33], and object detection [55]. Dynamic inference networks, aimed at reducing computational redundancy, execute an adaptive path for a given sample at the inference time. A typical solution [40, 44, 51] is to assign a router to each convolutional block to decide whether the current block should be executed based on the output of the last block.

In essence, prevalent routers for dynamic inference make executing decisions in a short-range manner. Since a router only takes the output of the last convolutional block as the input, an executing path is composed of a sequence of block-by-block executing decisions by individual routers. In this way, the features of the preceding routers are neglected when a subsequent router makes executing decision. Here arises a question: do the features of preceding routers benefit the decisions of subsequent ones? In this paper, we manage to explicitly model the relation between routers, enabling routers to make decisions in a long-range manner.



(a) Prevalent methods

(b) Our proposed relational routers

Figure 1. Illustration of our relational routers. (a) The traditional router (circle) only considers the output features of the last block (rectangle). (b) With a relation graph, the proposed relational router aggregates the features of preceding routers and makes the executing decision in a long-range manner. In principle, the further the distance between routers, the less the effect from preceding routers to the current one.

We propose a novel relational router to relate the current router to others: a relational router aggregates the features of preceding routers and propagates its features to subsequent ones. Specifically, the current relational router employs a router-wise directed relation graph, taking a certain amount of preceding routers into account. In the relation graph, a node stands for a relational router, and a directed edge stands for the effect of a router on another router where the further the distance between routers, the less the effect. Based on the relation graph, the current relational router

*Corresponding author

performs graph convolution [26] to aggregate the features of preceding routers and make an executing decision for the current convolutional block, as shown in Figure 1. In this way, routers make decisions in a long-range manner.

The relational routers mentioned above establish the relation between routers given a single sample. Furthermore, how about the relation between executing paths of different samples? It is intuitive that correlated samples should go along correlated executing paths. Therefore, we propose to introduce the correlation between samples to their executing paths, namely, making correlated samples go along correlated executing paths. However, because samples and their executing paths are distributed in distinct spaces [44], directly regularizing on executing paths according to the distance between samples is challenging. To address this problem, we propose an alternative solution, making the ranking of executing paths consistent with that of the distance between samples. Specifically, we present a Sample Relation Module (SRM) that measures the distance between samples and regularizes their executing paths in triplets based on the ranking of the distance, *i.e*., the closer the distance, the more similar their executing paths.

As a whole, our proposed method, namely, the Relational Dynamic Inference Network (RDI-Net), models the relation between routers and the relation between samples. The main contributions are summarized as follows:

- We design a novel type of router called relational router for dynamic inference to aggregate and propagate features of historical routers so that executing decisions are made in a long-range manner.
- Considering different samples, we model their relation and regularize their executing paths to encourage correlated samples to go along correlated executing paths.
- Extensive experiments show that the proposed method obtains the state-of-the-art results w.r.t. performance and computational cost reduction.

2. Related Work

We introduce relevant methods in three categories: dynamic inference networks, early prediction networks and model compression methods. Specifically, dynamic inference networks and early prediction networks are typical sample-adaptive methods. The former focuses on skipping part of units, while the latter is characterized by multiple exits. Differently, model compression methods concentrate on reducing the number of parameters of networks, which adopt the sample structure to all samples.

Dynamic inference networks. Dynamic inference networks emerge as a promising technique for inference acceleration [40, 51, 48, 1, 41]. Most of these methods selectively skip unnecessary network components by specially designed modules, *e.g*., the gates in Conv-AIG [40], SkipNet [51] and the policy network in BlockDrop [48]. Spatial dynamic convolutions are proposed in [42, 53, 37, 50, 8] by masking regions in a feature map. Channel-based dynamic routing methods [36, 25] are introduced as well. They selectively drop channels according to input samples. Recently performance-oriented methods employ the idea of dynamic inference. Multi-kernel methods [4] select different CNN kernels. DR-ResNet [11] recursively utilized convolutional layers. The dynamic inference is also applied to other applications, such as action recognition [21, 33] and object detection [55]. Different from these approaches which focus on enabling the dynamic inference on different tasks, we explore dynamic inference from the perspective of modeling the relations between the routers and between the samples.

Table 1. Notations

F_n	the n-th block of a dynamic network
x_n	the input of the F_n and the output of F_{n-1}
R_n	the n-th router feature extractor
u_n	the discrete executing decision of F_n
v_n	the continuous relaxation of u_n
s_n	the features of the relational router of F_n
r	the number of earlier routers being considered
γ	the decay rate of a router affects the next one
X_i	the i-th sample in a training batch
P_i	the executing path of X_i
$\mathcal{D}$	the distance matrix of samples
$\mathcal{M}$	the sorted indices of matrix $\mathcal{D}$

Early prediction networks. Early prediction networks are characterized by multiple exits. The networks exit once a criterion is satisfied at an intermediate layer. Cascade detectors [7, 43] are the earliest methods that exploit this idea in computer vision. Initially, methods like [35, 17, 10, 8, 29] proposed a halting unit to realize early prediction. Considering multi-scale inputs, [32, 18, 52] introduced early-exit branches based on DenseNet [20]. Instead of bypassing residual units, the methods [9, 49] generated decisions to save the computational cost for channels. Different from our method, these techniques dynamically execute different modules of a network model by different exits. We also compare RDI-Net with these methods in Section 4.2.

Model compression methods. Compression methods are proposed for reducing the number of parameters of heavy models with little performance compromised. Knowledge distillation [16, 3, 5, 54], low-rank factorization [22, 31, 23], and quantization [13, 47, 34] have been widely used to compress the structures and to prune the parameters of neural networks. Besides, recent research tends to prune unimpor-

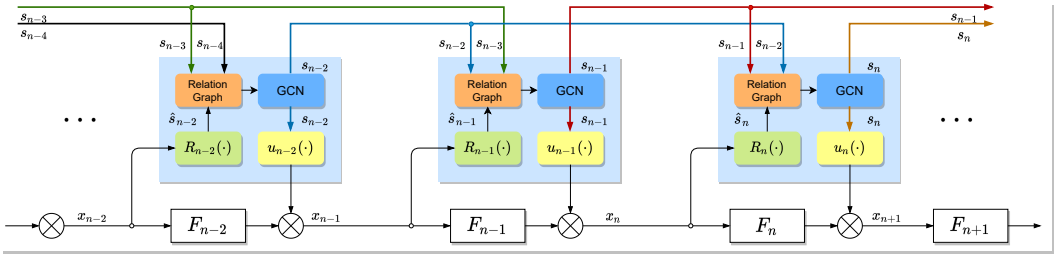


Figure 2. Illustration of feature propagation considering two previous routers ($r = 2$). The router features of the preceding two relational routers are aggregated to the current one. For example, the n-th relational router aggregates router features s_{n-1} and s_{n-2}.

tant filters or features [28, 15, 30, 45, 19] to compress or speed up the model. They identify ineffective channels or layers by examining the magnitude of the weight or activation. Recently, Neural Architecture Search achieves low-cost models as well, including MnasNet [38], Proxyless-NAS [2], EfficientNet [39], and FbNet [46]. In contrast to these works, we concentrate on dynamically deciding the network topology according to different samples at inference time.

3. Methods

In this section, we illustrate our Relational Dynamic Inference Network (RDI-Net) in detail. First, we formulate the problem of dynamic inference. Then, we establish the relations for dynamic inference from two aspects: we explicitly model the relation between routers with the relational routers and design the Sample Relation Module (SRM) for modeling the relation between samples. Finally, we illustrate the optimization strategy. For convenience, Table 1 summarizes the notations we use.

3.1. Problem Formulation

A dynamic inference network typically assigns a router to make an executing decision for each convolutional block depending on the sample. Formally, given an N-block network $\mathcal{F}$, let F_n be the n-th block, where $n \in \{1, \cdots, N\}$. The router of F_n makes a binary executing decision u_n for F_n; u_n is either 0 (skip the block) or 1 (execute the block). Then, given the input features of F_n, denoted by x_n, the output features are

$$x_{n+1} = u_n \cdot F_n(x_n) + (1 - u_n) \cdot x_n. \tag{1}$$

The routers of prevalent dynamic inference networks make executing decisions in a short-range manner. For instance, the router of F_n inputs the output features of the last block, *i.e.*, x_n, obtains its router features, denoted by s_n, and makes executing decision u_n. Let $R_n(\cdot)$ be the router feature extractor, and $U(\cdot)$ be the function that converts router features into a binary executing decision. Then, the executing decision u_n for F_n is

$$u_n := U_n(s_n), \text{where } s_n = R_n(x_n). \tag{2}$$

In this way, the executing path P of a sample X consists of a sequence of isolated executing decisions $(u_1, u_2, \cdots, u_N)$.

We propose to establish the relation between routers, making decisions in a long-range manner. The router of F_n considers not only x_n but also the router features $s_{n-1}, \cdots, s_{n-r}$ of r preceding routers. Then, the executing decision u_n becomes

$$u_n := U_n(s_n), \text{where } s_n = R_n(x_n, s_{n-1}, \cdots, s_{n-r}). \tag{3}$$

3.2. Modeling Relation between Routers

In this section, we design relational routers to model the relation between routers. A relational router employs *a router-wise relation graph* to represent the relation between routers. Based on the relation graph, the relational router performs graph convolution to *aggregate features* of preceding routers to the current router. Finally, with the aggregated feature, the relational router *makes the executing decision* for the current block.

Router-wise relation graphs. To model the effects of preceding routers on subsequent ones, we build a directed weighted graph where a node stands for a router and an edge for the effect of a router on another one. Suppose that the effect of a router on subsequent routers decays at a constant rate $\gamma \in [0, 1]$. Then, the effect that the m-th router affects the n-th router is γ^{n-m}. Taking r preceding routers into consideration, the router-wise relation graph can be represented as an adjacency matrix $\mathcal{A} \in [0, 1]^{(r+1)\times(r+1)}$, where

$$\mathcal{A}_{m,n} = \begin{cases} \gamma^{n-m} & n \geq m, \\ 0 & n < m. \end{cases} \tag{4}$$

Note that the effects of subsequent routers on preceding ones are assigned to zeros because the subsequent executing decisions cannot affect preceding routers. To clarify special cases, if r is larger than n, the dimension of $\mathcal{A}$ would be $n \times n$. When $\gamma = 0$, it means that there is no relation between routers, resulting in a short-range manner.

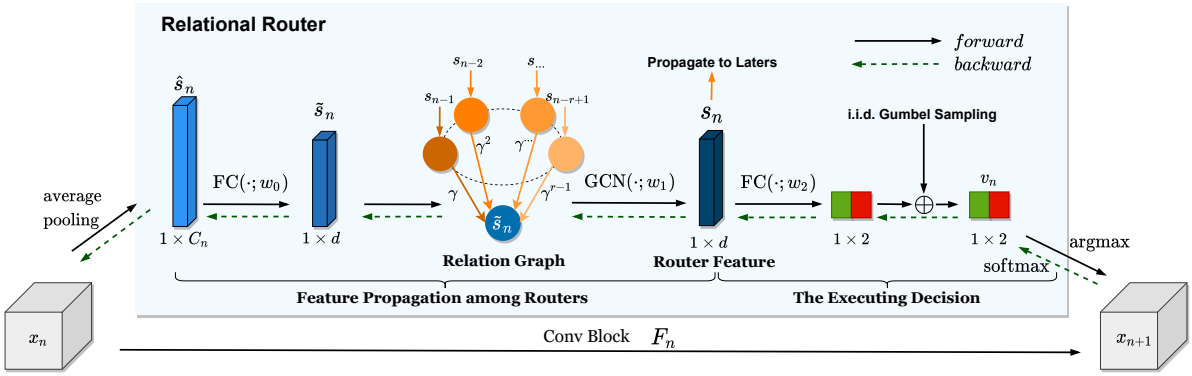


Figure 3. Overview of the relational router for the n-th convolution block. **Feature propagation among routers**: First, we apply global average pooling to x_n. Then, we align the pooling result $\hat{s}_n$ by an FC layer and get $\tilde{s}_n$. Next, we gather and aggregate router features $s_{n-r}, \cdots, s_{n-1}$ from previous relational routers and obtain s_n. **The executing decision**: We make the executing decision based on the router feature s_n. Orange arrows represent router feature propagation. Best viewed in color.

Feature propagation among routers. Based on the router-wise relation graph $\mathcal{A}$, we propagate the features of preceding routers to the current one. First, we apply global average pooling to the output of the last block, x_n, as follows

$$\hat{s}_n = \frac{1}{H_n \times W_n} \sum_{h=1}^{H_n} \sum_{w=1}^{W_n} x_n[h, w, C_n], \tag{5}$$

where C_n is the channel dimension of x_n, and $\hat{s}_n$ is of $1 \times C_n$ dimension. To enable the propagation over the relation graph, we align the dimension of $\hat{s}_n$ to $1 \times d$ through a fully-connected (FC) layer as

$$\tilde{s}_n = g(\mathrm{FC}(\hat{s}_n; w_n^0)), \tag{6}$$

where $w_n^0 \in \mathbb{R}^{C_n \times d}$, and $g(\cdot)$ denotes a nonlinear activation function. Next, we collect the features $s_{n-r}, \cdots, s_{n-1}$ of r preceding routers and stack them with $\tilde{s}_n$. Let $S_n = (s_{n-r}, \cdots, s_{n-1}, \tilde{s}_n)$ be the matrix of stacked features. Then, we aggregate features through a graph convolution network (GCN) [26],

$$s_n = g(\mathrm{GCN}(S_n; \mathcal{A}, w_n^1))[r, :] = g(\mathcal{A} \circ S_n \circ w_n^1)[r, :], \tag{7}$$

where $w_n^1 \in \mathbb{R}^{d \times d}$ denotes the trainable weights of the GCN layer and $(\circ)$ denotes matrix multiplication. Moreover, It's worth noting that s_n in $1 \times d$ dimension will be propagated to subsequent routers.

The executing decision. A relational router makes an executing decision for its block with the aggregated router feature, as shown in Figure 3. To optimize the binary decision in an end-to-end fashion, we relax it into a continuous form, denoted by $v_n \in [0, 1]$. First, the router feature, s_n, goes through a fully-connected layer. Next, with a relaxation function Gumbel-Softmax [24], we obtain the executing decision for the current block. Let $w_n^2 \in \mathbb{R}^{d \times 2}$ be the parameters of the fully-connected layer. Thus, the relaxed decision v_n is calculated by

$$v_n = \mathrm{softmax}(\log(\mathrm{FC}(s_n; w_n^2) + G)/\tau)[1], \tag{8}$$

where G is the Gumbel sampling and τ is the temperature.

With the relaxed executing decision v_n, the output of the n-th block in dynamic inference at training time is represented as

$$x_{n+1} = v_n \cdot F_n(x_n) + (1 - v_n) \cdot x_n. \tag{9}$$

In this way, we establish the relation between routers and make executing decisions in a long-range manner.

3.3. Modeling Relation between Samples

The routers designed above model the relation between routers. Based on the relational routers, the executing path P for the sample X is relaxed by $(v_1, v_2, \cdots, v_N)$ at the training time. Furthermore, we explore the relation between different samples and design the sample relation module (SRM) to regularize the executing paths w.r.t. correlated samples. First, we measure the distance between samples. Then, we encourage correlated samples going along close paths based on the distance.

Correlation between samples. To benefit from correlated samples, we first need to measure the distance between samples. Let $H(\cdot)$ be a mapping function, which can be a clustering approach, a self-learning strategy, or a statistic method, as discussed in Supplementary. Then, the distance between sample X_i and X_j can be calculated by

$$\mathcal{D}_{i,j} = \left\| H(X_i) - H(X_j) \right\|_2, \tag{10}$$

where $\|\cdot\|_2$ is the L^2 norm. In this way, $\mathcal{D}$ represents the distance matrix of samples in the same train batch. Next,

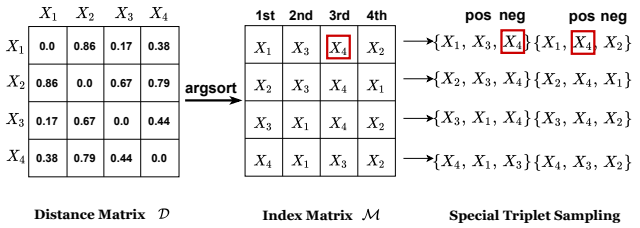


Figure 4. Overview of SRM. We first obtain the distance matrix $\mathcal{D}$ by a mapping function. Next, we sort the distance matrix row-wise and get an index matrix $\mathcal{M}$ in which each item is the index of a sample. Finally, we regularize the path between samples based on $\mathcal{M}$. For example, when we regularize the executing path of X_1, X_4 is a negative sample in triplet $\{X_1, X_3, X_4\}$ and a positive sample in triplet $\{X_1, X_4, X_2\}$.

we sort the distance matrix $\mathcal{D}$ row-wise and get the indices in ascending order,

$$\mathcal{M} = \text{argsort}(\mathcal{D}). \tag{11}$$

Specifically, each item in $\mathcal{M}$ is an index of a sample. The i-th row stands for the ranking of the distance from other samples to X_i in ascending order, as shown in Figure 4.

Regularizing executing paths. Based on the ordered indices, we regularize the executing paths in triplets to enforce a consistent ranking between executing paths and distance of samples. Specifically, for sample X_i, we regularize its executing path P_i with a series of positive and negative triplets.

To make the optimization efficient, we perform a special-designed triplet sampling method that takes every two adjacent items in $\mathcal{M}_{i,:}$, as a positive and a negative sample for X_i. As illustrated in Figure 4, when regularizing the executing path of X_1, we first optimize it in triplet $\{X_1, X_3, X_4\}$ and then in triplet $\{X_1, X_4, X_2\}$. It is worth noting that X_4 is a negative sample in the former triplet, while positive in the latter.

Let $\mathcal{M}_{ij}$ be the index of the j-th closest sample for X_i. Thus, $P_{\mathcal{M}_{ij}}$ is the path of the $\mathcal{M}_{ij}$-th sample. The regularization of the executing paths triplets is defined as

$$\mathcal{L}_{rank} = \sum_{i=1}^{L}\sum_{j=2}^{L-1}[\left\|P_i - P_{\mathcal{M}_{ij}}\right\|_2 - \left\|P_i - P_{\mathcal{M}_{i,j+1}}\right\|_2 + \phi]_+, \tag{12}$$

where P_i is the path of the i-th sample in the batch, L is the batch size, and $[x]_+ = \max(0, x)$ denotes the hinge. It is worth noting that ϕ is the triplet margin, which prevents the optimization from degenerating into only working on the first and last terms.

In conclusion, the goal of $\mathcal{L}_{rank}$ is to regularize the ranking of executing paths consistent with the ranking of distance between samples.

3.4. Optimization

At training time, we optimize our network in two stages. First, we warm up the parameters of the blocks with a strategy called Uniform Sampling Warm-up (USW). Second, we train our RDI-Net by optimizing all parameters including those of blocks and relational routers.

Warm-up strategy. A fundamental principle behind dynamic inference is that any executing path in a dynamic inference network is a subgraph of the whole network topology. Thus, the key to optimizing a dynamic inference network is parameter sharing. Let $\mathcal{F} = \{F_1, ..., F_N\}$ and $\mathcal{R} = \{R_1, ..., R_N\}$, where R_n stands for the relational router for F_n. The optimization of a dynamic inference network is

$$w^*_{\mathcal{F}}, w^*_{\mathcal{R}} = \underset{w_{\mathcal{F}}, w_{\mathcal{R}}}{\text{argmin}}\, \mathcal{L}_{cls}(\mathcal{F}, \mathcal{R}), \tag{13}$$

where $w_{\mathcal{F}}$ and $w_{\mathcal{R}}$ stand for the parameters of $\mathcal{F}$ and $\mathcal{R}$ respectively, and $\mathcal{L}_{cls}$ is the cross-entropy loss. In this way, $\mathcal{F}$ and $\mathcal{R}$ are optimized simultaneously. However, the difficulty of finding an optimal executing path is different from sample to sample. To avoid falling into local optimizations, every potential path should be optimized equally before being assigned to a given sample.

Inspired by SPOS [12], we propose a warm-up strategy called Uniform Sampling Warm-up (USW) to train each executing path of RDI-Net uniformly. We build a standard uniform sampler and sample a sequence of choices for each block and warm up the network as

$$w^*_{\mathcal{F}} = \underset{w_{\mathcal{F}}}{\text{argmin}}\, \mathcal{L}_{cls}(\mathcal{F}, P), \tag{14}$$

where $P \sim \mathcal{U}$ stands for a sampled executing path and $\mathcal{U}$ is N-dimensional uniform distribution.

Objective functions. After warming-up, we optimize all the parameters of RDI-Net.

To customize the computational budget for a dynamic inference network adaptively, we constrain how often each block is used. For each block, we introduce an extra execution rate loss term that regularizes each block to be executed at a predefined rate t as follows

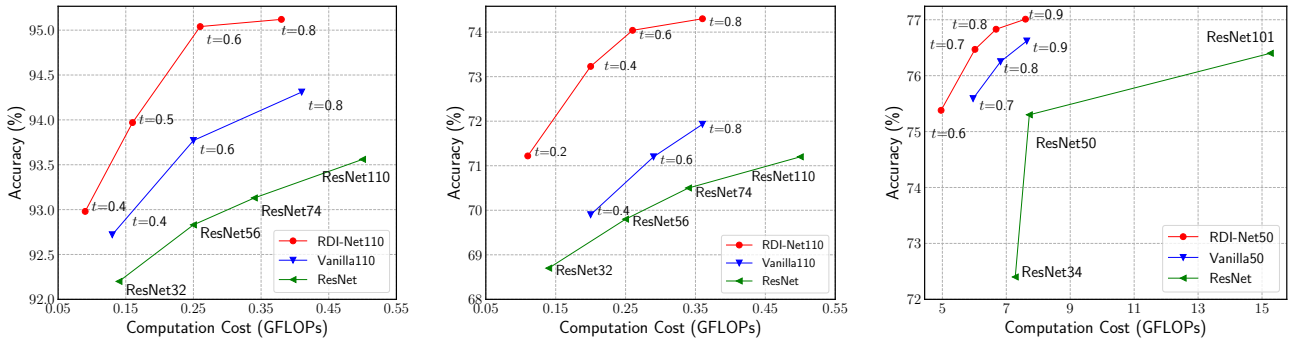
$$\mathcal{L}_{cost} = \sum_{n=1}^{N}(v_n - t)^2, \tag{15}$$

where v_n is the relaxed executing decision of the n-th block as discussed in Equation (9).

Putting all the losses together, the overall objective is

$$\mathcal{L}_{total} = \mathcal{L}_{cls} + \mathcal{L}_{cost} + \alpha \cdot \mathcal{L}_{rank}, \tag{16}$$

where $\mathcal{L}_{cls}$ is the cross entropy loss, $\mathcal{L}_{rank}$ is the loss in Equation (12), and $\mathcal{L}_{cost}$ is the loss in Equation (15).



(a) Accuracy against cost on CIFAR-10 (b) Accuracy against cost on CIFAR-100 (c) Accuracy against cost on ImageNet

Figure 5. Accuracy against computation cost under different executing rates. (a) On CIFAR-10, our RDI-Net110 reduces 73% of the cost with an accuracy gain of 0.13% accuracy compared to ResNet-110. (b) On CIFAR-100, our RDI-Net110 reduces 65% of the cost with an accuracy gain of 0.42% compared to ResNet-110. (c) On ImageNet, our RDI-Net50 reduces 35% of the computational cost compared to ResNet-50. It is worth noting that the vanilla method in each figure is the dynamic inference method with short range routers.

4. Experiments

In this section, we first introduce our experimental details and compare with other state-of-the-art methods on three classification benchmarks in terms of GFLOPs and accuracy. Furthermore, we conduct ablation studies to validate the proposals in our method. Lastly, we present qualitative evaluation of our method. Our code is available at https://github.com/huanyuhello/RDI-Net.

4.1. Experimental Settings

Datasets and models. We evaluate RDI-Net on three widely used classification benchmarks: CIFAR-10 [27], CIFAR-100 [27], and ImageNet (ISLVRC2012) [6]. CIFAR-10/100 consist of 50,000 training images and 10,000 testing images with 10/100 classes at a resolution of 32×32. ImageNet consists of 1,281,167 training images and 50,000 validation images with 1000 classes at a resolution of 224×224. Our method is based on ResNets [14], commonly used backbones for dynamic inference. We adopt ResNet-32/110 for CIFAR-10/100 and ResNet-50/101 for ImageNet.

Implementation details. At training time, we warm up the model with USW for 30 and 10 epochs on CIFAR-10/100 and ImageNet, respectively. After warming up, we train for 320 epochs with a batch size of 256 on CIFAR-10/100. As for ImageNet, we train for 90 epochs with the same batch size as CIFAR-10/100. In terms of our modules, the router feature dimension d and the effect γ of relational routers are set to 256 and 0.5, respectively. In SRM, the margin ϕ for $\mathcal{L}_{rank}$ is 0.5 and the weight α is set to 0.2. We use histogram statistics as the mapping function $H(\cdot)$. Moreover, we ignore the cases that the distance of positive and negative samples are about the same to the anchors in Equation (12). Finally, through adjusting t, we obtain a series of models under different computational budgets.

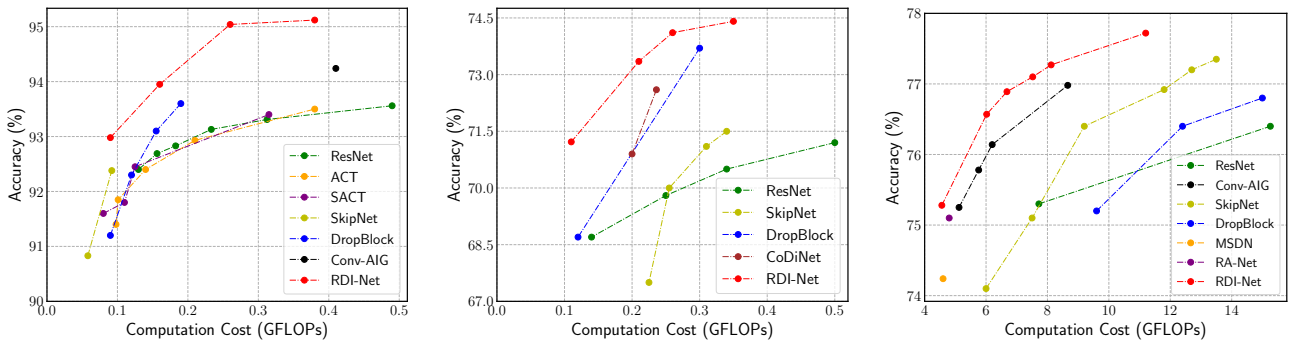
4.2. Performance Comparison

We illustrate the advantages of RDI-Net by comparing it with other widely used methods on different datasets. First, we compare RDI-Net with original ResNets and baseline methods. Then, we compare RDI-Net with state-of-the-art dynamic inference methods.

Comparisons with baselines. We make comparisons between RDI-Net and the baseline methods in terms of accuracy, parameters, and GFLOPs. To better illustrate the effectiveness of our method, we provide the accuracy against cost by adjusting the rate t on CIFAR10, CIFAR-100, and ImageNet as shown in Figure 5. On CIFAR-10, our method only needs 27% computational cost of the ResNet-110, and achieves better results as shown in Figure 5(a). Similarly, on CIFAR-100, our method achieves significant improvement, which reduces 65% computational cost with a higher accuracy on ResNet-110, as shown in Figure 5(b).

In terms of ImageNet, RDI-Net achieves impressive results as shown in Figure 5(c). Particularly, RDI-Net accomplishes even 0.3% higher accuracy than SENet. The RDI-Net achieves about 1.17% improvement on ResNet-50 with 1.54 GFLOPs reduction. Besides, with a higher accuracy, RDI-Net50 reduces 35% of the computational cost than ResNet-50. Moreover, comparing to the vanilla dynamic inference method, our method consistently surpass the Vanilla50 under different computation budgets. Concrete results are provided in supplemental material.

Comparisons with state-of-the-arts. Next, we compare RDI-Net with other state-of-the-art dynamic inference methods on three benchmarks. On CIFAR-10 and CIFAR-100, we compare RDI-Net with the following methods: SkipNet [51], BlockDrop [48], Conv-AIG [40], ACT [10], SACT [8], and CoDiNet [44]. Conv-AIG and SkipNet are prevalent methods that apply Gumbel-SoftMax and LSTM to dynamic inference. As shown in Figure 6(a) and Fig-



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

Figure 6. Comparison with state-of-the-art methods on CIFAR-10, CIFAR-100, and ImageNet. The results in terms of GFLOPs are the average computational costs. All methods are based on ResNets. Detailed results are provided in the supplemental material.

Table 2. Effectiveness of the proposed modules on CIFAR-10/100 with ResNet-110. USW: Uniform Sampling Warm-up; R-R: Relational Router; SRM: Sample Relation Modeling Module. The first row refers to using routers without router feature fusion.

Methods			CIFAR-10		CIFAR-100	
USW	R-R	SRM	GFLOPs	Acc. (%)	GFLOPs	Acc. (%)
—	—	—	0.30	93.21	0.29	70.32
✓	—	—	0.28	93.52	0.27	70.81
✓	✓	—	0.31	94.57	0.30	73.71
✓	✓	✓	0.25	95.06	0.26	74.11

ure 6(b), RDI-Net consistently surpasses other methods at different computational cost. Specifically, it achieves 95.06% accuracy with 0.25 GFLOPs on CIFAR-10 and 74.21% with 0.35 GFLOPs on CIFAR-100, respectively.

On ImageNet, we compare RDI-Net with several SOTA methods and yield similar results to those on CIFAR as shown in Figure 6(c). We observe that RDI-Net outperforms other methods under similar settings. Compared to prevalent methods, our method surpasses SkipNet [51], and BlockDrop [48] by a large margin. Our models win about 0.47% and 0.41% accuracies than Conv-AIG [40] with a comparable cost on ResNet-50 and ResNet-101. The highest accuracy our method obtained is 77.01% with 6.62 GFLOPs and 77.68% with 11.4 GFLOPs, based on ResNet-50 and ResNet-101, respectively. To summarize, our models achieve more accurate classification results than these SOTA methods with similar computational cost.

4.3. Ablation Study

In this part, we first evaluate the effectiveness of each module, *i.e.*, relational routers, Sample Relation Module, and Uniform Sampling Warm-Up. Next, we study different factors *i.e.*, decay rates γ and the number of nodes r in the graph of each relational router. Finally, we conduct studies on the weight α for Sample Relation Module.

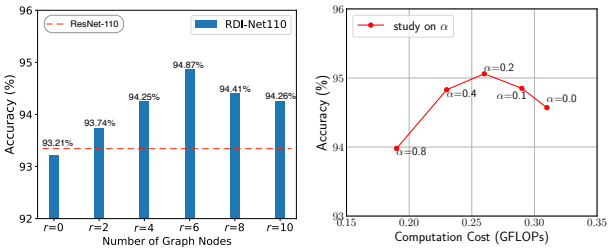
Effectiveness of the proposals. As shown in Table 2, we present the improvement on CIFAR-10 and CIFAR-100 brought by our proposed USW, relational routers, and SRM to evaluate the effectiveness of each module. The standard baseline is a vanilla dynamic inference network, which is trained without router feature propagation. On CIFAR-10, the vanilla achieves an accuracy of 93.21% and a computational cost of 0.30 GFLOPs. By adding the warm-up method, USW, the accuracy is increased by 0.36%, while computational cost decreases to 0.28 GFLOPs. Then, we replace the basic routers with our proposed relational routers. Accuracy is again improved by 1.05%. Lastly, we add the SRM defined in Equation (12) between samples, the accuracy increases to 95.06% with 0.25 GFLOPs.

Table 3. Comparison of different feature propagation methods in relational routers on CIFAR-10 with ResNet-110. Auto refers to learnable weights. γ is the decay rate defined in Table 1.

Methods	$\gamma = 1.0$	$\gamma = 0.5$	$\gamma = 0.25$	$\gamma = 0.0$	Auto
GFLOPs	0.30	0.31	0.27	0.30	0.28
Acc. (%)	93.81	94.87	94.12	93.21	94.19

Studies on the relational routers. In this part, we study different decay rates γ in relational routers. *i.e.*, the weight to propagate the features from earlier blocks to the current block. Specifically, experiments are conducted without SRM and USW. As shown in Table 3, we show decay rates from $\gamma = 0$ to $\gamma = 1$ and learnable weights method. When $\gamma = 1.0$, the method degenerates into a sum of all decision features, result in 93.81% accuracy with 0.30 GFLOPs. In terms of learnable weights, we use the cosine similarity between features as edges in the graph, which achieves 94.19% accuracy with 0.28 GFLOPs. As a result, when the decay rate is set to 0.5, our method achieves the best result.

Next, we analyze the number of nodes taken into account for decision feature propagation. A relational router aggregates the features from the previous r blocks and makes the executing decision of the current block in a long-range manner. The experiments are conducted without SRM and USW. The cost of all experiments are about 0.30 GFLOPs. With the increasing of r, the accuracy increase at first and



(a) A study on r (b) A study on α

Figure 7. Studies on r and α on CIFAR-10 with ResNet-110. r is the number of graph nodes. α is the loss weight for $\mathcal{L}_{rank}$.

then decrease when r is larger than six as shown in Figure 7(a). Since distant routers are at different semantic levels, introducing these router features would increase the difficulty of modeling, result in the performance dropping.

Studies on the Sample Relation Module. Finally, we perform studies on the weight α for $\mathcal{L}_{rank}$. As shown in Figure 7(b), with the increasing of α, there is an evidently decreasing tendency in computational cost. It shows that the SRM, introducing the relation between samples to their paths, benefits the cost reduction. Moreover, when α = 0.2, the RDI-Net achieves the highest performance on accuracy, which is 95.06% with 0.26 GFLOPs.

4.4. Qualitative Analysis

In this part, we conduct experiments to analyze our proposals qualitatively. First, we visualize the correlated samples that go along the same path. Next, we do statistics on the executing rates of each block in our RDI-Net and the vanilla dynamic inference methods.

Visualization of samples in the inference path. We visualize some images with the same executing paths of the ImageNet validate set and images in the same row execute the same path, as shown in Figure 8. In the first row, images execute the path A which contains 15 blocks out of 16 from ResNet-50. Images with similar executing paths are usually of the same colors and similar backgrounds in RDI-Net since foreground objects are too small to capture by statistical approaches. Moreover, path B consists of less executing blocks (10 out of 16), probably due to the less complex background of the images. Finally, since we regularize executing paths w.r.t. correlated attributes of samples instead of semantic classes, images in the same executing path do not necessarily have the same class.

Executing rates for each block. We do statistics on the blocks executing rates and compare it with the vanilla dynamic inference network, as shown in Figure 9. Based on ResNet-50, skipping mainly takes place in the intermediate blocks, namely from the third block to the thirteenth block. Moreover, in RDI-Net, the executing rate of blocks after downsampling (designed in ResNet-50), *i.e.*, the fourth and the eighth block, are executed more frequently than the

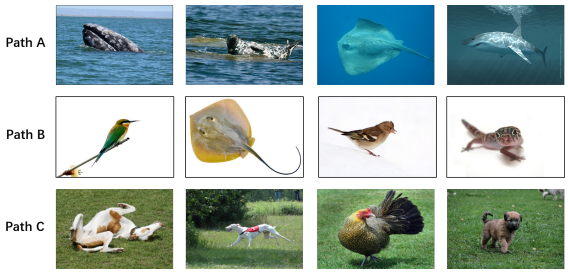


Figure 8. Visualization of samples executing the same path on the ImageNet validation set. Images along path A share similar blue hues, images along path B share a white background, and images along path C have green backgrounds. Path A, path B, and path C have 15, 10, and 14 blocks, respectively. Best viewed in color.

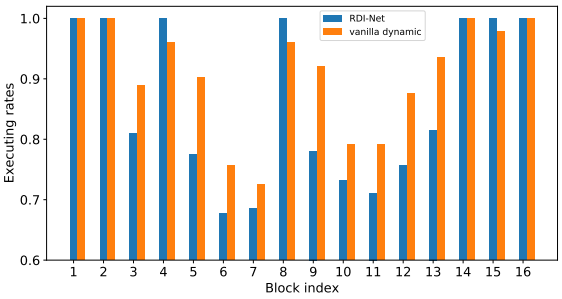


Figure 9. Comparison of executing behaviors based on ResNet-50 on the ImageNet validation set. Blue bars stand for the executing rates for each block of RDI-Net. Orange bars stand for the executing rates of vanilla dynamic inference.

vanilla dynamic inference. The executing rates of the other intermediate blocks are less than the vanilla dynamic inference network, improving the efficiency considerably.

5. Conclusion

In this paper, we have proposed a Relational Dynamic Inference Network which explicitly models the contextual relation among routers by relational routers in a long-range manner. Concretely, the current relational router makes use of graph convolution to sequentially aggregate the historical router features to obtain a discriminative feature representation for the executing decision. To model the correlation between samples and their executing paths, we present a Sample Relation Module that regularizes correlated samples to go along correlated executing paths. As a result, extensive experiments have shown that our method achieves state-of-the-art performance and computational cost reduction.

Acknowledgement

This work is 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] Amjad Almahairi, Nicolas Ballas, Tim Cooijmans, Yin Zheng, Hugo Larochelle, and Aaron Courville. Dynamic capacity networks. In *Int. Conf. Mach. Learn.*, 2016. 1, 2

[2] Han Cai, Ligeng Zhu, and Song Han. Proxylessnas: Direct neural architecture search on target task and hardware. In *Int. Conf. Learn. Represent.*, 2019. 3

[3] Guobin Chen, Wongun Choi, Xiang Yu, Tony Han, and Manmohan Chandraker. Deep residual learning for image recognition. In *Adv. Neural Inform. Process. Syst.*, 2017. 2

[4] Jin Chen, Xijun Wang, Zichao Guo, Xiangyu Zhang, and Jian Sun. Dynamic region-aware convolution. *arXiv:2003.12243*, 2020. 2

[5] Zhenghua Chen, Le Zhang, Zhiguang Cao, and Jing Guo. Distilling the knowledge from handcrafted features for human activity recognition. In *IEEE Trans. Indust. Inform.*, 2018. 2

[6] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *IEEE Conf. Comput. Vis. Pattern Recog.*, 2009. 6

[7] Pedro F Felzenszwalb, Ross B Girshick, and David McAllester. Cascade object detection with deformable part models. In *IEEE Conf. Comput. Vis. Pattern Recog.*, 2010. 2

[8] Michael Figurnov, Maxwell D Collins, Yukun Zhu, Li Zhang, Jonathan Huang, Dmitry Vetrov, and Ruslan Salakhutdinov. Spatially adaptive computation time for residual networks. In *IEEE Conf. Comput. Vis. Pattern Recog.*, 2017. 2, 6

[9] Xitong Gao, Yiren Zhao, Lukasz Dudziak, Robert Mullins, and Cheng-zhong Xu. Dynamic channel pruning: Feature boosting and suppression. In *Int. Conf. Learn. Represent.*, 2018. 2

[10] Alex Graves. Adaptive computation time for recurrent neural networks. *arXiv:1603.08983*, 2016. 2, 6

[11] Qiushan Guo, Zhipeng Yu, Yichao Wu, Ding Liang, Haoyu Qin, and Junjie Yan. Dynamic recursive neural network. In *IEEE Conf. Comput. Vis. Pattern Recog.*, 2019. 2

[12] Zichao Guo, Xiangyu Zhang, Haoyuan Mu, Wen Heng, Zechun Liu, Yichen Wei, and Jian Sun. Single path one-shot neural architecture search with uniform sampling. In *Eur. Conf. Comput. Vis.*, 2020. 5

[13] Song Han, Huizi Mao, and William J Dally. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. In *Int. Conf. Learn. Represent.*, 2016. 2

[14] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *IEEE Conf. Comput. Vis. Pattern Recog.*, 2016. 6

[15] Yihui He, Xiangyu Zhang, and Jian Sun. Channel pruning for accelerating very deep neural networks. In *Int. Conf. Comput. Vis.*, 2017. 3

[16] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. In *Adv. Neural Inform. Process. Syst.*, 2015. 2

[17] Hanzhang Hu, Alexander Grubb, J Andrew Bagnell, and Martial Hebert. Efficient feature group sequencing for anytime linear prediction. In *Conf. Uncert. Artif. Intell.*, 2014. 2

[18] Gao Huang, Danlu Chen, Tianhong Li, Felix Wu, Laurens van der Maaten, and Kilian Q Weinberger. Multi-scale dense networks for resource efficient image classification. In *Int. Conf. Learn. Represent.*, 2018. 2

[19] Gao Huang, Shichen Liu, Laurens Van der Maaten, and Kilian Q Weinberger. Condensenet: An efficient densenet using learned group convolutions. In *IEEE Conf. Comput. Vis. Pattern Recog.*, 2018. 3

[20] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *IEEE Conf. Comput. Vis. Pattern Recog.*, 2017. 2

[21] Noureldien Hussein, Mihir Jain, and Babak Ehteshami Bejnordi. Timegate: Conditional gating of segments in long-range activities. *arXiv:2004.01808*, 2020. 1, 2

[22] Yani Ioannou, Duncan Robertson, Jamie Shotton, Roberto Cipolla, and Antonio Criminisi. Training cnns with low-rank filters for efficient image classification. In *Int. Conf. Learn. Represent.*, 2016. 2

[23] Max Jaderberg, Andrea Vedaldi, and Andrew Zisserman. Speeding up convolutional neural networks with low rank expansions. In *Brit. Mach. Vis. Conf.*, 2014. 2

[24] Eric Jang, Shixiang Gu, and Ben Poole. Categorical reparameterization with gumbel-softmax. In *Int. Conf. Learn. Represent.*, 2017. 4

[25] Artur Jordao, Maiko Lie, and William Robson Schwartz. Discriminative layer pruning for convolutional neural networks. *IEEE J. of Selected Topics Signal Process.*, 2020. 2

[26] Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *Int. Conf. Learn. Represent.*, 2017. 2, 4

[27] Alex Krizhevsky et al. Learning multiple layers of features from tiny images. In *Citeseer*, 2009. 6

[28] Hao Li, Asim Kadav, Igor Durdanovic, Hanan Samet, and Hans Peter Graf. Pruning filters for efficient convnets. In *Int. Conf. Learn. Represent.*, 2017. 3

[29] Hao Li, Hong Zhang, Xiaojuan Qi, Ruigang Yang, and Gao Huang. Improved techniques for training adaptive deep networks. In *Int. Conf. Comput. Vis.*, 2019. 2

[30] Jian-Hao Luo, Jianxin Wu, and Weiyao Lin. Thinet: A filter level pruning method for deep neural network compression. In *IEEE Conf. Comput. Vis. Pattern Recog.*, 2017. 3

[31] Lane McIntosh, Niru Maheswaranathan, David Sussillo, and Jonathon Shlens. Convolutional neural networks with low-rank regularization. In *Int. Conf. Learn. Represent.*, 2016. 2

[32] Lane McIntosh, Niru Maheswaranathan, David Sussillo, and Jonathon Shlens. Recurrent segmentation for variable computational budgets. In *IEEE Conf. Comput. Vis. Pattern Recog.*, 2018. 2

[33] Yue Meng, Chung-Ching Lin, Rameswar Panda, Prasanna Sattigeri, Leonid Karlinsky, Aude Oliva, Kate Saenko, and

Rogerio Feris. Ar-net: Adaptive frame resolution for efficient action recognition. In *Eur. Conf. Comput. Vis.*, 2020. 1, 2

[34] Antonio Polino, Razvan Pascanu, and Dan Alistarh. Model compression via distillation and quantization. In *Int. Conf. Learn. Represent.*, 2018. 2

[35] Lev Reyzin. Boosting on a budget: Sampling for feature-efficient prediction. In *Int. Conf. Mach. Learn.*, 2011. 2

[36] Zhuo Su, Linpu Fang, Wenxiong Kang, Dewen Hu, Matti Pietikäinen, and Li Liu. Dynamic group convolution for accelerating convolutional neural networks. 2020. 2

[37] Fei Sun, Minghai Qin, Tianyun Zhang, Liu Liu, Yen-Kuang Chen, and Yuan Xie. Computation on sparse neural networks: an inspiration for future hardware. *arXiv:2004.11946*, 2020. 2

[38] Mingxing Tan, Bo Chen, Ruoming Pang, Vijay Vasudevan, Mark Sandler, Andrew Howard, and Quoc V Le. Mnasnet: Platform-aware neural architecture search for mobile. In *IEEE Conf. Comput. Vis. Pattern Recog.*, 2019. 3

[39] Mingxing Tan and Quoc V Le. Efficientnet: Rethinking model scaling for convolutional neural networks. In *Int. Conf. Mach. Learn.*, 2019. 3

[40] Andreas Veit and Serge Belongie. Convolutional networks with adaptive inference graphs. In *Eur. Conf. Comput. Vis.*, 2018. 1, 2, 6, 7

[41] Andreas Veit, Michael Wilber, and Serge Belongie. Residual networks behave like ensembles of relatively shallow networks. In *Adv. Neural Inform. Process. Syst.*, 2016. 1, 2

[42] Thomas Verelst and Tinne Tuytelaars. Dynamic convolutions: Exploiting spatial sparsity for faster inference. In *IEEE Conf. Comput. Vis. Pattern Recog.*, 2020. 2

[43] Paul Viola and Michael Jones. Robust real-time face detection. In *Int. J. Comput. Vis.*, 2004. 2

[44] Huanyu Wang, Zequn Qin, and Xi Li. Codinet: Path distribution modeling with consistency and diversity for dynamic routing. *arXiv:2005.14439*, 2020. 1, 2, 6

[45] Wei Wen, Chunpeng Wu, Yandan Wang, Yiran Chen, and Hai Li. Learning structured sparsity in deep neural networks. In *Adv. Neural Inform. Process. Syst.*, 2016. 3

[46] Bichen Wu, Xiaoliang Dai, Peizhao Zhang, Yanghan Wang, Fei Sun, Yiming Wu, Yuandong Tian, Peter Vajda, Yangqing Jia, and Kurt Keutzer. Fbnet: Hardware-aware efficient convnet design via differentiable neural architecture search. In *IEEE Conf. Comput. Vis. Pattern Recog.*, 2019. 3

[47] Jiaxiang Wu, Cong Leng, Yuhang Wang, Qinghao Hu, and Jian Cheng. Quantized convolutional neural networks for mobile devices. In *IEEE Conf. Comput. Vis. Pattern Recog.*, 2016. 2

[48] Zuxuan Wu, Tushar Nagarajan, Abhishek Kumar, Steven Rennie, Larry S Davis, Kristen Grauman, and Rogerio Feris. Blockdrop: Dynamic inference paths in residual networks. In *IEEE Conf. Comput. Vis. Pattern Recog.*, 2018. 1, 2, 6, 7

[49] Wenhan Xia, Hongxu Yin, Xiaoliang Dai, and Niraj K Jha. Fully dynamic inference with deep neural networks. *IEEE Trans. on Emerg. Topics in Comput.*, 2020. 2

[50] Zhenda Xie, Zheng Zhang, Xizhou Zhu, Gao Huang, and Stephen Lin. Spatially adaptive inference with stochastic feature sampling and interpolation. *arXiv:2003.08866*, 2020. 2

[51] Wang Xin, Yu Fisher, Dou Zi-Yi, Darrell Trevor, and E. Gonzalez Joseph. Skipnet: Learning dynamic routing in convolutional networks. In *Eur. Conf. Comput. Vis.*, 2018. 1, 2, 6, 7

[52] Le Yang, Yizeng Han, Xi Chen, Shiji Song, Jifeng Dai, and Gao Huang. Resolution adaptive networks for efficient inference. In *IEEE Conf. Comput. Vis. Pattern Recog.*, 2020. 2

[53] Jiahui Yu and Thomas S Huang. Universally slimmable networks and improved training techniques. In *Int. Conf. Comput. Vis.*, 2019. 2

[54] Ruichi Yu, Ang Li, Chun-Fu Chen, Jui-Hsin Lai, Vlad I Morariu, Xintong Han, Mingfei Gao, Ching-Yung Lin, and Larry S Davis. Nisp: Pruning networks using neuron importance score propagation. In *IEEE Conf. Comput. Vis. Pattern Recog.*, 2018. 2

[55] Pengyi Zhang, Yunxin Zhong, and Xiaoqiong Li. Slimyolov3: Narrower, faster and better for real-time uav applications. In *IEEE Conf. Comput. Vis. Pattern Recog.*, 2019. 1, 2