Meta-GF: Training Dynamic-Depth Neural Networks Harmoniously

Yi Sun[1], Jian Li[1⋆], and Xin Xu[1]

The College of Intelligence Science and Technology, National University of Defense Technology, Changsha 410000, China
sunyi13@nudt.edu.cn; lijian@nudt.edu.cn; xinxu@nudt.edu.cn;

Abstract. Most state-of-the-art deep neural networks use static inference graphs, which makes it impossible for such networks to dynamically adjust the depth or width of the network according to the complexity of the input data. Different from these static models, depth-adaptive neural networks, e.g. the multi-exit networks, aim at improving the computation efficiency by conducting adaptive inference conditioned on the input. To achieve adaptive inference, multiple output exits are attached at different depths of the multi-exit networks. Unfortunately, these exits usually interfere with each other in the training stage. The interference would reduce performance of the models and cause negative influences on the convergence speed. To address this problem, we investigate the gradient conflict of these multi-exit networks, and propose a novel meta-learning based training paradigm namely **Meta-GF**(meta gradient fusion) to harmoniously train these exits. Different from existing approaches, Meta-GF takes account of the importances of the shared parameters to each exit, and fuses the gradients of each exit by the meta-learned weights. Experimental results on CIFAR and ImageNet verify the effectiveness of the proposed method. Furthermore, the proposed Meta-GF requires no modification on the network structures and can be directly combined with previous training techniques. The code is available at https://github.com/SYVAE/MetaGF.

Keywords: Dynamic networks; gradient conflict; meta learning.

1 Introduction

Deep neural networks have achieved tremendous progress in many applications such as recognition[8, 18] and detection[3, 13]. Due to the implicit regularization caused by over-parameterization[16, 9], the very deep networks with large number of parameters empirically have stronger representation capacity and more robust generalization ability compared with the small ones. Yet they are always much more computationally expensive, especially when deployed on resource-constrained platforms. Furthermore, due to the varying complexity of input

⋆ Yi Sun and Jian Li are the co-first authors with equal contributions. Xin Xu is the corresponding author.

data, hence not every input requires the same network depth. To realize better accuracy with efficient inference speed, network pruning approaches[37] and various lightweight networks[41, 45] are proposed for obtaining lightweight models. However, the capacity of the resulting small networks is limited, and they cannot dynamically extend their depth or width for dealing with more complicated input. Both the large networks and the small ones perform static inference graphs, which limits the efficiency of the large models and the representation ability of the small models respectively. To sum up, the large networks cannot dynamically reduce their computation complexity to efficiently deal with easy inputs, and the capacity of the small ones are limited which cannot be dynamically extended to handle challenging inputs.

Recently, numerous researches[20, 47, 26, 39, 25, 49, 48] propose solutions to achieving robust and efficient prediction via implementing depth-adaptive inference, such as the MSDnet[20] and SDN[26]. The inference depth of the above mentioned adaptive networks are conditioned on the input. As the most popular adaptive architecture, multi-exit networks[20, 39, 25] realize depth-adaptive inference by early exiting, i.e. allowing the result of "easy" inputs to be output from the shallow exits without executing deeper layers[17]. Toward this end, multiple intermediate output exits are attached to the networks at different depths. In the inference stage, the multi-exit models dynamically decide to stop inference at which output exit based on the predefined exit-policy such as the confidence[20] or learned policy[4]. Depth-adaptive inference is a valuable mechanism for networks to save unnecessary computation costs while keeping prediction performance.

Existing works mainly focus on designing more excellent adaptive structures[26, 39, 23, 54] or better inference algorithms[4], but the interference between different intermediate exits have attracted less attention. To be specific, large numbers of model parameters are shared by these exits, and in the training stage, these shared parameters always receive conflicted gradients from different exits. As defined in [55], "gradients conflict" in this work represents that two gradients have a negative cosine similarity value i.e. the conflicted update directions. The interference between different exits degrades the overall performance and the convergence speed of the multi-exit networks.

To alleviate such a interference, specially designed networks such as the MSDnet[20] were proposed. In [20], they added dense connections between the deeper exits and the early modules of the networks to reduces the negative impact of the early exits on the deeper exits. Knowledge-distillation based approaches[49, 40] are also proposed to align the learning targets of shallow exits with the deeper exits. Another kind of solution is performed by gradient adjustment. The Gradient Equilibrium(GE) proposed in [29] reduces the gradient variance of the deeper exits, which is useful for reducing the negative impact of the deep exits on shallow exits. In the works of the [55] and [48], when two gradients conflict with each other, they project each gradient onto the normal plane of the other for suppressing the interfering components of the gradients. This kind of gradient

surgery method is termed Pcgrad, and it was verified in [55, 48] that Pcgrad can reduce the conflict between different tasks.

Despite the effectiveness of existing approaches, there's room for improvement. Firstly, approaches based on knowledge-distillation and network architecture designs conduct less necessary analysis about the interference at gradient level, and it's considered that gradient directly participates in the model optimization. Secondly, GE controls the gradient variance but doesn't take account of adjusting gradient direction. However, gradient direction decides the update trend of the networks, and the gradient direction conflict is one of the essence of the interference. Thirdly, we find that not all shared parameters are equally important to each exit. The over-parameterization of networks have adequate capacity to allow different exits to own their preferred parameters. Unconstrained gradient re-projection policy might hinder the convergence of the networks. To be specific, when gradients of two exits conflict with each other, the re-projection policy is supposed to take account of the importances of the shared parameters for different exits, instead of simply suppressing the interfering components in the two conflicted gradients.

To tackle the above mentioned issues, we propose a novel gradient fusion method named Meta-GF for training the multi-exit networks. In contrast to the previous approaches, the proposed Meta-GF takes a meta-learned weighted fusion policy to combine the gradients of each exit, while taking account of the different importances of the shared parameters for different exits. Due to the over-parameterization, there are adequate model capacity, which makes the Meta-GF could implicitly disentangle the shared parameters of different tasks as more as possible, e.g. finding exit-specific parameters for each exit by the learned fusion weights. By fusing the gradient of different exits with the meta-weight fusion policy, the proposed approach achieves more harmonious training of the multi-exit networks. Extensive experiments have been conducted on CIFAR and ImageNet datasets. The experimental results demonstrate the effectiveness of the proposed approach. We investigate the gradient conflict problem and introduce Meta-GF in Sec.3. The experimental results are introduced in Sec.4.

2 Related works

2.1 Dynamic deep neural networks

Dynamic deep neural network is a promising research field, where the networks take conditional computation by using adaptive parameters[36, 31, 53, 5, 56], networks width or networks depth[26, 39, 25, 23]. Specially, the depth-adaptive deep neural networks aim at achieving trade-off between the robustness and efficiency by dynamically adjusting the network inference depth. To be specific, the networks conditionally adjust their inference depth according to the complexity of inputs. There are mainly two kinds of depth-adaptive neural network structures: the multi-exit networks and the skip-style networks.

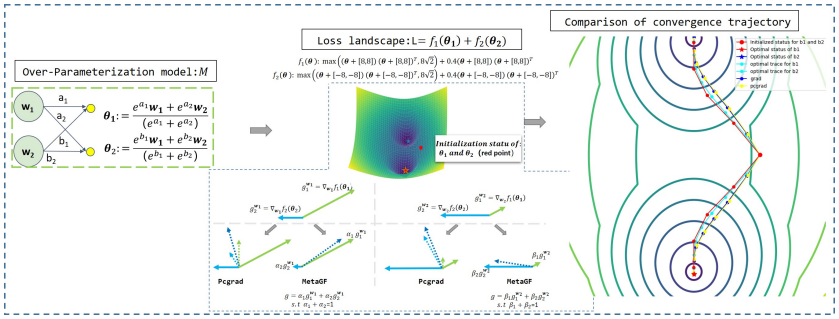


Fig. 1. Meta-Gradient Fusion: given an over-parameterization networks, of which two task exit:$\{\boldsymbol{\theta_1}, \boldsymbol{\theta_2}\}$ are linearly combined by six learnable parameters: $\{\mathbf{w}_1, \mathbf{w}_2, (a_1, a_2), (b_1, b_2)\}$. The total loss surface of both two tasks is $f_1(\boldsymbol{\theta_1}) + f_2(\boldsymbol{\theta_2})$, which is shown in the middle subfigure. When jointly training the two tasks, the gradients of two tasks conflict with each other, i.e $g_1^{\mathbf{w}_1} \cdot g_2^{\mathbf{w}_1} < 0$ and $g_1^{\mathbf{w}_2} \cdot g_2^{\mathbf{w}_2} < 0$. The proposed Meta-GF takes a meta-weight fusion policy to combine the expected gradients of two tasks, which takes account of the different importances of the shared parameters for different exits. Due to the over-parameterization, there are adequate model capacity, which makes the Meta-GF could disentangle the shared parameters of different tasks as more as possible, and achieve harmonious joint training process. In the right subfigure, the model convergence trajectory when using the Meta-GF are more close to the trajectories when training the two tasks independently. It indicates the Meta-GF achieves more harmoniously training, and verifies the effectiveness of Meta-GF.

multi-exit networks multi-exit structure is commonly adopted to construct inference depth-adaptive networks. The most intuitive approach to designing multi-exit networks are attaching different output exits at different depth of the model. [26, 39, 25, 23]. By designing early-exiting policy such as the confidence-based criterion[26, 52, 42, 20, 50] or the learned policy networks[25, 19, 19] for evaluating the complexity of inputs, the multi-exit networks can adaptively select the output exits and thus adjust the inference depth according to the input complexity. In [57], they force the networks to exit when the predicted confidence score doesn't change for a predefined depth. Compared with the confidence-based criterion, the learned policy networks trained by reinforcement learning[25, 19, 2, 7] or variational Bayes optimization[4] have better expandability that can be transferred to other tasks such as the object tracking[19]. Besides, Liu et al.[35] propose an approach to estimate the complexity of the input according to the reconstruction loss of the data. To make the shallow module of the multi-exit networks can obtain multi-scale receptive fields, the MSDnet proposed in [20] adopts parallel multi-resolution calculation. Recently, the transformer-style networks have shown powerful and robust representation ability, and it's also can be modified into multi-exit style[1, 11] to improve the efficiency of the transformer-style networks.

However, different exits might interfere with each other. Specifically speaking, the model parameters shared by these intermediate exits always receive conflicted

update gradients from different exits in the training stage. To address this interference, some recent works[40, 49, 32] proposed training algorithms based on knowledge distillation to align the learning objective of each exit. Differently, the gradient-surgery approaches such as the Pcgrad [55, 48] reduce the conflict between different exits by performing gradient re-projection policy. Gradient equilibrium[29] or weighted-loss [10] is applied to adjust the gradient norm of each exit for optimizing the training progress.

skip-style networks In addition to the multi-exit networks, the skip-style networks dynamically adjust the inference depth of the networks by adaptively skip the non-linear neural modules. Early works like the stochastic depth networks[21] randomly skip the residual modules in ResNet[18] in the training stage. In [12], Angela et al. applied a learned drop-out rate to randomly skip transformer layers in the testing stage. To achieve more controllable depth-adaptive inference in the testing stage, the networks proposed in [24, 47, 51, 46] set series of gating modules in the position of skip-connections. The Skipnet[47] and Blockdrop[51] adopt reinforcement learning to make the gating modules learn discrete decision strategies, i.e "skip" or "not skip". Instead of skipping the whole layer, the fractional skipping policies proposed in [30, 44, 28] dynamically select part of the layer channels to execute, which can be also regarded as dynamic channel pruning approaches[14]. In this work, we mainly focus on the multi-exit depth adaptive networks.

2.2 Gradient de-conflict in multi-task learning

The gradient conflict problems also exist in the regime of multi-task learning, because different task heads shared the models. If the gradients of different task objectives are not well aligned[33], the average gradients would not provide a well convergence direction for the multi-task networks. To tackle this issue, different gradient adjustment approaches were proposed[6, 34, 15, 43, 33, 22]. Magnitude-rescaling algorithm such as the GradNorm[6] or IMTL[34], is one kind of the gradient adjustment strategies, which aims at balancing the gradient magnitudes of different tasks. In [15], they propose an adaptive loss-weighting policy to prioritize more difficult tasks. The MGDA-UB[43] and CAGrad[33] optimize the overall objective of the multi-task models to find a Pareto optimal solution. In order to reduce the level of gradient conflict, the PCgrad[55] projects each conflict gradient onto the normal plane of the other for suppressing the interfering components. These methods mentioned above all concentrate on solving the task conflicts on the shared parts of the multi-task models. In contrast to these gradient-adjustment approaches, the proposed Meta-GF takes a meta-learned weighted fusion policy to combine the gradients of each exit, while taking account of the different importances of the shared parameters for different exits.

3 Method

In this section, we firstly investigate the gradient conflict by using a toy experiment as shown in Fig.1.Then we introduce details of the proposed Meta-GF.

3.1 Gradient Conflict

Without loss of generality, we make analysis of the gradient conflict problems on the two-exit over-parameterization networks :$M = \{\boldsymbol{\theta_1}, \boldsymbol{\theta_2}\}$. The $\{\boldsymbol{\theta_1}, \boldsymbol{\theta_2}\}$ are the parameters of the two exits respectively. Defining the shared parameters are $\{\mathbf{w}_1, \mathbf{w}_2\}$, and the objective function of the two exits are $\{f_1(\boldsymbol{\theta}), f_2(\boldsymbol{\theta})\}$ as shown in Fig.1. We initialize the model M with $\{\mathbf{w}_1, \mathbf{w}_2, a_1, a_2, b_1, b_2\} = \{(5, 1.5), (5, 0.5), 1, 1, 1, 1\}$. **We take the gradient conflict on $\mathbf{w}_1$ as example** when $\nabla_{\mathbf{w}_1} f_1 \cdot \nabla_{\mathbf{w}_1} f_2 < 0$. The loss degradation of f_1 can be approximately by the First-order Taylor expansion as shown in Eq.(1) when the learning rate ϵ is small:

$$\begin{aligned} \Delta f_1^{g_1+g_2} &\approx -\epsilon \left(g_1^2 + g_1 g_2\right) + o(\epsilon^2) \\ &\leq -\epsilon \left(g_1^2\right) + o(\epsilon^2) \approx \Delta f_1^{g_1} \end{aligned}, \qquad g_1 \cdot g_2 < 0, \tag{1}$$

where (g_1, g_2) denote $(\nabla_{\mathbf{w}_1} f_1, \nabla_{\mathbf{w}_1} f_2)$, and ϵ is the low learning rate($\epsilon = 0.001$). Obviously, the convergence of f_1 is negatively influenced by the gradient of exit-2: g_2. The total loss degradation when updating $\mathbf{w}_1$ is calculated as follows:

$$\Delta L = \Delta f_1^{g_1+g_2} + \Delta f_2^{g_1+g_2} \approx -\epsilon \left(g_1^2 + g_2^2 + 2 g_1 g_2\right) + o(\epsilon^2). \tag{2}$$

The gradient de-conflict approach Pcgrad proposed in [55] projects each conflict gradient onto the normal plane of the other for suppressing the interfering components, which improves the performance of multi-task networks. The gradient re-projection can be formulated as:

$$\hat{g_1} = (g_1 - \frac{g_1 g_2}{\|g_2\|^2} g_2), \hat{g_2} = (g_2 - \frac{g_1 g_2}{\|g_1\|^2} g_1). \tag{3}$$

The loss degradation ΔL by re-projecting both two gradients is:

$$\Delta L_{g_2 \rightleftharpoons g_1} = -\epsilon \left(g_1^2 + g_2^2 - \frac{(g_1 g_2)^2}{\|g_1\|^2} - \frac{(g_1 g_2)^2}{\|g_2\|^2} + 2 g_1 g_2 ((cos\alpha)^2 - 1) \right) + o(\epsilon^2). \tag{4}$$

The $(cos\alpha)$ is the cosine similarity between g_1 and $g_2((cos\alpha) < 0)$.

If we only re-project the gradient g_1, then the total loss degradation $\Delta L_{g_1 \rightarrow g_2}$ is:

$$\Delta L_{g_1 \rightarrow g_2} = -\epsilon \left(g_1^2 + g_2^2 - \frac{(g_1 g_2)^2}{\|g_2\|^2} \right) + o(\epsilon^2). \tag{5}$$

Similarly, when we only re-project the g_2:

$$\Delta L_{g_2 \rightarrow g_1} = -\epsilon \left(g_1^2 + g_2^2 - \frac{(g_1 g_2)^2}{\|g_1\|^2} \right) + o(\epsilon^2). \tag{6}$$

It's obvious that $\Delta L_{g_2 \leftleftharpoons g_1}$ isn't always larger than $\Delta L_{g_1 \to g_2}$ or $\Delta L_{g_2 \to g_1}$. Specifically, assuming $\|g_1\| > \|g_2\|$, the following inequation only holds when the norms of two gradients have a limited difference (please refer to **??**):

$$\Delta L_{g_2 \leftleftharpoons g_1} > \Delta L_{g_2 \to g_1}, \qquad \text{when } \frac{\|g_2\|}{\|g_1\|} > \frac{-0.5(cos\alpha)}{(sin\alpha)^2}. \tag{7}$$

Otherwise, re-projecting the small one gradient g_2 is better than adjusting both gradients. This analysis indicates that the shared parameters are sometimes not actually "shared" equally, i.e. the importances of the shared parameters for different exits are different. In conclusion, it would be better to take account of the importances of the shared parameters for different exits when jointly training multi-exit networks. The toy experiment in Fig.1 verifies this assumption.

3.2 Meta Weighted Gradient Fusion

Assuming $\|g_1\| > \|g_2\|$ and $g_1 \cdot g_2 < 0$, our previous analysis indicates that the optimal gradient fusion policy for $\mathbf{w}_1$ is:

$$g_f = \begin{cases} (1 - \frac{\|g_1\|}{\|g_2\|} cos\alpha) g_1 + (1 - \frac{\|g_2\|}{\|g_1\|} cos\alpha) g_2, & \text{if } \frac{\|g_2\|}{\|g_1\|} > \frac{-0.5(cos\alpha)}{(sin\alpha)^2} \\ (1 - \frac{\|g_1\|}{\|g_2\|} cos\alpha) g_1 + g_2, & \text{otherwise.} \end{cases} \tag{8}$$

It can be seen in Eq.(8) that the larger gradient is always enhanced, which illustrates the preference of the networks to the dominant gradient in fusing gradients of different exits. The above mentioned analysis suggests that it's an potential solution to gradient conflict by weighting the gradients from different exits according to their importances. Hence, we further describe the fusion policy in a weighted-fusion policy:

$$g_f = \eta_1 g_1 + \eta_2 g_2, \qquad \eta_1 > \eta_2 > 0. \tag{9}$$

The Meta-GF algorithm Considering there are always millions of parameters in nowadays deep neural networks, it's computationally expensive to calculate the inner production between gradients and their norms. Inspired by the meta-learning approaches, we instead learn the fusion weights of gradients in a data-driven manner. The number of exits is referred to as n, and the parameter set of the networks is referred to as $\mathbf{W}$. For a given shared parameters $\mathbf{w} \in \mathbf{W}$, the fusion gradient is:

$$g_f^{\mathbf{w}} = \frac{\sum_{i=1}^{n} e^{\eta_i^{\mathbf{w}}} g_i^{\mathbf{w}}}{\sum_{i=1}^{n} e^{\eta_i^{\mathbf{w}}}}. \tag{10}$$

The $\eta = \{\eta_1^{\mathbf{w}}, ..., \eta_n^{\mathbf{w}} | \mathbf{w} \in \mathbf{W}\}$ are the learnable fusion weights, which are dynamically optimized to minimize the cost of the objective functions F:

$$\eta = \arg\min_{\eta} F(\mathbf{W} - \epsilon \frac{\sum_{i=1}^{n} e^{\eta_i} g_i}{\sum_{i=1}^{n} e^{\eta_i}}). \tag{11}$$

The default values of the fusion weights are all set to 1 in the initialization stage. The objective function F is the cross-entropy loss function because we verify the proposed approach on the multi-exit classification network: MSDnet[20] and SDN[26].

It's also important to reduce the noises in the gradients of each exit. In the current mini-batch training settings, the gradient variance in each mini-batch should be considered. The uncertainty of each gradient will cause negative influence on the gradient de-conflict performance. There exists an easy method to estimate the expected gradient of each exit. Inspired by the Reptile algorithm[38], the expected gradient Eg of each task is obtained by training the task independently for one epoch, where the independent training process share the same initial model $\mathbf{W}_0$. This method can be formulated as:

$$\mathbf{W}_{task} = \arg\min_{\mathbf{W}} F_{task}(\mathbf{W}; \mathbf{W}_0, D), \tag{12}$$

$$Eg_{task} = \mathbf{W}_{task} - \mathbf{W}_0, \tag{13}$$

where D is the training set and F_{task} is the objective function. The details of the proposed Meta-GF approaches are illustrated in **Algorithm** 1.

Algorithm 1 Meta Weighted Gradient Fusion:

Input: Initial parameters:$\mathbf{W}_0$, training dataset:D, learning rate:ϵ, fusion weight:η. The number of exits is n. $F = \{F_1, ..., F_n\}$ is the objective function of the exits.

Output: $\mathbf{W}$

while $i <$ MaxIter **do**
  ▼ *1. Calculating expected gradients*:
  for j=1,...,n **do**
    $\hat{\mathbf{W}} = \mathbf{W}_0$
    $\mathbf{W}_j = \arg\min_{\mathbf{W}} F_j(\mathbf{W}; \hat{\mathbf{W}}, D)$
    $g_j = \mathbf{W}_j - \hat{\mathbf{W}}$,
  end for
  ▼ *2. Meta Weighted Gradient Fusion*:
  $\eta = \arg\min_{\eta} F(\mathbf{W}_0 - \epsilon \frac{\sum_{i=1}^{n} e^{\eta_i} g_i}{\sum_{i=1}^{n} e^{\eta_i}}; D)$.
  $\mathbf{W} = \mathbf{W}_0 - \epsilon \frac{\sum_{i=1}^{n} e^{\eta_i} g_i}{\sum_{i=1}^{n} e^{\eta_i}}$
  $\mathbf{W}_0 = \mathbf{W}$
end while
return Y

4 Experiments

To verify the effectiveness of the proposed Meta-GF, we conduct extensive experiments on the representative image classification dataset CIFAR[27] and ILSVRC 2012(ImageNet). Besides, we make detailed analysis of the proposed meta-fusion method. For fair comparison, all of the methods for comparison in this work use the same multi-exit networks: MSDnet[20] and SDN[26].

Datasets. CIFAR100 and CIFAR10 both contain 60000 RGB images of size 32×32. 50000 of them are applied for training and 10000 for test in the two datasets. The images in CIFAR10 and CIFAR100 are corresponding to 10 classes and 100 classes respectively. We adopt the same data augmentation policy as introduced in [29], which includes random crop, random flip and data normalization. We select 5000 of the training sets on CIFAR100 and CIFAR10 respectively for validation. The ImageNet dataset contains 1000 classes, where the input size of images is set to 224×224. The training set have 1.2 million images and we select 50000 of them for validation, where the public validation set of the ImageNet is referred to as the test set in this work because the true test set has not been made public.

Implementation Details. We optimize all models by using stochastic gradient descent with batch size of 64 on CIFAR and 512 on ImageNet. The momentum weight and weight decay are set to 0.9 and 10^{-4} respectively. We train the MSDnet for $maxiter = 300$ epochs on both CIFAR datasets and for $maxiter = 90$ epochs on ImageNet. For the SDN, the maximum epoch is set to 100 on CIFAR datasets. The adjustment of the learning rate is achieved by multi-step policy, where we divide the learning rate by a factor of 10 after $0.5 \times maxiter$ and $0.75 \times maxiter$ epochs. The initial learning rate is 0.1.

The fusion weights are all initialized as 1. Those fusion weights are optimized by Adam optimizer, of which initial learning rate is set to 10^{-1}, and we take the same multi-step policy as above to adjust the learning rate. During an training epoch, we first train each exit independently for estimating the expected updating directions. However, we find that independently training each exit without optimizing other exits might cause negative influence on the Batch Normalization layers of the deeper exits. Therefore, except for the current selected exit, we actually train other exits with a very small learning rate. Then we train the fusion weights with the training sets for one iteration. Finally ,we merge the expected gradients of each task with the proposed Meta-GF.

Compared methods. We compare the proposed Meta-GF with four representative approaches. The Pcgrad[55] and Cagrad[33] are proposed for the multi-task learning problem, and we apply them to the multi-exit neural networks.

- MSDnet[20]/SDN[26]. In this work, the proposed method and other compared training methods all adopt the MSDNet/SDN as the network structure. It serves as a baseline in the experiments. It takes the SGD as the optimizer for training.
- Gradient Equilibrium(GE)[29]. It rescales the magnitude of gradients along its backward propagation path. It helps to reduce gradient variance and stabilize the training procedure.
- Pcgrad[55]. When two gradient conflict with each other, it projects each gradient onto the normal plane of the other for suppressing the interfering components of the gradients.
- Cagrad[33]. To regularize the algorithm trajectory, it looks for an gradient fusion policy that maximizes the worst local improvement of any objective

in the neighborhood of the average gradient. Different from Pcgrad, it aims at forcing the models to converge to a Pareto set of the joint objectives. As same as Pcgrad, we implement the Cagrad on the MSDnet and the SDN to adjust the direction of gradients.

On the CIFAR datasets, we set the exit number of the MSDnet to 7. The depth of 7 exits are $\{4, 6, 8, 10, 12, 14, 16\}$ respectively. The input size of the image is 32×32. We train the MSDnet with three-scale features, i.e, 32×32, 16×16 and 8×8. On the ImageNet, there are 5 exits in the MSDnet, which are respectively inserted at the depth of $\{4, 8, 12, 16, 20\}$. The input size on ImageNet datasets is 224×224, and we use four-scale MSDnet, i.e the multi-scale feature maps are $\{56 \times 56, 28 \times 28, 14 \times 147 \times 7\}$. All of the compared methods use the same MSDnet architectures on the CIFAR and ImageNet. For the SDN-style networks, we take the same training settings as described in [26], and conduct experiments on the Resnet-SDN and Vgg-SDN.

4.1 Prediction accuracy of each exit

In the anytime prediction setting[20], the model maintains a progressively updated distribution over classes, and it can be forced to output its most up-to-date prediction at an arbitrary time[19]. Therefore, the prediction accuracy of different exits is much significant for the performance of anytime prediction.

As illustrated in Table 1, we compare the prediction top-1 accuracy of the MSDnet when using different training approaches on CIFAR. On CIFAR100, the previous gradient adjustment approaches: GE, Cagrad and Pcgrad perform better than the baseline model, which indicates that the adjustment of gradient can effectively alleviate the conflicts between different exits. Especially, Cagrad performs better than other methods at shallow exits which demonstrates that Cagrad successfully maximizes the worst local improvement of any objective in the neighborhood of the average gradient. But it hurts the performance of the deeper exits.

	Params(M)	flops(M)	CIFAR100					CIFAR10				
			MSDnet	GE	Cagrad	Pcgrad	ours	MSDnet	GE	Cagrad	Pcgrad	ours
Exit-1	0.90	56.43	66.41	67.74	**68.78**	67.06	67.97	91.13	92.02	92.19	91.66	**92.38**
Exit-2	1.84	101.00	70.48	71.87	**72.55**	71.37	72.27	92.91	93.53	93.49	93.59	**94.22**
Exit-3	2.80	155.31	73.25	73.81	74.23	74.86	**75.06**	93.98	94.14	94.47	94.32	**94.49**
Exit-4	3.76	198.10	74.02	75.13	74.97	**75.78**	75.77	94.46	94.49	94.45	94.60	**94.96**
Exit-5	4.92	249.53	74.87	75.86	75.35	76.25	**76.38**	94.68	94.73	94.48	94.81	**94.82**
Exit-6	6.10	298.05	75.33	76.23	75.82	76.95	**77.11**	94.78	94.89	94.53	94.83	**94.97**
Exit-7	7.36	340.64	75.42	75.98	76.08	76.71	**77.47**	94.64	94.96	94.48	94.82	**94.97**
Average	-	-	72.83	73.80	73.96	74.14	**74.57**	93.80	94.11	94.01	94.09	**94.54**

Table 1. Classification accuracy of individual classifiers in multi-exit MSDnet on CIFAR-100 and CIFAR10.

Different from the Pcgrad and Cagrad, which treat all the shared parameters as a whole and manipulate the gradients, the Meta-GF aims at softly weighting the gradients of each shared parameter by considering its importance for each

exit. As shown in Table.1, despite that our approach doesn't always achieve the highest score at each exit, the proposed Meta-GF enables the multi-exit MSDnet obtain the best overall accuracy on both the CIFAR10 and CIFAR100 datasets.

To further verify the effectiveness of the proposed Meta-GF, we compare the existing approaches and our method on the ImageNet. The MSDnet trained on the ImageNet have 5 exits. As shown in Table 2, the overall performance of the proposed Meta-GF outperforms GE, Pcgrad and the baseline. The shallow exits of the baseline achieve the best prediction accuracy, but interfere the performance of the deeper exits.

	Params(M)	flops(M)	ImageNet			
			MSDnet	GE	Pcgrad	ours
Exit-1	4.24	339.90	**58.48**	57.75	57.62	57.43
Exit-2	8.77	685.46	**65.96**	65.54	64.87	64.82
Exit-3	13.07	1008.16	68.66	**69.24**	68.93	69.08
Exit-4	16.75	1254.47	69.48	70.27	71.05	**71.67**
Exit-5	23.96	1360.53	71.03	71.89	72.45	**73.27**
Average	-	-	66.72	66.94	66.98	**67.25**

Table 2. Classification accuracy of individual classifiers on ImageNet.

We also conduct experiments on two shallow depth networks proposed in [26], i.e. the Resnet-SDN and the Vgg-SDN. We take the same training settings as described in [26]. The results shown in Table 3 and Table 4 demonstrate that the proposed Meta-GF not only works in the well-designed multi-exit networks– MSDnet, but can also improve the performance of other multi-exit networks. It's worthy noting that we regarded all the filters of each layer as one parameter in the MSDNet and Resnet-SDN, however in the Vgg-SDN, we regard the filters belong to each output channel as a parameter. Because the structure of Vgg is flat-style without skip-connection, if the middle-layers become very task-specific, the performance of the deeper exits will inevitably influenced.

	Params(M)	flops(M)	CIFAR100					CIFAR10				
			SDN-vgg	GE	Cagrad	Pcgrad	ours	SDN-vgg	GE	Cagrad	Pcgrad	ours
Exit-1	0.05	39.76	44.42	44.46	**53.08**	43.59	49.91	69.03	68.97	**76.27**	67.41	74.92
Exit-2	0.29	96.52	61.08	61.0	61.39	**63.02**	61.09	84.72	84.52	86.3	85.28	**88.69**
Exit-3	1.22	153.25	69.8	69.54	70.9	70.04	**71.38**	92.15	92.02	92.4	91.8	**92.75**
Exit-4	1.85	191.08	72.23	72.11	71.55	73.14	**75.77**	92.5	92.62	92.79	92.74	**93.07**
Exit-5	5.47	247.81	72.48	72.32	72.41	72.59	**74.12**	92.46	92.78	92.99	92.75	**93.13**
Exit-6	7.86	285.68	72.63	72.38	72.45	72.54	**74.23**	**93.59**	92.83	93.07	92.7	93.12
Exit-7	15.47	314.45	71.76	71.58	71.43	71.39	**73.1**	92.61	92.85	93.0	**93.69**	93.07
Average	-	-	66.34	66.19	67.60	66.61	**68.51**	88.15	88.08	89.54	88.05	**89.82**

Table 3. Classification accuracy of individual classifiers in multi-exit Vgg-SDN[26] on CIFAR-100 and CIFAR10.

We take further comparisons with the distillation-based works proposed in [3] as shown in Table 5. The knowledge distillation-based methods are mainly applied to provide soft target distributions, or in other words, distill the knowledge by deeper networks for improving the generalization ability of shallow networks. The distillation-based works is complementary to the gradient-adjust ap-

	Params(M)	flops(M)	CIFAR100					CIFAR10				
			SDN-Resnet	GE	Cagrad	Pcgrad	ours	SDN-Resnet	GE	Cagrad	Pcgrad	ours
Exit-1	0.02	19.50	40.20	42.10	**48.73**	40.10	44.41	71.64	71.37	**80.94**	69.74	76.04
Exit-2	0.04	38.54	45.45	46.91	47.05	45.67	**47.17**	78.10	77.11	**80.24**	77.24	78.11
Exit-3	0.10	56.47	59.08	**59.85**	57.77	60.04	59.70	87.32	87.21	86.31	**87.75**	86.43
Exit-4	0.18	75.43	62.40	**63.81**	62.62	63.47	63.25	**89.85**	89.63	88.62	89.79	89.09
Exit-5	0.36	93.32	67.88	**68.52**	67.16	67.78	68.38	91.45	91.51	90.73	**91.53**	91.48
Exit-6	0.67	112.25	70.06	69.88	69.26	69.70	**70.25**	92.26	92.33	91.31	92.17	**92.33**
Exit-7	0.89	126.44	70.02	69.63	68.40	70.07	**70.08**	92.33	92.21	91.19	92.09	**92.87**
Average	-	-	59.29	60.10	60.14	59.54	**60.32**	86.13	85.91	**87.04**	85.76	86.62

Table 4. Classification accuracy of individual classifiers in multi-exit Resnet-SDN[26] on CIFAR-100 and CIFAR10.

	Params(M)	flops(M)	CIFAR100			
			MSDnet	Meta-GF	KD	Meta-GF(KD)
Exit-1	0.90	56.43	50.82	52.44	56.66	57.43
Exit-2	1.84	101.00	54.38	55.37	58.35	59.05
Exit-3	2.80	155.31	56.29	57.51	59.39	60.10
Exit-4	3.76	198.10	57.54	58.83	60.05	60.23
Exit-5	4.92	249.53	58.42	60.28	60.35	60.78
Exit-6	6.10	298.05	58.28	60.55	60.2	61.02
Exit-7	7.36	340.64	58.96	60.55	59.66	60.54
Average	-	-	56.38	57.93	59.23	59.87

Table 5. Classification accuracy of individual classifiers on CIFAR100(150).

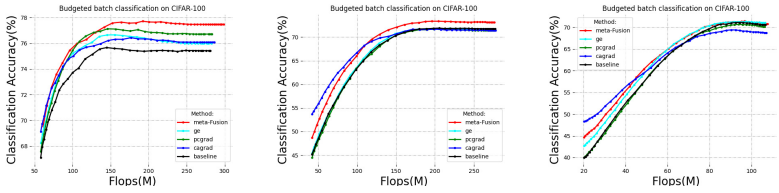
proaches[29, 55, 33], and thus is also complementary to the proposed Meta-GF. For simplicity, we takes 150 samples per class on CIFAR100 datasets for training. As shown in Table 5, by integrating the knowledge distillation with the proposed Meta-GF, the performance of the multi-exit networks can be further improved.

4.2 Adaptive inference

In budgeted batch prediction mode[20], the computational budget is given in advance and the model is supposed to allocate different resources according to the complexity of inputs. For example, "easy" inputs are usually predicted by the shallow exits for saving the computation resources. When the multi-exit network conducts budgeted batch prediction, it forwards the input through the intermediate exits from the shallow ones to the deep ones. If the prediction confidence at certain exit, which is the highest softmax probability in this experiment, is higher than a threshold, then the inference stops at this exit and the network outputs the prediction of this exit as the result. Otherwise, the subsequent exits is evaluated, until a sufficient high confidence has been obtained, or the last exit is evaluated[29]. The threshold is calculated on the validation set as described in [20]. We refer the readers to the work proposed in [20] for more details. As shown in Fig.2, the proposed Meta-GF achieves competitive performance on three kinds of multi-exit networks when compared with other approaches. These results are consistent with the anytime prediction experiments, and demonstrate that the Meta-GF effectively balance the learning behavior of each exit.

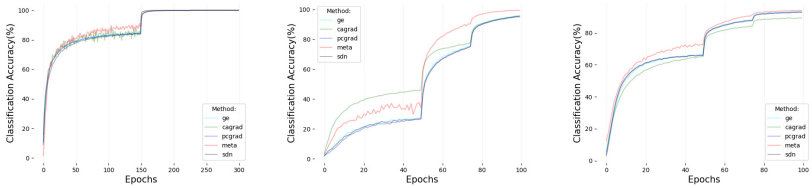
4.3 Analysis about Meta-GF

In this section, we first make analysis about the convergence speed of the multi-exit networks when using the proposed Meta-GF. Then we investigate whether



(a) MSDnet (b) Vgg-SDN (c) Resnet-SDN

Fig. 2. Performance comparison: classification accuracy of budgeted batch classification as a function of average computational budget per image on the CIFAR-100.



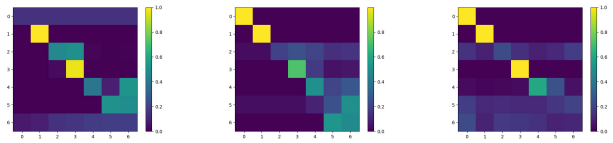
(a) MSDnet (b) Vgg-SDN (c) Resnet-SDN

Fig. 3. The accuracy of the multi-exit networks in the training stage on CIFAR100.

the learned fusion weights have the ability to reflect the importances of the shared parameters for the corresponding exits.

The convergence speed when using the proposed approach As shown in Fig.3, the solid line represents the accuracy of the last exit on the training set. Note that the convergence speed in this work is evaluated by the training accuracy at the specific training iterations. Compared with the previous methods, the proposed Meta-GF obviously improve the convergence speed of the model, which benefits from the meta-learned expected gradient fusion. Yet for the Vgg-SDN, the convergence speed of the model when using the Meta-GF falls behind using the Cagrad at the first 50 epochs, but exceed it at the last 50 epochs. The final performance by using the Meta-GF stills surpass the Cagrad as shown in Table.3.

Analysis about the learned fusion weights We further make analysis of the learned fusion weights on CIFAR100. As mentioned above, the gradient fusion weights η_i of each exit are supposed to reflect the importances of the shared-parameters for the exit. We preliminarily define the parameter w with large η_i^w as the important parameter for the ith exit, where $\frac{e^{\eta_i^w}}{\sum_{j=1}^{n} e^{\eta_j^w}} > 0.5$. As shown in Fig.4, we iteratively prune the important parameters of each exit from the 1st exit to the 7th exit, the relative accuracy degradation is shown along the horizontal axis. It can be seen that when we prune the important parameters of



(a) Resnet-SDN (b) MSDnet (c) Vgg-SDN

Fig. 4. The accuracy degradations when pruning the important shared parameters of different exits.

one exit, it mainly reduces the accuracy of this exit. This result in Fig.4 indicates that the learned fusion weights effectively capture the importances of the share-parameters for each exit. Therefore, it's feasible to use the Meta-GF to alleviate the interference between different exits. It's worthy noting that not all exits in these multi-networks can own sufficient task-specific shared-parameters. Hence in Fig.4, we can still see that pruning the important parameters of some exits doesn't cause the largest accuracy degradation to the associated exits.

5 Conclusion and discussion

In this work, we propose a meta gradient fusion approach to tackle the gradient conflict problems when training multi-exit depth adaptive networks. The proposed Meta-GF takes a meta-learned weighted fusion policy to combine the expected gradients from each exit. We conduct extensive experiments on CIFAR and ImageNet, and the experimental results demonstrate the effectiveness of our approach.

However there is still room for improvement, we will further develop the Meta-GF through three aspects. Firstly, the meta fusion progress, in the current settings, doesn't take account of the parameters of the BatchNorm layer, and we will extend the Meta-GF to effectively fuse the BatchNorm layer either. Secondly, we plan to explicitly design objective functions for alleviating the gradient conflicts between different exits, which is not used in this work. Finally, we believe that the idea of the Meta-GF can be relevant to the network pruning researches to some extent. Though the learned fusion weights by the Meta-GF cannot be directly applied to prune the multi-exit networks for each exit, the results in Fig.4 demonstrate a promising direction for our future researches, which means that we can combine the Meta-GF with the network pruning approaches for better disentangling the shared parts of the multi-exit networks, and also for alleviating the interference between different exits.

Acknowledgement

This work was supported by the National Natural Science Foundation of China (NSFC) under Grants 61825305 and 61973311.

References

1. Bakhtiarnia, A., Zhang, Q., Iosifidis, A.: Multi-exit vision transformer for dynamic inference (2021)
2. Bolukbasi, T., Wang, J., Dekel, O., Saligrama, V.: Adaptive neural networks for efficient inference. In: International Conference on Machine Learning. pp. 527–536. PMLR (2017)
3. Carion, N., Massa, F., Synnaeve, G., Usunier, N., Kirillov, A., Zagoruyko, S.: End-to-end object detection with transformers. In: European Conference on Computer Vision. pp. 213–229. Springer (2020)
4. Chen, X., Dai, H., Li, Y., Gao, X., Song, L.: Learning to stop while learning to predict. In: International Conference on Machine Learning. pp. 1520–1530. PMLR (2020)
5. Chen, Y., Dai, X., Liu, M., Chen, D., Yuan, L., Liu, Z.: Dynamic convolution: Attention over convolution kernels. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 11030–11039 (2020)
6. Chen, Z., Badrinarayanan, V., Lee, C.Y., Rabinovich, A.: Gradnorm: Gradient normalization for adaptive loss balancing in deep multitask networks. In: International Conference on Machine Learning. pp. 794–803. PMLR (2018)
7. Dai, X., Kong, X., Guo, T.: Epnet: Learning to exit with flexible multi-branch network. In: Proceedings of the 29th ACM International Conference on Information & Knowledge Management. pp. 235–244 (2020)
8. Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., et al.: An image is worth 16x16 words: Transformers for image recognition at scale. In: International Conference on Learning Representations (2020)
9. Du, S., Lee, J.: On the power of over-parametrization in neural networks with quadratic activation. In: International Conference on Machine Learning. pp. 1329–1338. PMLR (2018)
10. Duggal, R., Freitas, S., Dhamnani, S., Chau, D.H., Sun, J.: Elf: An early-exiting framework for long-tailed classification. arXiv preprint arXiv:2006.11979 (2020)
11. Elbayad, M., Gu, J., Grave, E., Auli, M.: Depth-adaptive transformer. In: ICLR 2020-Eighth International Conference on Learning Representations. pp. 1–14 (2020)
12. Fan, A., Grave, E., Joulin, A.: Reducing transformer depth on demand with structured dropout. In: International Conference on Learning Representations (2019)
13. Farhadi, A., Redmon, J.: Yolov3: An incremental improvement. In: Computer Vision and Pattern Recognition. pp. 1804–2767. Springer Berlin/Heidelberg, Germany (2018)
14. Gao, X., Zhao, Y., Dudziak, Ł., Mullins, R., Xu, C.z.: Dynamic channel pruning: Feature boosting and suppression. In: International Conference on Learning Representations (2018)
15. Guo, M., Haque, A., Huang, D.A., Yeung, S., Fei-Fei, L.: Dynamic task prioritization for multitask learning. In: Proceedings of the European conference on computer vision (ECCV). pp. 270–287 (2018)
16. Guo, S., Alvarez, J.M., Salzmann, M.: Expandnets: Linear over-parameterization to train compact convolutional networks. Advances in Neural Information Processing Systems **33** (2020)
17. Han, Y., Huang, G., Song, S., Yang, L., Wang, H., Wang, Y.: Dynamic neural networks: A survey. IEEE Transactions on Pattern Analysis and Machine Intelligence pp. 1–1 (2021). https://doi.org/10.1109/TPAMI.2021.3117837

18. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 770–778 (2016)
19. Huang, C., Lucey, S., Ramanan, D.: Learning policies for adaptive tracking with deep feature cascades. In: Proceedings of the IEEE international conference on computer vision. pp. 105–114 (2017)
20. Huang, G., Chen, D., Li, T., Wu, F., van der Maaten, L., Weinberger, K.Q.: Multi-scale dense networks for resource efficient image classification. arXiv preprint arXiv:1703.09844 (2017)
21. Huang, G., Sun, Y., Liu, Z., Sedra, D., Weinberger, K.Q.: Deep networks with stochastic depth. In: European conference on computer vision. pp. 646–661. Springer (2016)
22. Javaloy, A., Valera, I.: Rotograd: Gradient homogenization in multi-task learning (2021)
23. Jeon, G.W., Choi, J.H., Kim, J.H., Lee, J.S.: Larvanet: Hierarchical super-resolution via multi-exit architecture. In: European Conference on Computer Vision. pp. 73–86. Springer (2020)
24. Jiang, Y.G., Cheng, C., Lin, H., Fu, Y.: Learning layer-skippable inference network. IEEE Transactions on Image Processing **29**, 8747–8759 (2020)
25. Jie, Z., Sun, P., Li, X., Feng, J., Liu, W.: Anytime recognition with routing convolutional networks. IEEE transactions on pattern analysis and machine intelligence (2019)
26. Kaya, Y., Hong, S., Dumitras, T.: Shallow-deep networks: Understanding and mitigating network overthinking. In: International Conference on Machine Learning. pp. 3301–3310. PMLR (2019)
27. Krizhevsky, A., Hinton, G., et al.: Learning multiple layers of features from tiny images (2009)
28. Li, F., Li, G., He, X., Cheng, J.: Dynamic dual gating neural networks. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 5330–5339 (2021)
29. Li, H., Zhang, H., Qi, X., Yang, R., Huang, G.: Improved techniques for training adaptive deep networks. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 1891–1900 (2019)
30. Li, Y., Song, L., Chen, Y., Li, Z., Zhang, X., Wang, X., Sun, J.: Learning dynamic routing for semantic segmentation. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 8553–8562 (2020)
31. Li, Y., Chen, Y.: Revisiting dynamic convolution via matrix decomposition. In: International Conference on Learning Representations (2021)
32. Liu, B., Rao, Y., Lu, J., Zhou, J., Hsieh, C.J.: Metadistiller: Network self-boosting via meta-learned top-down distillation. In: European Conference on Computer Vision. pp. 694–709. Springer (2020)
33. Liu, B., Liu, X., Jin, X., Stone, P., Liu, Q.: Conflict-averse gradient descent for multi-task learning. Advances in Neural Information Processing Systems **34** (2021)
34. Liu, L., Li, Y., Kuang, Z., Xue, J., Chen, Y., Yang, W., Liao, Q., Zhang, W.: Towards impartial multi-task learning. ICLR (2021)
35. Liu, Y., Meng, F., Zhou, J., Chen, Y., Xu, J.: Faster depth-adaptive transformers. In: Proceedings of the AAAI Conference on Artificial Intelligence. vol. 35, pp. 13424–13432 (2021)
36. Ma, N., Zhang, X., Huang, J., Sun, J.: Weightnet: Revisiting the design space of weight networks. In: European Conference on Computer Vision. pp. 776–792. Springer (2020)

37. Molchanov, P., Tyree, S., Karras, T., Aila, T., Kautz, J.: Pruning convolutional neural networks for resource efficient inference. In: 5th International Conference on Learning Representations, ICLR 2017-Conference Track Proceedings (2019)
38. Nichol, A., Achiam, J., Schulman, J.: On first-order meta-learning algorithms. arXiv preprint arXiv:1803.02999 (2018)
39. Passalis, N., Raitoharju, J., Tefas, A., Gabbouj, M.: Efficient adaptive inference for deep convolutional neural networks using hierarchical early exits. Pattern Recognition **105**, 107346 (2020)
40. Phuong, M., Lampert, C.H.: Distillation-based training for multi-exit architectures. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 1355–1364 (2019)
41. Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., Chen, L.C.: Mobilenetv2: Inverted residuals and linear bottlenecks. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 4510–4520 (2018)
42. Schwartz, R., Stanovsky, G., Swayamdipta, S., Dodge, J., Smith, N.A.: The right tool for the job: Matching model and instance complexities. In: Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics. pp. 6640–6651 (2020)
43. Sener, O., Koltun, V.: Multi-task learning as multi-objective optimization. Advances in neural information processing systems **31** (2018)
44. Shen, J., Wang, Y., Xu, P., Fu, Y., Wang, Z., Lin, Y.: Fractional skipping: Towards finer-grained dynamic cnn inference. In: Proceedings of the AAAI Conference on Artificial Intelligence. vol. 34, pp. 5700–5708 (2020)
45. Tan, M., Le, Q.: Efficientnet: Rethinking model scaling for convolutional neural networks. In: International Conference on Machine Learning. pp. 6105–6114. PMLR (2019)
46. Veit, A., Belongie, S.: Convolutional networks with adaptive inference graphs. In: Proceedings of the European Conference on Computer Vision (ECCV). pp. 3–18 (2018)
47. Wang, X., Yu, F., Dou, Z.Y., Darrell, T., Gonzalez, J.E.: Skipnet: Learning dynamic routing in convolutional networks. In: Proceedings of the European Conference on Computer Vision (ECCV). pp. 409–424 (2018)
48. Wang, X., Li, Y.: Gradient deconfliction-based training for multi-exit architectures. In: 2020 IEEE International Conference on Image Processing (ICIP). pp. 1866–1870. IEEE (2020)
49. Wang, X., Li, Y.: Harmonized dense knowledge distillation training for multi-exit architectures. In: Proceedings of the AAAI Conference on Artificial Intelligence. vol. 35, pp. 10218–10226 (2021)
50. Wołczyk, M., Wójcik, B., Bałazy, K., Podolak, I.T., Tabor, J., Śmieja, M., Trzcinski, T.: Zero time waste: Recycling predictions in early exit neural networks. Advances in Neural Information Processing Systems **34**, 2516–2528 (2021)
51. Wu, Z., Nagarajan, T., Kumar, A., Rennie, S., Davis, L.S., Grauman, K., Feris, R.: Blockdrop: Dynamic inference paths in residual networks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 8817–8826 (2018)
52. Xin, J., Tang, R., Lee, J., Yu, Y., Lin, J.: Deebert: Dynamic early exiting for accelerating bert inference. In: Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics. pp. 2246–2251 (2020)
53. Yang, B., Bender, G., Le, Q.V., Ngiam, J.: Condconv: Conditionally parameterized convolutions for efficient inference. In: Advances in Neural Information Processing Systems. pp. 1307–1318 (2019)

54. Yang, L., Han, Y., Chen, X., Song, S., Dai, J., Huang, G.: Resolution adaptive networks for efficient inference. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 2369–2378 (2020)
55. Yu, T., Kumar, S., Gupta, A., Levine, S., Hausman, K., Finn, C.: Gradient surgery for multi-task learning. Advances in Neural Information Processing Systems **33** (2020)
56. Zhou, J., Jampani, V., Pi, Z., Liu, Q., Yang, M.H.: Decoupled dynamic filter networks. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 6647–6656 (2021)
57. Zhou, W., Xu, C., Ge, T., McAuley, J., Xu, K., Wei, F.: Bert loses patience: Fast and robust inference with early exit. Advances in Neural Information Processing Systems **33**, 18330–18341 (2020)