Improved Techniques for Training Adaptive Deep Networks

Hao Li[1*] Hong Zhang[2*] Xiaojuan Qi[3] Ruigang Yang[2] Gao Huang[1†]

[1]Tsinghua University [2]Baidu Inc. [3]University of Oxford

{lihaothu,fykalviny,qxj0125}@gmail.com

yangruigang@baidu.com gaohuang@tsinghua.edu.cn

Abstract

Adaptive inference is a promising technique to improve the computational efficiency of deep models at test time. In contrast to static models which use the same computation graph for all instances, adaptive networks can dynamically adjust their structure conditioned on each input. While existing research on adaptive inference mainly focuses on designing more advanced architectures, this paper investigates how to train such networks more effectively. Specifically, we consider a typical adaptive deep network with multiple intermediate classifiers. We present three techniques to improve its training efficacy from two aspects: 1) a Gradient Equilibrium algorithm to resolve the conflict of learning of different classifiers; 2) an Inline Subnetwork Collaboration approach and a One-for-all Knowledge Distillation algorithm to enhance the collaboration among classifiers. On multiple datasets (CIFAR-10, CIFAR-100 and ImageNet), we show that the proposed approach consistently leads to further improved efficiency on top of state-of-the-art adaptive deep networks.

1. Introduction

Convolutional neural networks (CNNs) have gained remarkable success on a variety of visual recognition tasks [21, 10, 27, 9]. Modern CNNs such as GoogleNet [32], ResNet [10] and DenseNet [16] are endowed with unprecedented network depth to achieve state-of-the-art accuracy. However, very deep models usually come along with high computational cost, which prevents them from performing real-time inference on resource-constrained platforms like smart phones, wearable devices and drones.

Extensive efforts have been made to improve the inference efficiency of deep CNNs in recent years. Popular approaches include efficient architecture design [30, 28, 15, 40], network pruning [8, 23, 26], weight quantization [4, 8, 17] and adaptive inference [7, 14, 2, 35, 6, 34]. Among them, adaptive inference is gaining increasing attention recently, due to its remarkable advantages. First, it is compatible with almost all the other approaches, i.e., adaptive inference can be performed on highly optimized architectures like MobileNets [30] and ShuffleNets [28], and can also benefit from model pruning and weight quantization. Second, by conditioning the computation of a deep model on its inputs, adaptive inference can save a considerable amount of computational cost on "easy" samples and/or less important regions, drastically reducing the average inference time. Third, adaptive inference algorithms usually have a set of tunable parameters that dynamically control the accuracy-speed tradeoff. This is a valuable property in many scenarios, where the computational budget may change over time or vary across different devices. In contrast to static models which have a fixed computational cost, adaptive models are able to trade accuracy for speed or vise verse on-the-fly, to meet the dynamically changing demand.

Existing works on adaptive inference in the context of deep learning mainly focus on designing more specialized network architectures [14, 33, 34] or better inference algorithms [2]. In comparison, less effort has been made to improve the training process. But in fact, adaptive CNNs usually have quite different architectures as conventional deep models, and training strategies optimized for the later may not be optimal for adaptive models. In this paper, we consider a representative type of adaptive models that have multiple intermediate classifiers at different depths of the network. With this architecture, adaptive computation can be performed by early exiting "easy" samples to speed up the inference. The multi-scale dense network (MSDNet) proposed in [14] represents the state-of-the-art of this type of models. We aim to improve the training efficacy of such multi-exits networks from the following two perspectives.

First, we need to resolve the conflict while jointly optimizing all the classifiers. It has been observed in [14] that the individual classifiers in the network tend to negatively affect the learning of one other, and [31] discussed that the backbone network may not converge well due to

*First two authors contributed equally

†Corresponding author

the accumulataion of gradients from several classifiers in a multi-head neural network. By introducing dense connections, MSDNet has addressed the problem that early classifiers interfere with later ones. However, deep classifiers may also negatively affect earlier classifiers. To this end, we present a *Gradient Equilibrium (GE)* technique that rescale the magnitude of gradients along its backward propagation path. This allows the gradient to have a constant scale across the network, which helps to reduce gradient variance and stabilize the training procedure.

Second, we aim to encourage collaboration among different classifiers. This is achieved by introducing two modules, named *Inline Subnetwork Collaboration (ISC)* and *One-for-all Knowledge Distillation (OFA)*, respectively. The motivation for ISC is that later exits may benefit from the prediction of early exits. Therefore, we use the prediction logits from previous stage as a prior to facilitate the learning of current and subsequent classifiers. The OFA follows from the intuition that the last exit always yield the highest accuracy among all the classifiers, and thus it could serve as a teacher model whose knowledge could be distilled into earlier exits.

We conduct extensive experiments on three image-classification datasets (CIFAR-10, CIFAR-100, and ImageNet). The experiments demonstrate that the proposed techniques consistently improve the efficiency of state-of-the-art adaptive deep networks.

2. Related Work

Computationally Efficient Deep Networks. Approaches to computationally efficient deep networks can be summarized as follows. One stream focuses on designing efficient network architectures [30, 28, 15, 40, 13], including depth-wise separable convolution [30], point-wise group convolution with channel shuffling [39], and learned group convolution [15], to name a few. The other line of research explores methods to prune [8, 23, 11] or quantize [4, 8, 17] neural network weights. These strategies are effective when neural networks have a substantial amount of redundant weights, which can be safely removed or quantized without sacrificing accuracy.

Adaptive Inference. Recently, a new emerging direction which employs adaptive learning for efficient inference has drawn increasing research attention, with representative works proposed in [14, 2, 25, 35, 19, 6, 24, 38, 36, 29, 37, 18]. Adaptive inference aims at achieving efficient resource allocation during the inference stage without sacrificing accuracy, by strategically save computation on "easy" samples. Compared with other directions to improve network efficiency, adaptive inference gains advantages due to its compatibility, flexibility and high performance.

Most prior works are dedicated to learning adaptive network topology selection policies. Bolukbasi *et al*. [2] adopted an ensemble model with multiple deep networks of varying size, and proposed to learn an adaptive decision function to determine in which stage the example should exit. Huang *et al*. [14] designed a novel multi-scale convolutional network with multiple intermediate classifiers with various computational budgets, which can be adaptively selected during the inference stage. On top of ResNet [10] architecture, Veit *et al*. [35] and Wang *et al*. [36] designed gating functions to dynamically choose layers for efficient inference. Figurnov *et al*. [6] further made the gating policy adaptive to spatial locations. To further enable adaptive inference in pixel-labeling tasks, pixel-wise attentional gating (PAG) [19] was introduced in [19] for adaptively selecting a subset of spatial locations to process, and an RNN architecture was proposed in [29] for dynamically determining the number of RNN steps according to allowed time budget. Adaptive inference was also studied in accelerating visual tracking systems in [38].

Almost all the prior works focus on designing network architectures or algorithms for adaptive inference. In this work, we made an orthogonal effort by exploring effective strategies to facilitate the training of deep networks for adaptive inference. Our method is model-agnostic and can be applied to various adaptive inference architectures with multiple intermediate classifiers including [14, 2, 29]

Knowledge Distillation. Our work is also related to knowledge distillation strategies explored in [12, 22, 3, 1], in which outputs from teacher networks are utilized to supervise the training of student networks. Different from previous trials in training separate teacher models in advance, Lan *et al*. [22] proposed a one-stage online distillation framework by utilizing multi-branch network ensemble to enhance the target network. Our *One-for-all Knowledge Distillation* strategy also shares a spirit similar to these knowledge distillation strategies by acquiring knowledge from larger models with higher accuracy. However, in contrast to previous work deploying knowledge distillation to obtain better student models, we show such strategies can promote collaborations between multi-scale classifiers inside one single network, and improve the efficacy of adaptive inference.

3. Method

In this section, we first set up the adaptive inference model with multiple exits (classifiers). Then we discuss our proposed techniques for improving the training in detail. Roughly, the first technique, *Gradient Equilibrium (GE)*, is introduced to resolve the gradients conflict of different classifiers; the second and third techniques, *Inline Subnetwork Collaboration (ISC)* and *One-for-all Knowledge Dis-*

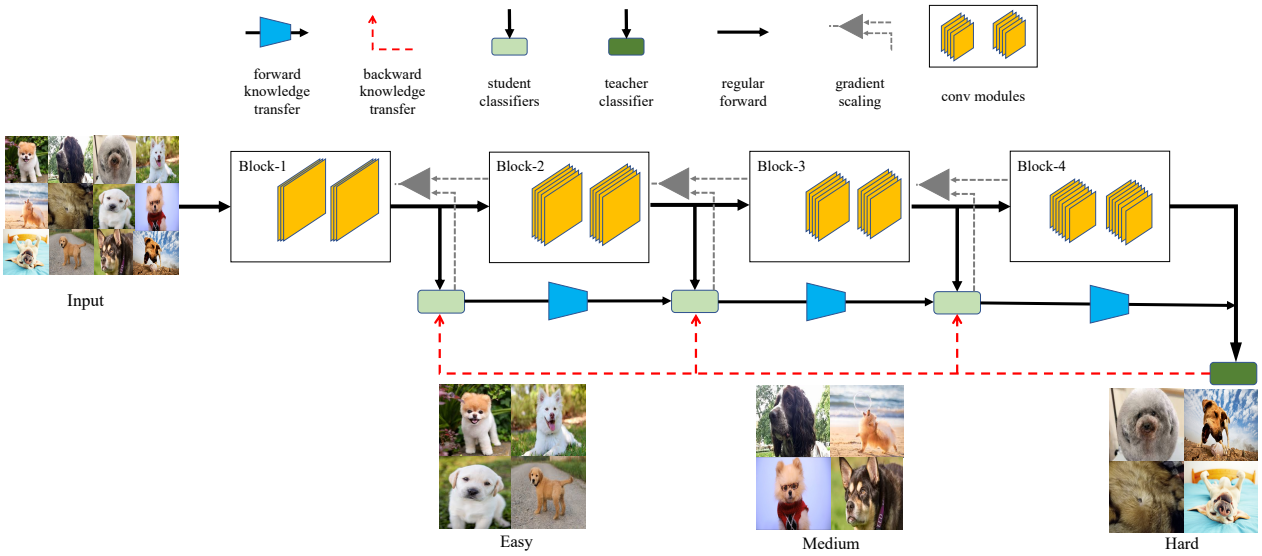


Figure 1. Overview of the proposed training strategies for adaptive inference on a deep convolutional network. **Gradient Equilibrium (GE)** is applied to resolve the gradients conflict among different exits. **Inline Subnetwork Collaboration (ISC)** and **One-for-all Knowledge Distillation (OFA)** are proposed to enhance the collaboration of different classifiers.

tillation (OFA), are proposed to enhance the collaboration among classifiers. Figure 1 gives an overview of the proposed approach.

Adaptive inference model. We set up the adaptive inference model as a network that is composed of k classifiers. The model can be viewed as a conventional CNN with $k-1$ intermediate classifiers attached at varying depths of the network. Each classifier is also referred to as an *exit*. The model can generate a set consisting of k predictions, one from each of the exits, i.e.,

$$[y_1, ..., y_k] = f(x;\theta) = [f_1(x;\theta_1), \ldots, f_k(x;\theta_k)],$$

where x is the input image, and f_i and θ_i $(i = 1, \ldots, k)$ represent the transformation learned by the i-th classifier and its corresponding parameters, respectively. Note that θ_i's have shared parameters here.

At test time, the inference is performed dynamically conditioned on each input. Formally, the prediction for a test sample x is given by $\hat{y} = f_{I(x)}(x, \theta_{I(x)})$, where $I(x) \in \{1, \ldots, k\}$ is a function of x, which is usually obtained by a certain decision function. In our experiment, we simply follow [14] to use a confidence-based approach to compute $I(x)$.

3.1. Gradient Equilibrium

Adaptive inference can be considered as a sequential prediction process by a set of subnetworks. A straightforward way to train an adaptive network is to train the subnetworks sequentially. However, this method is far from optimal due to the conflict between two optimization goals: to learn discriminative features for the current classifier, and to maintain necessary information for generating high-quality features for later classifiers [14]. A more effective training strategy is to jointly optimize all the subnetworks. For example, the MSDNet [14] minimizes a weighted cumulative loss function:

$$L(y, f(x;\theta)) = \sum_i \lambda_i \mathrm{CE}(y, f_i(x;\theta_i)), \tag{1}$$

where $\lambda_i > 0$ is the coefficient for the i-th $(i = 1, \ldots, k)$ classifier, and $\mathrm{CE}(\cdot,\cdot)$ denotes the cross-entropy loss function. In MSDNet, all the λ_i's are simply set to 1.

This form of loss functions may lead to a *gradient imbalance* issue due to the overlap of the subnetworks. Specifically, consider training a k-exit adaptive network using the sum of the cross-entropy losses of all the classifiers. The backward graph can be described by a binary tree with depth k, where the gradients come from leave nodes and propagate from child nodes to father nodes. The gradient of the i-th block is contributed by the i-th node as well as the subsequent $(k-i)$ leave nodes:

$$\nabla_{w_i} L = \sum_{i \le j \le k} \lambda_i \nabla_{w_i} \mathrm{CE}_j, \tag{2}$$

where w_i denotes the features at the i-th stage.

From the above equation, it is easy to see that the total variance of ∇L can become very large as the gradients propagate backward. Formally, consider a situation that the gradients of the loss w.r.t. each w_i are irrelevant, and the total variance of $\nabla_{w_i} L$ is computed by:

$$\mathrm{Var}(\nabla_{w_i} L) = \sum_{i \le j \le k} \lambda_j^2 \mathrm{Var}(\nabla_{w_i} \mathrm{CE}_j). \tag{3}$$

As the number of subnetworks increases, the variance of the gradient may grow overly large, leading to unstable training. Note that this issue can be fixed by averaging the cumulative loss, i.e., $\frac{1}{k}\sum_i \lambda_i \text{CE}_i$, but it tends to result in overly small gradients, which hinders the convergence.

To address this issue, we propose a **Gradient Equilibrium (GE)** method which re-normalizes the gradients at father nodes while maintaining the information flow in the forward procedure. The GE method consists of a series of *gradient re-scaling* operations $R(\cdot; s)$ where s is a scaling factor:

$$R(x; s) = x; \quad \nabla_x R(x; s) = s. \tag{4}$$

To stabilize the backward procedure and resolve the gradient conflict, we propose to re-normalize the gradients in the following manner. For the i-th branch, we add two re-scaling modules for the gradients contributed by the current i-th classifier and the subsequent $(k-i)$ classifiers, setting their s to $\frac{1}{k-i+1}$ and $\frac{k-i}{k-i+1}$, respectively. This ensures that the gradients have a bounded scale. To see this, we first calculate the gradient of L_j w.r.t. w_i, with the re-scaling factors given above:

$$\begin{aligned}\nabla^{(\text{GE})}_{w_i} L_j &= \prod_{i \le h < j} \frac{k-h}{k-h+1} \times \frac{1}{k-j+1} \times \nabla_{w_i} L_j \\ &= \frac{1}{k-i+1} \nabla_{w_i} L_j.\end{aligned} \tag{5}$$

For simplicity, we let $n = k-i+1$ and $X_j = \nabla_{w_i} L_{i+j-1}$. Then we have

$$\begin{aligned}&\text{Var}\left(\sum_{j=i}^{k} \nabla^{(\text{GE})}_{w_i} L_j\right) \\ &= \text{Var}\left(\frac{1}{k-i+1}\sum_{j=i}^{k} \nabla_{w_i} L_j\right) \\ &= \text{Var}\left(\frac{1}{n}\sum_{j=1}^{n} X_j\right) \\ &= \frac{1}{n^2}\left(\sum_{j=1}^{n} \text{Var}(X_j) + \sum_{m \ne j} \text{Cov}(X_m, X_j)\right) \\ &\le \frac{1}{n^2}\left(\sum_{j=1}^{n} \text{Var}(X_j) + \sum_{m \ne j} \sqrt{\text{Var}(X_m)\text{Var}(X_j)}\right) \\ &\le \frac{1}{n^2}\left(n \max_l(\text{Var}(X_l)) + n(n-1)\max_l(\text{Var}(X_l))\right) \\ &\le \frac{2}{n^2} * n^2 * \max_l(\text{Var}(X_l)) \\ &= 2\max_l(\text{Var}(X_l)) < \infty\end{aligned} \tag{6}$$

3.2. Forward Knowledge Transfer

In this and the following subsections, we aim to encourage collaboration among different classifiers in adaptive networks. In existing work, different exits are usually treated as independent models, except that their losses are simply summed up during training process. In fact, these classifiers heavily share parameters, and they are combined to solve the same task at test time, hinting that a collaborated learning process may significantly improve the training efficacy. Therefore, we propose two approaches to distill this insight into practical algorithms.

Our first approach is to promote forward knowledge transfer in adaptive networks. Specifically, we add a *knowledge transfer path* between every two adjacent classifiers, to directly bypass the prediction at the i-th stage to the $(i+1)$-th classifier ($i = 1, \ldots, k-1$). The knowledge transfer path may correspond to a tiny fully connected network or some functions without learning ability. Our experimental results show that even the simplest identity transform improves the performance of adaptive inference, where each classifier (except the very first one) can be considered as performing residual learning. Note that in this case, we discard the gradients along knowledge transform path in back propagation to prevent a classifier from being negatively affected by the latter ones.

Similar to the Knowledge Distillation algorithm [12], we use the logits of the i-th classifier as the knowledge to facilitate the learning of its subsequent classifier, and we call the above approach **Inline Subnetwork Collaboration (ISC)**. Although being very simple, ISC consistently improves the performance of adaptive networks in our experiments.

3.3. Backward Knowledge Transfer

In the previous subsection, we introduce how to use the prediction from an early classifier to boost the performance for the latter classifiers. Here we introduce an approach to utilize the deepest classifier to help the learning of shallow classifiers. In a k-exit adaptive network, the last classifier usually achieves the best accuracy due to its highest capacity. This motivates us to adopt the knowledge distillation algorithm in the network. We call the approach **One-for-all Knowledge Distillation (OFA)**, as all the intermediate exits are supervised by the last classifier.

In specific, the loss function for the i-th classifier consists of two parts weighted by a coefficient α:

$$L_i = \alpha \text{CE}_i + (1-\alpha)\text{KLD}_i, \tag{7}$$

where CE_i is the Cross-Entropy loss, and KLD_i quantifies the alignment of *soft* class probabilities between the teacher and student models using the Kullback Leibler divergence:

$$\text{KLD}_i = -\sum_{c \in Y} p_k(c \mid x; \theta, T) \log \frac{p_i(c \mid x; \theta, T)}{p_k(c \mid x; \theta, T)}. \tag{8}$$

4. Experiments

To demonstrate the effectiveness of our approach, we conducted extensive experiments on three representative image classification datasets, i.e., the CIFAR-10, CIFAR-100 [20] and ILSVRC 2012 (ImageNet) [5]. In addition, ablation studies are performed to analyze the three components of our method. All of our experiments are conducted on the multi-scale dense network (MSDNet) proposed in [14], with the model re-implemented in PyTorch. Code to reproduce our results is avaliable at https://github.com/kalviny/IMTA.

Datasets. The CIFAR-10 and CIFAR-100 datasets contain RGB images of size 32×32, corresponding to 10 and 100 classes, respectively. They both contain $50,000$ images for training and $10,000$ images for testing. Following [14], we hold out $5,000$ training images as a validation set to search the confidence threshold for adaptive inference. We apply standard data augmentation schemes [10]: 1) images are zero-padded with 4 pixels on each side, and then randomly cropped to produce 32×32 inputs; 2) images are horizontally flipped with probability 0.5; 3) RGB channels are normalized by subtracting the corresponding channel mean and divided by their standard deviation.

The ImageNet dataset contains $1,000$ classes, with 1.2 million training images and $50,000$ for testing [1]. We hold out $50,000$ images from the training set as the validation set. We follow the practice in [10, 16] for data augmentation at training time. At test time, images are firstly rescaled to 256×256 followed by a single 224×224 center crop, which are finally classified by the network.

Training Details. On the two CIFAR datasets, we optimize all models using stochastic gradient descent (SGD) with a mini-batch size 64. We use Nesterov momentum with a momentum weight of 0.9 without dampening, and a weight dacay of 10^{-4}. The training is split into two phases. In phase I, the models are trained from scratch with Gradient Equilibrium for 300 epochs, with an initial learning rate of 0.1, which is further divided by a factor of 10 after 150 and 225 epochs. In phase II, we start with the model obtained from phase I, and fine-tune only the last layer of each classifier with the proposed One-for-all Knowledge Distillation (OFA) and Inline Subnetwork Collaboration (ISC). This stage lasts for 180 epochs, with an initial learning rate of 0.1 and divided by 10 after 90 and 135 epochs, respectively. We apply the same training scheme to the ImageNet dataset, except that we increase the mini-batch size to 256,

[1]This subset is usually referred to as the validation set, as the true test set has not been made public. But in order to avoid confusion with the additional validation set we hold out from the training set, we view this subset as our test set.

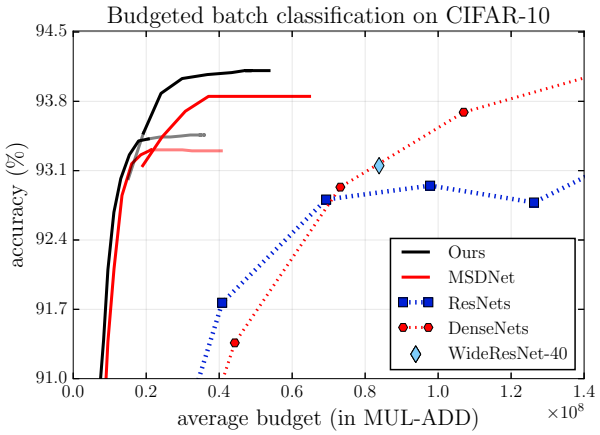


Figure 2. Accuracy (top-1) of *budgeted batch classification* as a function of average computational budget per image on the CIFAR-10.

and all the models are trained for 90 epochs both in phase I and II with learning rate drops after 30 and 60 epochs.

Adaptive inference with MSDNet. Following [14], we evaluate our model in the adaptive inference setting. For a given input image, we forward it through the intermediate classifiers in an one-by-one manner. At each exit, we compare the prediction confidence, which is the highest softmax probability in our experiment, to a threshold, which is dependent on the given computational budget. If the current classifier is sufficiently confident about its prediction, i.e., the confidence value is greater than the threshold, the current prediction is used as the final prediction, and the latter classifiers are not evaluated. Otherwise, the subsequent classifier is evaluated, until a sufficient high confidence has been obtained, or the last classifier is evaluated. Intuitively, "easy" examples are predicted by early classifiers, while only "hard" examples are propagated through the latter classifiers of the network. In practice, most samples are relatively easy, thus this adaptive evaluation procedure can drastically improve the inference efficiency by saving computation on those large portion of "easy" samples in the dataset.

Compared Models. To verify the effectiveness of our approach for adaptive inference, we mainly compare with the following baseline models.

- MSDNet [14]. As our proposed learning strategies are also performed on MSDNet, it serves as a direct baseline for our experiments.

- ResNet [10] and DenseNet [16]. We also compare our approach with ResNets and DenseNets. We do not perform adaptive inference on these models, since they

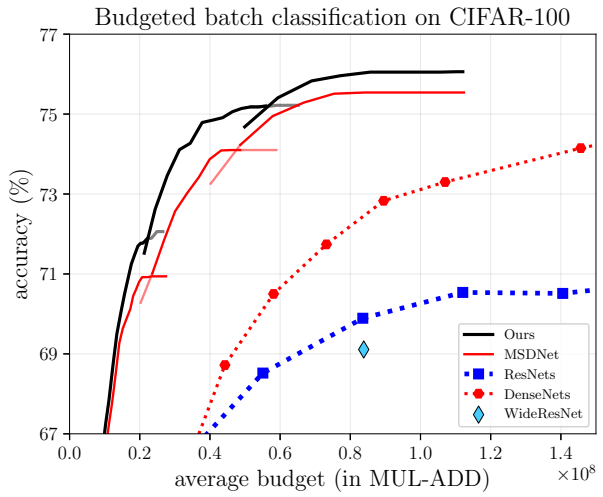
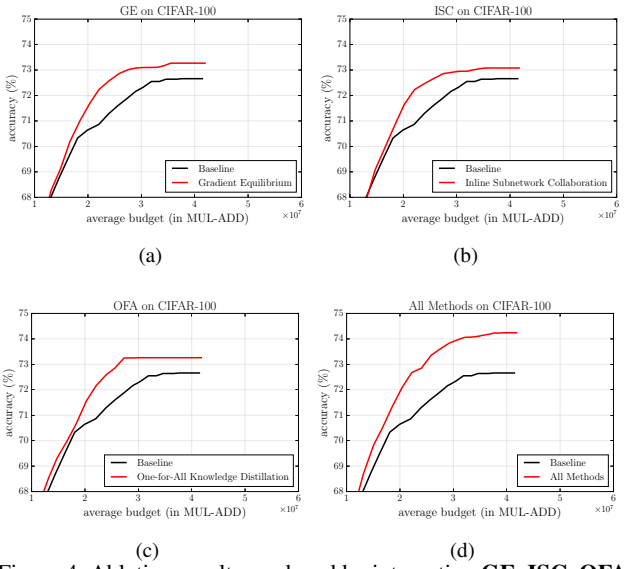


Figure 3. Accuracy (top-1) of *budgeted batch classification* as a function of average computational budget per image on the CIFAR-100.



(a) (b) (c) (d)

Figure 4. Ablation results produced by integrating **GE, ISC, OFA** on CIFAR-100.

are not designed for this purpose, and are shown to yield inferior results compared to MSDNet [14].

As we mainly focus on the training strategy of adaptive CNNs, we do not compare with efficient models with more advance architecure designs, such as the MobileNet [13], ShuffleNet [39] and NASNet [40]. As discussed earlier in the paper, the architecture innovations for these models, like the depth separable convolutions, are orthogonal to adaptive inference methods, and in principle, they may benefit the adaptive models as well. To focus on the adaptive learning setting, we leave investigations on this direction for future work.

Model	Params ($\times 10^6$)	Inference MADDs $\times 10^6$	Accuracy (Top-1)
Hydra-Res-d1	1.28	52	65.81
Hydra-Res-d2	2.86	118	71.24
Hydra-Res-d3	4.43	184	72.30
Hydra-Res-d4	6.01	251	73.35
Hydra-Res-d5	7.59	317	73.84
Hydra-Res-d6	9.17	383	74.29
Hydra-Res-d7	10.74	449	74.71
Hydra-Res-d9	13.90	581	75.26
MSDNet-Exit1	0.3	6.86	64.1
MSDNet-Exit2	0.65	14.35	67.46
MSDNet-Exit3	1.11	27.29	70.34
MSDNet-Exit4	1.73	48.45	72.38
MSDNet-Exit5	2.38	76.43	73.06
MSDNet-Exit6	3.05	108.9	73.81
MSDNet-Exit7	4.0	137.3	73.89
Ours-Exit1	0.3	6.86	64.00
Ours-Exit2	0.65	14.35	68.41
Ours-Exit3	1.11	27.29	71.86
Ours-Exit4	1.73	48.45	73.50
Ours-Exit5	2.38	76.43	74.46
Ours-Exit6	3.05	108.9	75.39
Ours-Exit7	4.0	137.3	75.96

Table 1. Classification accuracy of individual classifiers on CIFAR-100.

4.1. Evaluation on CIFAR

Baselines. On the two CIFAR datasets, following the MSDNet we train networks with three-scale features, *i.e.*, 32×32, 16×16, 8×8. To better evaluate our approaches, on CIFAR-10, the MSDNets are with $\{6, 8\}$ exits and the depths are $\{21, 36\}$. On CIFAR-100, we train MSDNets with $\{4, 5, 6, 8\}$ exits and the depths are $\{10, 15, 21, 36\}$ respectively. In this setting, we also compare with the original baseline MSDNet, four ResNets with different depths, and six DensNets with varying depths.

Results on CIFAR. The evaluation results on CIFAR-10 and CIFAR-100 are shown in Figure 2 and Figure 3, respectively. The results for baseline MSDNet are plotted by red curves (corresponding to three MSDNets with different sizes), and the results for our proposed training strategy are shown by black curves. As shown in the figures, MSDNet trained with our proposed training strategy clearly achieves better accuracy than the baseline MSDNet under the same time budget. Moreover, the improvement increases as we have more time budgets. For example, with 1×10^8 FLOPS, our training strategy improves MSDNet baseline by more than 0.5% in terms of top-1 accuracy. This demonstrates that our training strategy can facilitate training of deeper

Method			Accuracy @TOP1				
GE	ISC	OFA	E-1	E-2	E-3	E-4	E-5
-	-	-	60.09	63.73	67.89	70.48	71.81
✓			60.35	64.38	68.72	70.65	71.94
	✓		60.19	64.72	68.07	70.94	73.28
		✓	60.39	64.20	68.10	70.65	71.85
✓	✓	✓	**60.78**	**65.54**	**69.98**	**72.27**	**73.45**

Table 2. Accuracy at different exits on CIFAR-100. The results produced by integrating **GE, ISC, OFA**.

Method			Accuracy @TOP1				
GE	ISC	OFA	E-1	E-2	E-3	E-4	E-5
-	-	-	56.64	65.14	68.42	69.77	71.34
✓			57.08	65.29	69.08	70.55	72.14
	✓		57.03	66.2	69.73	71.15	71.65
		✓	57.15	65.77	68.87	70.23	71.39
✓	✓	✓	**57.28**	**66.22**	**70.24**	**71.71**	**72.43**

Table 3. Accuracy at different exits on ImageNet. The results produced by integrating **GE, ISC, OFA**.

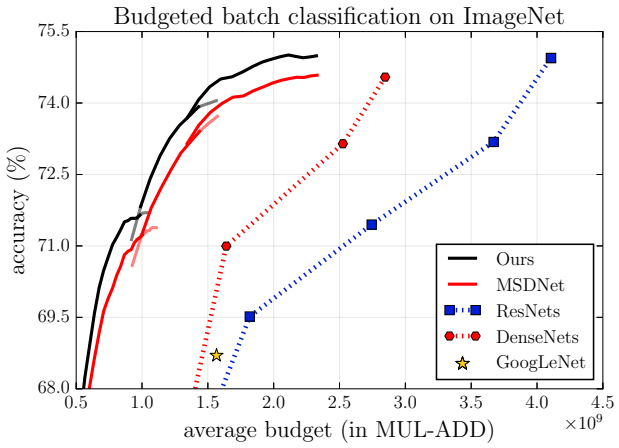


Figure 5. Top-1 accuracy of *budgeted batch classification* as a function of average computational budget per image on ImageNet.

layers. Besides, compared with ResNets and DenseNets, adaptive inference based approaches (MSDNet with and without our training strategies) perform significantly better with the same amount of computation. For instance, to achieve the same accuracy on CIFAR-100 (Figure 3), our approach requires half the amount of computation as DenseNet and 1/3 of the computation as ResNet.

In Table 1, we report the classification accuracy of all the individual classifiers of our model, and compare it with MSDNet as well as the recently proposed HydraNets [34]. We can observe that the results of each individual classifier of our network are competitive with state-of-the-art models.

4.2. Evaluation on ImageNet

Baselines. On ImageNet, we use the four-scale MSDNet, *i.e.*, 56×56, 28×28, 14×14, 7×7. Each of the MSDNet has five classifiers inserted at different depths. Specifically, the i^{th} classifier is attached in the $(t \times i + 3)^{th}$ layer where $i \in \{1, \ldots, 5\}$, $t \in \{4, 6, 7\}$ is the step for each network block. We also compare with other competitive approaches, ResNet [10], DenseNet [16] and GoogleNet [32].

Results on ImageNet. The results on ImageNet are shown in Figure 5. One can observe that our method with dynamic evaluation built on the top of MSDNets consistently surpass the baseline network. With 1×10^8 FLOPS computation budget, we improve the baseline method by around 0.5% in terms of top-1 accuracy. Again, for the same MSDNet architecture, our method improves the baseline by a larger margin as more allowed computational budgets. This further verifies that our proposed method is effective for deeper adaptive networks. Moreover, with the same FLOPs, our method is more accurate and efficient than the models of ResNets and DenseNets. For instance, with around 1×10^8 FLOPs, our approach outperforms the ResNet and DensNet by more than 6%.

4.3. Ablation Study

To investigate the effectiveness of the individual modules of the proposed approach, i.e., **GE, ISC** and **OFA**, we conduct ablative analysis on the CIFAR-100. We set the baseline MSDNets on CIFAR-100 with three scales and five classifiers and on ImageNet we use MSDNet with four scales and five classifiers. Quantitive results are shown in Table 2 and Table 3. Our full model consistently improves the accuracy on both CIFAR-100 and ImageNet. For stages two to four, our full model improves the baseline method by more than 1% on CIFAR-100 dataset, and surpasses the baseline by more than 0.5%. All the strategies consistently improve the performance on both datasets.

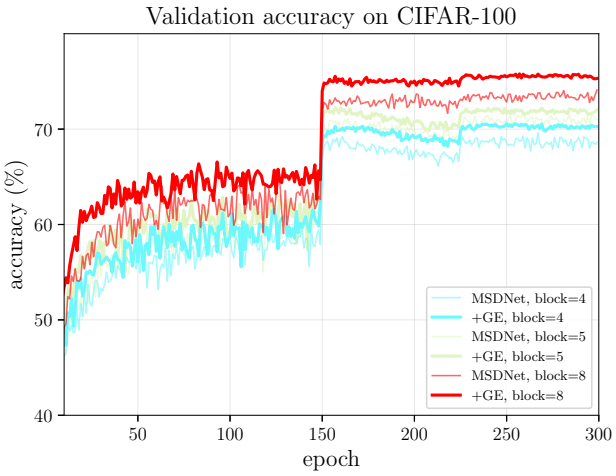


Figure 6. **Validation accuracy** on CIFAR-100 at different epochs. We plot the results of three different depth of networks with 4, 6, 8 exits respectively. The losses of the models trained with **GE** are consistently lower than the baseline models.

Gradient Equilibrium. To further evaluate the effectiveness of *Gradient Equilibrium*, we compare different MSDNet architecture trained with *Gradient Equilibrium* and with the baseline MSDNet. Validation accuracy in both settings are shown in Figure 6. With *Gradient Equilibrium*, the validation accuracy is consistently higher and the training procedure is more stable than the baseline algorithm for all the compared architectures. This demonstrates *Gradient Equilibrium* can help stabilize the training process and at the same time improve the accuracy which is also shown in Table 2(line 1 vs line 2) and Table 3.

Inline Subnetwork Collaboration. Inline Subnetwork Collaboration (ISC) can consistently improve the performance as shown in Table 2 (line 1 vs line 3) on CIFAR-100 dataset. Deeper layers typically benefit more from ISC, *e.g.*, exit 5 (E-5) improves by more than 1.4% in Top-1 accuracy. This might be explained by that deeper layer can acquire more information from other classifiers with our inline subnetwork collaboration module. In Figure 7, we plot the confidence rank of all the validation samples before and after applying ISC. One can observe a clear trend that the red dots, which corresponding to the results with ISC, is more concentrated than the blue dots, which corresponding to the results without ISC. This demonstrates that with ISC, the consistency between the rank of samples at different exits (exit-1 and exit-2 here) has significantly increased, and partially explains the effectiveness of ISC.

One-for-all Knowledge Distillation. Quantitative improvements with the OFA strategy are shown in Table. 2 (line 1 vs line 4). Lower layers tend to gain larger improvement, showing that they benefited from the supervision from the deepest classifier. These results further show that knowledge distillation is indeed effective for the adaptive network to exploit its own prediction.

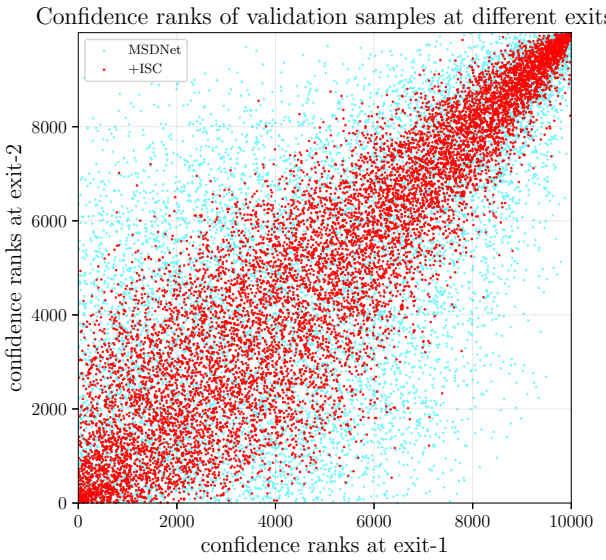


Figure 7. The distribution of *confidence scores* at different exits. In order to investigate the ISC's effects to different classifiers, we compare the ranks of each sample on different exits. It's obvious that the distribution is more consistent with ISC, i.e. more closer to the identity mapping, which indicates ISC helps the collaboration among exits.

5. Conclusion

In this paper, we have presented three techniques to improve the training of adaptive neural network with multiple exits. On one hand, a Gradient Equilibrium (GE) approach is proposed to stabilize the training procedure and resolve the conflict of learning objectives of different classifiers. One the other hand, we have introduced two techniques to strengthen collaboration among classifiers. Although being simple, the proposed techniques have shown its effectiveness on a number of image recognition datasets, and significantly improved the efficiency of the recently proposed MSDNet. Future research may focus on extending our results to other types of adaptive networks, e.g., spatially adaptive networks [6], or applying them to other computer vision tasks, such as object detection, semantic segmentation and image generation.

Acknowledgements. Gao Huang is supported in part by Beijing Academy of Artificial Intelligence under grant BAAI2019QN0106. Hao Li is supported in part by Tsinghua University Initiative Scientific Research Program and Tsinghua Academic Fund for Undergraduate Overseas Studies. We thank Danlu Chen for helpful discussions.

References

[1] Jimmy Ba and Rich Caruana. Do deep nets really need to be deep? In *NIPS*, 2014. 2

[2] Tolga Bolukbasi, Joseph Wang, Ofer Dekel, and Venkatesh Saligrama. Adaptive neural networks for fast test-time prediction. In *ICML*, 2017. 1, 2

[3] Cristian Bucilua, Rich Caruana, and Alexandru Niculescu-Mizil. Model compression. In *ACM SIGKDD*, 2006. 2

[4] Wenlin Chen, James Wilson, Stephen Tyree, Kilian Q Weinberger, and Yixin Chen. Compressing convolutional neural networks in the frequency domain. In *ACM SIGKDD*, 2016. 1, 2

[5] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *CVPR*, 2009. 5

[6] Michael Figurnov, Maxwell D Collins, Yukun Zhu, Li Zhang, Jonathan Huang, Dmitry Vetrov, and Ruslan Salakhutdinov. Spatially adaptive computation time for residual networks. *arXiv preprint arXiv:1612.02297*, 2016. 1, 2, 8

[7] Alex Graves. Adaptive computation time for recurrent neural networks. *arXiv preprint arXiv:1603.08983*, 2016. 1

[8] Song Han, Huizi Mao, and William J Dally. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. In *ICLR*, 2016. 1, 2

[9] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *ICCV*, 2017. 1

[10] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016. 1, 2, 5, 7

[11] Yihui He, Ji Lin, Zhijian Liu, Hanrui Wang, Li-Jia Li, and Song Han. Amc: Automl for model compression and acceleration on mobile devices. In *ECCV*, 2018. 2

[12] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. In *NIPS Deep Learning Workshop*, 2014. 2, 4

[13] Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017. 2, 6

[14] 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 *ICLR*, 2018. 1, 2, 3, 5, 6

[15] Gao Huang, Shichen Liu, Laurens Van der Maaten, and Kilian Q Weinberger. Condensenet: An efficient densenet using learned group convolutions. In *CVPR*, 2018. 1, 2

[16] Gao Huang, Zhuang Liu, Kilian Q Weinberger, and Laurens van der Maaten. Densely connected convolutional networks. In *CVPR*, 2017. 1, 5, 7

[17] Itay Hubara, Matthieu Courbariaux, Daniel Soudry, Ran El-Yaniv, and Yoshua Bengio. Binarized neural networks. In *NIPS*, 2016. 1, 2

[18] Di Kang, Debarun Dhar, and Antoni Chan. Incorporating side information by adaptive convolution. In *NIPS*, 2017. 2

[19] Shu Kong and Charless Fowlkes. Pixel-wise attentional gating for parsimonious pixel labeling. *arXiv preprint arXiv:1805.01556*, 2018. 2

[20] Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. *Tech Report*, 2009. 5

[21] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*, 2012. 1

[22] Xu Lan, Xiatian Zhu, and Shaogang Gong. Knowledge distillation by on-the-fly native ensemble. *arXiv preprint arXiv:1806.04606*, 2018. 2

[23] Hao Li, Asim Kadav, Igor Durdanovic, Hanan Samet, and Hans Peter Graf. Pruning filters for efficient convnets. In *ICLR*, 2017. 1, 2

[24] Zhichao Li, Yi Yang, Xiao Liu, Feng Zhou, Shilei Wen, and Wei Xu. Dynamic computational time for visual attention. In *ICCV*, 2017. 2

[25] Ji Lin, Yongming Rao, Jiwen Lu, and Jie Zhou. Runtime neural pruning. In *NIPS*, 2017. 2

[26] Zhuang Liu, Jianguo Li, Zhiqiang Shen, Gao Huang, Shoumeng Yan, and Changshui Zhang. Learning efficient convolutional networks through network slimming. In *ICCV*, 2017. 1

[27] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *CVPR*, 2015. 1

[28] Ningning Ma, Xiangyu Zhang, Hai-Tao Zheng, and Jian Sun. Shufflenet v2: Practical guidelines for efficient cnn architecture design. In *ECCV*, 2018. 1, 2

[29] Lane McIntosh, Niru Maheswaranathan, David Sussillo, and Jonathon Shlens. Recurrent segmentation for variable computational budgets. In *CVPR Workshops*, 2018. 2

[30] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *CVPR*, 2018. 1, 2

[31] Guocong Song and Wei Chai. Collaborative learning for deep neural networks. In *NIPS*, 2018. 1

[32] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *CVPR*, 2015. 1, 7

[33] Surat Teerapittayanon, Bradley McDanel, and HT Kung. Branchynet: Fast inference via early exiting from deep neural networks. In *ICPR*, 2016. 1

[34] Ravi Teja Mullapudi, William R Mark, Noam Shazeer, and Kayvon Fatahalian. Hydranets: Specialized dynamic architectures for efficient inference. In *CVPR*, 2018. 1, 7

[35] Andreas Veit and Serge Belongie. Convolutional networks with adaptive inference graphs. In *ECCV*, 2018. 1, 2

[36] Xin Wang, Fisher Yu, Zi-Yi Dou, Trevor Darrell, and Joseph E Gonzalez. Skipnet: Learning dynamic routing in convolutional networks. In *ECCV*, 2018. 2

[37] Zuxuan Wu, Tushar Nagarajan, Abhishek Kumar, Steven Rennie, Larry S Davis, Kristen Grauman, and Rogerio Feris. Blockdrop: Dynamic inference paths in residual networks. In *CVPR*, 2018. 2

[38] Chris Ying and Katerina Fragkiadaki. Depth-adaptive computational policies for efficient visual tracking. In *EMMCVPR*, 2017. 2

[39] Xiangyu Zhang, Xinyu Zhou, Mengxiao Lin, and Jian Sun. Shufflenet: An extremely efficient convolutional neural network for mobile devices. In *CVPR*, 2018. 2, 6

[40] Barret Zoph, Vijay Vasudevan, Jonathon Shlens, and Quoc V Le. Learning transferable architectures for scalable image recognition. In *CVPR*, 2018. 1, 2, 6