Harmonized Dense Knowledge Distillation Training for Multi-Exit Architectures

Xinglu Wang, Yingming Li [*]
College of Information Science & Electronic Engineering, Zhejiang University, China
{xingluwang,yingming}@zju.edu.cn

Abstract

Multi-exit architectures, in which a sequence of intermediate classifiers are introduced at different depths of the feature layers, perform adaptive computation by early exiting "easy" samples to speed up the inference. In this paper, a novel Harmonized Dense Knowledge Distillation (HDKD) training method for multi-exit architecture is designed to encourage each exit to flexibly learn from all its later exits. In particular, a general dense knowledge distillation training objective is proposed to incorporate all possible beneficial supervision information for multi-exit learning, where a harmonized weighting scheme is designed for the multi-objective optimization problem consisting of multi-exit classification loss and dense distillation loss. A bilevel optimization algorithm is introduced for alternatively updating the weights of multiple objectives and the multi-exit network parameters. Specifically, the loss weighting parameters are optimized with respect to its performance on validation set by gradient descent. Experiments on CIFAR100 and ImageNet show that the HDKD strategy harmoniously improves the performance of the state-of-the-art multi-exit neural networks. Moreover, this method does not require within architecture modifications and can be effectively combined with other previously-proposed training techniques and further boosts the performance.

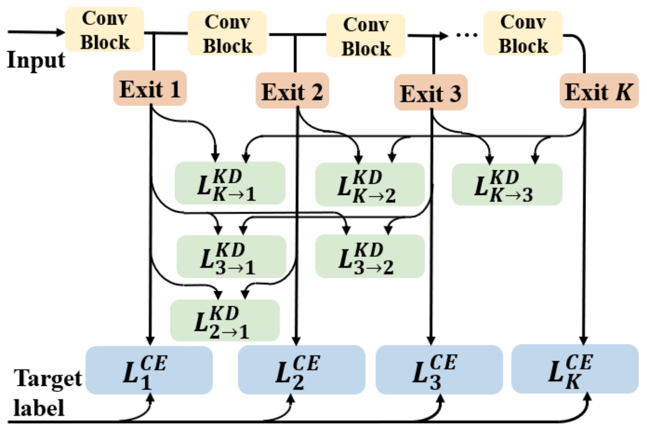

Figure 1: Dense knowledge distillation training for multi-exit architecture. Given an input, the prediction of different exits will be sequentially generated. Classification loss at exit i is denoted as L_i^{CE}. Dense distillation loss from exit i to exit j, denoted as L_{ij}^{KD}, allows each exit to learn from all its later exits.

Introduction

Deep learning methods, especially convolutional neural networks (CNNs) have achieved remarkable success on a variety of computer vision tasks, such as image classification and object detection. However, these extraordinary achievements usually rely on very deep models accompanied by high computational demands, which prevents them from being deployed in resource-constrained platforms like mobile devices.

Recently, there has been a rising interest in accelerating CNNs at inference time. Common methods include direct network pruning (Han et al. 2015; Zhu and Gupta 2017), knowledge distillation (Hinton, Vinyals, and Dean 2015; Romero et al. 2014), weight quantization (Courbariaux and Bengio 2016; Rastegari et al. 2016), novel architecture designs (Howard et al. 2017), and adaptive inference (Liu and Deng 2017; Wu et al. 2018). Among them, the adaptive inference has attracted much attention as it does not modify network structure and can be applied in tandem with other approaches to further reduce the inference time (Lin et al. 2017; Wang et al. 2018a). Given the fact that natural data is usually composed of easy samples and difficult samples, adaptive inference aims at achieving efficient resource allocation without loss of accuracy, by positing that during the inference stage easy samples do not require the full power of a massive deep neural network.

Multi-exit architecture has emerged as a representative approach for adaptive inference (Bolukbasi et al. 2017; Li et al. 2019). It implements adaptive computation by early exiting "easy" samples to speed up the inference. In this architecture, early exits are introduced at different depths of the feature layers. At these early exits, a sequence of intermediate classifiers of increasing complexity are incorporated. As shown in the Fig. 1, the representations and computations of the earlier layers are reused by the later layers. The prediction for an input image is progressively updated throughout

[*]Corresponding author

the network.

Multi-exit architecture learning is inherently a multi-objective problem. Different objectives from different classifiers may interfere, for example when the intermediate classifier loss facilitates the early features to be optimized for the short-term and not for the final exits. Multi-scale dense network (MSDNet) alleviates the negative effect of early classifiers via dense connections (Huang et al. 2017a). Building upon MSDNet, recent studies incorporate distillation loss to improve the training procedure by encouraging early exits to mimic the output distributions of the last exit. The motivation is that the last exit has the maximum network capacity and should be more accurate than the ones from the earlier exits.

Though the distillation-based multi-exit learning methods achieve better performance, there are still some limitations. On the one hand, they employ a naive weighted sum of losses and the loss weights are uniform or manually tuned, where the tradeoff between the multi-exit classification loss and distillation loss is not well considered. Since different objectives in a joint optimization might be competing or even conflicting (Yu et al. 2020), such a simple weighting scheme would affect the joint learning of multi-exit classification and knowledge distillation. On the other hand, the existing methods mainly employ the *last* exit as a teacher model and distill knowledge into earlier exits. However, to fully release the potential of knowledge distillation, it is necessary to adaptively learn from *all later* exits for each exit.

To further improve the generalization performance of multi-exit architecture learning, in this paper we investigate the multi-exit learning problem with a new perspective of considering dense knowledge distillation and the harmonized weighting scheme simultaneously. A new framework for training multi-exit architectures with Harmonized Dense Knowledge Distillation (HDKD) is introduced to fully exploit the beneficial supervision information. As shown in the Fig. 1, each exit learns from all its later exits through dense knowledge distillation to improve the classification accuracy. Further, a harmonized weight learning method is employed to obtain the best trade-off between the multiple objectives of multi-exit classification and dense distillation. Instead of searching over a discrete set of candidate weights, HDKD learns to adaptively assign weights to the different objectives of different exits based on their gradient directions. In particular, similar to meta-learning, it adopts a gradient descent step on the current loss weights with respect to its performance on the validation set. Validation is performed at every training iteration to dynamically update the loss weights of the current batch. The overall updating of the multiple objective weights and the model parameters in the network are solved with a bilevel optimization procedure.

Extensive evaluations are conducted to investigate the performances of training multi-exit architectures with harmonized dense knowledge distillation. The experimental results on two public image datasets show that HDKD consistently improves the performance of the state-of-the-art multi-exit neural network.

Related Work

As an important technique for accelerating CNNs at inference time, adaptive inference strategically allocates less computation on “easy” samples and more resource to the “hard” samples (Wang et al. 2018b). Various adaptive inference schemes are explored: (Bolukbasi et al. 2017) learn an adaptive decision function by reinforcement learning to determine the stage examples exit at. Gating functions are designed upon ResNet architectures to dynamically choose layers for efficient inference (He et al. 2016; Veit and Belongie 2018; Wang et al. 2018a). An RNN architecture is proposed by (McIntosh et al. 2018) for dynamically determining the number of steps according to the time budget.

MSDNet leverages multi-scale mechanism and has attracted much attention for its simplicity and performance (Huang et al. 2017a; Ke et al. 2018; You et al. 2018). Based on MSDNet, follow-up methods improve from different perspectives: (Li et al. 2019) introduces a gradient equilibrium mechanism that keeps a fast learning rate for each classifier while reducing variance of the shared parameter gradients. (Wang and Li 2020) propose a gradient deconfliction training algorithm to mitigate the interference between exits. (Zhang, Ren, and Urtasun 2018) design a Graph HyperNetwork to automatically find the best task-specific topology for multi-exit networks.

Among the existing methods, the closest work is (Phuong and Lampert 2019). It focuses on training with distillation loss by encouraging early exits to imitate the last exit. Our method is distinct in dense knowledge distillation that allows each exit to learn from all the later exits and adaptive weight learning that resolves the conflicts among multi-exit classification losses and dense distillation losses. Meanwhile, our method is architecture agnostic and can be directly applied to existing multi-exit architectures (Teerapittayanon, McDanel, and Kung 2016; Luan et al. 2019a). As an alternative way resolving conflicts, (Passalis et al. 2020) simultaneously trains all the exit layers with the base network frozen.

On the other hand, our method leverages an adaptively weighted loss rather than a fixed average loss and can be considered as an instantiation of meta-learning. In the sense of taking validation loss as a meta-objective and applying one-step update in each iteration, our work acts like the optimization-based meta-learning, e.g., MAML and its variants (Finn, Abbeel, and Levine 2017; Nichol, Achiam, and Schulman 2018; Sun et al. 2019; Rajeswaran et al. 2019). However, MAML aims at learning a transferable weight initialization for fast adaptation on few-shot learning task (Ravi and Larochelle 2016; Lorraine and Duvenaud 2018). The common goal is to adapt with only a few data points. In contrast, the proposed HDKD aims at facilitating multi-exit learning with dense knowledge distillation during regular training and is generally applicable to any large-scale classification task with thousands of classes.

Methodology

In this section, we first analyze the necessary background for multi-exit network learning. Then dense knowledge distillation is introduced into the training process of multi-exit learning. Further, we propose a harmonized weighting scheme for the multiple objective optimization problem of training with dense distillation loss. In particular, the loss weights and the multi-exit network parameters are alternatively updated within a bilevel optimization framework.

Multi-Exit Network

By attaching early exits at different depths, multi-exit architecture can be viewed as a dynamic version of the conventional neural network. As shown in Fig. 1, given an input image x and its label y, then a multi-exit architecture with K exits can be represented by a multi-prediction function:

$$f(x;\theta) = [f_1(x;\theta_1), \quad \dots \quad ,f_K(x;\theta_K)], \tag{1}$$

where f_i indicates the sub-network function learned by the i-th classifier with $\theta_i, i = 1, \dots, K$ as its accompanied parameters. The overall parameters of the network are encapsulated as θ.

During the training stage, all the subnetworks are jointly optimized with a cumulative loss function. Following (Huang et al. 2017a), the weights for all loss functions are simply set to 1 and cross-entropy (CE) loss is employed for all loss functions. Denote the loss function of the i-th exit. classifier $L_i^{CE}(\theta_i)$ as $L_i(y, f_i(x;\theta_i))$, the optimal parameter is optimized via $\theta^* = \arg\min_\theta \sum_{i=1}^{K} L_i^{CE}(\theta_i)$

At the inference stage, the multi-exit architecture would be beneficial in the two following settings with computational constraints: *budgeted batch classification mode*, where a set of examples share a fixed computational budget and the budget can be allocated unevenly across "easy" and "hard" samples; and *anytime prediction mode*, where the system can output a prediction at any given exit at anytime time. We follow the setup proposed by (Huang et al. 2017a). In experiments, for the batched budget classification, the confidence threshold for each exit classifier is first determined according to the computational cost of each corresponding stage on a hold-out subset. Then, the input examples would traverse the network and exit after the classifier f_i when its prediction confidence surpasses the predetermined threshold t_i. For anytime classification, the input propagates through the network until the budget is exhausted or users interrupt the computation. The latest prediction would be output.

Training with Harmonized Dense Knowledge Distillation

Knowledge distillation (KD) is a representative compression technique, which aims to transfer the knowledge of a large teacher network to a lightweight student network with fewer layers. For multi-exit learning, since the later exits have more network capacity and should be more accurate than the early exits, KD is a natural way to boost the performance of shallow exits by encouraging them to imitate the deep ones. As the student model, the early-exit is required to learn the soft-labels provided by the teacher model, the corresponding later exits. Selecting exit-k_2 as teacher and exit-k_1 as student ($k_2 > k_1$), the distance loss between the teacher's prediction and the student's prediction is defined as:

$$L_{k_2 \to k_1}^{KD} = -T^2 \, \mathrm{CE}(s_{k_1}, s_{k_2}) = -T^2 \sum_i s_{k_1}^i \log s_{k_2}^i \tag{2}$$

where s_k^i denotes i-th scalar of soft labels vector in the k-th exit, which is obtained by scaling the logits predicted by the exit with a temperature T and further normalizing with softmax function, and T^2 is multiplied to compensate the magnitudes of the gradients scaled by the soft targets (Hinton, Vinyals, and Dean 2015).

By employing the last exit as a teacher model distilling knowledge into earlier exits, the objective of the existing distillation based multi-exit learning (Phuong and Lampert 2019; Zhang et al. 2019; Luan et al. 2019b) is as follows:

$$\theta^* = \arg\min_\theta \sum_{i=1}^{K} L_i^{CE}(\theta_i) + \lambda \sum_{j=1}^{K-1} L_{K \to j}^{KD}(\theta_j), \tag{3}$$

where λ is a manually tuned hyper-parameter for the balance of classification and distillation loss. Though this way improves the training procedure and achieves better performance, it fails to exploit all potentially beneficial collaboration between exits. To fully release the potential abilities of knowledge distillation, we propose to train the multi-exit architecture with *dense* knowledge distillation. In particular, each exit learns from all its later exits and dense distillation is incorporated to improve the classification accuracy. The overall objective is to minimize the following integrated loss on the training set:

$$\theta^* = \arg\min_\theta \sum_{i=1}^{K} L_i^{CE}(\theta_i) + \lambda \sum_{i=2}^{K} \sum_{j=1}^{i-1} L_{i \to j}^{KD}(\theta_j), \tag{4}$$

where dense distillation loss is considered to fully exploit the beneficial supervision information.

However, the above setup in Eq. 4 does not fully consider the tradeoff between the multiple objectives. On the one hand, each distillation loss for any exit is weighted equally and thus the importance of knowledge from different exits are not reflected. On the other hand, different loss objectives are usually competing or even conflicting (Yu et al. 2020), which also influence the overall performance. To overcome these limitations, Harmonized Dense Knowledge Distillation (HDKD) training for multi-exit architecture is introduced to learn an appropriate weighting scheme for the multi-objective optimization. Instead of searching over a discrete set of candidate weights, HDKD learns to adaptively assign weights to the different objectives of different exits based on their gradient directions, where we minimize a weighted loss:

$$\theta^*(\gamma) = \arg\min_\theta \sum_{i=1}^{K} \gamma_i L_i^{CE}(\theta_i) + \sum_{i=2}^{K} \sum_{j=1}^{i-1} \gamma_{ij} L_{i \to j}^{KD}(\theta_j), \tag{5}$$

where γ_i controls the strength of cross-entropy loss at exit-i, γ_{ij} denotes the weight of distillation loss from exit-i to exit-j, and γ encapsulates all γ_i and γ_{ij}. Note that γ can be understood as a set of learnable training hyper-parameter, and the optimal value for γ is based on its validation performance:

$$\gamma^* = \arg\min_{\gamma} L_{\text{val}}\left(\theta^*(\gamma)\right) \tag{6}$$

Bilevel-Optimization With L_{train} and L_{val} denoting the training and validation loss respectively, they are determined not only by the loss weights γ, but also the model parameters θ in the network. The goal for training with harmonized dense distillation is to find γ^* that minimizes the validation loss $L_{val}(\theta^*(\gamma))$, where the parameters associated with the multi-exit architecture are obtained by minimizing the training loss $\theta^* = \arg\min_\theta L_{\text{train}}(\theta, \gamma^*)$.

Consequently, a bilevel optimization problem is formed with γ as the upper-level variable and θ as the lower-level variable:

$$\begin{array}{ll} \min_\gamma & L_{\text{val}}\left(\theta^*(\gamma)\right) \\ \text{s.t.} & \theta^*(\gamma) = \arg\min_\theta L_{\text{train}}(\theta, \gamma) \end{array} \tag{7}$$

where the L_{val} is optimized to find the optimal configuration γ in the upper-level and the L_{train} is minimized to obtain the best model parameters $\theta^*(\gamma)$ given a certain γ in the lower-level.

Directly solving the optimization in Eq. 7 is prohibitive, as evaluating the gradient of γ exactly involves another expensive inner optimization problem. We therefore adopt a one-step update approximation as follows:

$$\begin{aligned} &\nabla_\gamma L_{\text{val}}\left(\theta^*(\gamma)\right) \\ \approx&\nabla_\gamma L_{\text{val}}\left(\theta - \xi \nabla_\theta L_{\text{train}}(\theta, \gamma)\right), \end{aligned} \tag{8}$$

where θ denotes the current weights maintained by the algorithm, and ξ is the learning rate for the inner one-step optimization. The idea is to approximate $\theta^*(\gamma)$ by adapting θ using only one single training step, without training the model given γ until convergence and completely solving the inner optimization in Eq. 7.

As shown in Alg. 1, the bilevel optimization is performed with an iterative procedure. In particular, the calculating of gradient $\nabla_\gamma L_{val}(\theta^t - \xi \nabla_{\theta^t} L_{train}(\theta^t, \gamma))$ in the upper level involves a second-order gradient. It can be implemented by the modern automatic differentiation framework, which is capable of computing the Jacobian-vector products in a forward mode, so as to compute the Hessian-vector products, i.e., the gradient of the loss weights, in later backward mode. It is also worthy to note that the model parameters do not directly touch validation data and would not over-fit on it.

Balanced Validation Module It is a simple and direct option to measure the performance of the current network on a validation set adopting the sum of cross-entropy loss, $\sum_k L_k^{CE}(\theta^*(\gamma))$. However, the losses of early exits tend to be larger than the later ones and might be overemphasized, thereby influencing the optimization of later exits. In other words, adopting the absolute cross-entropy loss is not enough for a balanced optimization.

To mitigate this influence, the relative change of exit-wise loss is also considered. In specific, we propose a ratio constraint, i.e., the ratio of the current cross-entropy loss and

Algorithm 1 Harmonized Dense Knowledge Distillation Training Procedure for Multi-exit Learning

```
Require: Normal update steps $T$, learning rate $\xi$ for $\theta$ and learning rate $\epsilon$ for $\gamma$.
  while not converge do
    for $t = 0 \cdots T$ do
      Sample $X_{train}$ and $X_{val}$, calculate the validation loss
          $L_{val} = L_{val}(\theta^t - \xi \nabla_{\theta^t} L_{train}(\theta^t, \gamma^t))$
      Calculate $\nabla_\gamma L_{val}$ and update the loss weights:
          $\gamma^{t+1} = \gamma^t - \epsilon \nabla_{\gamma^t} L_{val}$
    end for
    for $t = 0 \cdots T$ do
      Sample $X_{train}$, calculate $L_{train}(\theta^t, \gamma^t)$
      Calculate $\nabla_{\theta^t} L_{train}(\theta^t, \gamma^t)$ and update the model parameters:
          $\theta^{t+1} = \theta^t - \xi \nabla_{\theta^t} L_{train}(\theta^t, \gamma^t)$
    end for
  end while
  return The converged model.
```

its estimation of expected loss $L_k^{CE}(\theta^*(\gamma))/\mathbb{E}(L_k^{CE})$ should smaller than 1, since the current cross-entropy loss of each exit is usually expected to be less than its estimation and a proper penalty should be imposed when this assumption is violated. The ratio constraint term would be integrated to validation loss as a regularization term, and for simplicity, we call it as *ratio regularizer*.

The estimation of expected exit-k loss $\mathbb{E}(L_k^{CE})$ is computed with a exponential moving average in the following way:

$$\mathbb{E}(L_k^{CE}) \leftarrow m\mathbb{E}(L_k^{CE}) + (1-m)L_k^{CE} \tag{9}$$

It is a common practice to approximate expectation by fixing momentum as $m = 0.9$ (Ioffe and Szegedy 2015; Li, Liu, and Wang 2019).

Further, we combine the absolute cross-entropy loss with the ratio regularizer to construct a more balanced validation loss for meta gradient updating. The joint loss function is written as follows:

$$\begin{aligned} L_{\text{val}}(\theta^*(\gamma)) = \alpha \sum_{k=1}^{K} L_k^{CE}(\theta^*(\gamma)) \\ +(1-\alpha)\sum_{k=1}^{K} f\left(\frac{L_k^{CE}(\theta^*(\gamma))}{\mathbb{E}(L_k^{CE})} - 1\right), \end{aligned} \tag{10}$$

where α is a balancing parameter, the softplus function $f(x) = \ln(1 + \exp(x))$ is the soft version of hinge loss. The second term in Eq. 10 is the ratio regularizer pursuing balanced relative change of cross-entropy loss. Even when the ratio constraint is not violated, the f would encourage a further decrease of the corresponding exit loss.

	flops	ImageNet(100)							ImageNet(300)						
	(G)	MSD Net	Impr	DBT	Naive DKD	HMSD Net	HDBT	HDKD	MSD Net	Impr	DBT	Naive DKD	HMSD Net	HDBT	HDKD
Exit-1	0.34	59.07	59.74	65.8	67.17	60.12	66.97	**68.3**	72.45	73.27	76.01	77.49	72.62	**78.9**	78.72
Exit-2	0.69	67.16	68.27	70.82	71.35	68.39	71.24	**72.24**	79.95	81.28	81.84	82.21	80.54	**83**	82.58
Exit-3	1.01	70.25	71.12	71.89	72.23	70.59	73.94	**74.02**	83.12	84.33	83.41	83.69	83.31	84.61	**85.13**
Exit-4	1.25	70.9	72.06	71.55	72.04	71.72	74.06	**74.09**	83.61	85.94	83.63	83.87	84.92	84.58	**85.87**
Exit-5	1.36	71.51	72.87	71.57	71.96	72.11	74.5	**74.85**	84.66	85.96	84.01	84.33	85.08	85.06	**86.37**
	flops	ImageNet(500)							ImageNet(full)						
	(G)	MSD Net	Impr	DBT	Naive DKD	HMSD Net	HDBT	HDKD	MSD Net	Impr	DBT	Naive DKD	HMSD Net	HDBT	HDKD
Exit-1	0.34	74.89	75.74	78.16	78.16	75.16	**81.17**	80.70	79.25	80.15	80.80	82.33	78.95	**83.06**	82.44
Exit-2	0.69	82.76	84.13	83.87	84.03	83.49	**85.07**	84.48	86.46	87.89	86.92	87.39	87.43	87.12	**87.91**
Exit-3	1.01	85.47	85.74	85.72	85.12	86.83	**86.81**	86.47	89.15	90.52	88.82	88.85	89.49	90.85	**91.47**
Exit-4	1.25	85.92	86.31	86.3	85.35	87.25	87.26	**87.42**	89.83	91.33	89.15	89.17	91.22	**91.9**	91.79
Exit-5	1.36	86.92	87.25	86.85	85.81	**88.56**	87.85	87.96	90.75	91.74	89.73	89.67	92.43	92.04	**92.69**

Table 1: Top-5 classification accuracy for each individual classifier in 5-exits 23-layers MSDNet. We compare the models trained by the exit-wise loss v.s. distillation-based training methods v.s. the proposed Harmonized weighting versions on ImageNet with 100, 300, 500, and all images per class. Bold values indicate statistically significant performance.

Experiments

To demonstrate the effectiveness of the proposed training approach, we conduct extensive experiments on two representative image classification datasets, CIFAR100 (Krizhevsky, Nair, and Hinton 2009) and ILSVRC 2012 ImageNet (Russakovsky et al. 2015). All of the experiments are built upon the MSDNet (Huang et al. 2017a). A fair comparison is performed by implementing the state-of-the-art methods with the same hyper-parameter setup in Tensorflow (Abadi et al. 2016) framework.

Experimental Setup

Datasets CIFAR100 dataset contains RGB images of size 32×32, with 50,000 images of 100 classes for training and 10, 000 images for testing. Following (Huang et al. 2017b), we hold out 5,000 training images as a validation set for searching the confidence threshold in budgeted batch classification and supporting the loss weights updating for the HDKD method. This validation set could be safely reused since the model trained by HDKD does not touch the ground-truth label of the validation set and will not overfit on it. Further, standard data augmentation schemes (He et al. 2016) are used for training.

ImageNet dataset contains 1,000 classes, with 1.2 million training images and 50,000 for testing. 50,000 images in the training set are held out and serve as the validation set. For data augmentation, we also follow the practice in (He et al. 2016; Huang et al. 2017b).

Following the setup of (Phuong and Lampert 2019), we conduct experiments from low-data (using only subsets of the available data) to the full-dataset setting. By CIFAR(X), we denote a dataset with X randomly selected examples from each CIFAR100 class. And ImageNet(X) denotes a dataset constructed analogously. The difference between ImageNet and CIFAR100 is that the number of images in a class is not strictly equal, varying from 700 to 1.25k images per class. Thus we denote the full-dataset setting as ImageNet(full), instead of ImageNet(1200).

Compared Models To verify the superiority of the proposed HDKD method, we compare it with the following state-of-the-art methods:
(1). MSDNet (Huang et al. 2017a): As our proposed method and other comparison methods are all based on MSDNet, it serves as a direct baseline in our experiments.
(2). DBT (Phuong and Lampert 2019): Distillation Based Training (DBT) is a self-distillation training strategy for MSDNet, encouraging the early exits to imitate the last exit.
(3). Impr (Li et al. 2019): It employs both gradient equilibrium and knowledge distillation techniques to improve the training for MSDNet.
(4). NaiveDKD: Naive Dense Knowledge Distillation introduces the dense knowledge distillation into multi-exit training with a simple uniform weighting scheme shown in Eq. 4.
(5). HMSDNet: Harmonized weighting scheme is only conducted for the multi-exit cross-entropy loss of MSDNet, i.e., Eq. 5 with all $\gamma_{ij} = 0$.
(6). HDBT: We also conduct the experiment for learning harmonized weights for the multiple objectives of DBT, optimizing Eq. 5 with all $\gamma_{ij} = 0$ for $i < K$.
(7). HDKD: Harmonized Dense Knowledge Distillation (HDKD) is our proposed training strategy.

Model Architecture We adopt the MSDNet (Huang et al. 2017a) as the basic architecture. For anytime prediction setting, the CIFAR MSDNet has 36 layers and 8 exits. The ImageNet MSDNet is of 23 layers and 5 exits. For the budgeted batch classification setting, three models with 5 exits and depth of $\{23, 33, 38\}$ are chosen for ImageNet classification. On CIFAR100, 4 models with $\{4, 5, 6, 8\}$ exits are leveraged, with the corresponding depth of $\{10, 15, 21, 36\}$.

Optimization and Hyper-parameters We train all models from the random initialization performed by SGD with a momentum of 0.9, and a weight decay of 10^{-4}. For CIFAR100, the model is trained with a batch size of 128 for

	flops	CIFAR(150)							CIFAR(250)						
	(M)	MSD Net	Impr	DBT	Naive DKD	HMSD Net	HDBT	HDKD	MSD Net	Impr	DBT	Naive DKD	HMSD Net	HDBT	HDKD
Exit-1	6.85	74.44	74.54	80.92	**82.20**	74.94	80.09	81.38	81.99	82.05	87.08	**87.57**	82.50	85.15	85.82
Exit-2	14.4	78.97	79.67	84.20	**84.72**	79.07	82.91	83.87	84.39	84.76	88.54	**88.49**	84.89	86.35	86.72
Exit-3	27.3	81.27	82.18	84.95	**84.87**	81.54	84.03	84.65	86.00	86.47	89.55	**89.63**	86.50	88.29	88.73
Exit-4	48.4	82.72	83.69	85.67	86.10	83.14	85.69	**86.33**	87.60	88.12	**90.25**	89.78	88.11	89.37	89.69
Exit-5	81.6	84.12	85.11	86.24	86.46	84.56	86.32	**86.76**	88.70	89.21	90.74	90.86	89.20	90.53	**90.86**
Exit-6	113	85.10	86.80	86.59	86.90	85.39	86.84	**87.20**	89.08	89.96	90.51	90.72	89.57	90.66	**90.94**
Exit-7	153	85.30	**87.21**	86.36	86.70	85.73	86.86	87.14	89.14	90.16	90.36	90.17	89.62	90.70	**90.97**
Exit-8	193	84.85	86.82	85.70	85.99	84.96	86.94	**87.43**	90.15	91.18	90.24	89.89	90.61	91.39	**91.59**
	flops	CIFAR(300)							CIFAR(500)						
	(M)	MSD Net	Impr	DBT	Naive DKD	HMSD Net	HDBT	HDKD	MSD Net	Impr	DBT	Naive DKD	HMSD Net	HDBT	HDKD
Exit-1	6.85	83.33	83.38	86.56	**87.66**	83.85	86.55	87.22	86.44	86.47	**88.91**	88.52	86.98	87.50	87.64
Exit-2	14.4	85.75	86.12	88.28	**88.92**	86.26	88.04	88.48	88.91	89.11	**91.09**	90.74	89.44	90.48	90.74
Exit-3	27.3	87.38	87.88	89.14	**89.29**	87.90	88.92	89.18	90.61	90.86	92.30	92.48	91.14	92.58	**92.94**
Exit-4	48.4	88.91	89.43	90.38	90.71	89.42	90.55	**90.83**	91.96	92.03	93.26	92.94	92.49	93.22	**93.40**
Exit-5	81.6	89.86	90.39	90.87	90.92	90.37	91.11	**91.29**	92.57	92.85	93.37	93.36	93.10	93.68	**93.83**
Exit-6	113	90.23	91.14	90.96	91.19	90.73	91.23	**91.35**	92.91	93.38	93.41	93.36	93.42	93.85	**93.96**
Exit-7	153	90.24	91.23	90.74	91.19	90.73	91.12	**91.22**	92.80	93.30	93.00	93.07	93.30	93.85	**93.98**
Exit-8	193	90.99	92.04	91.02	91.61	91.45	92.19	**92.37**	92.93	93.46	92.90	92.80	93.40	93.89	**94.02**

Table 2: Top-5 classification accuracy for each individual classifier in 8-exits MSDNet on CIFAR100. We compare the models trained by the exit-wise loss v.s. distillation-based training methods v.s. the proposed Harmonized weighting versions on CIFAR100 with 150, 250, 350, 500 images per class. Bold values indicate statistically significant performance.

300 epochs. The learning rate is initialized as 0.1, divided by 10 after epochs 150 and 225. For ImageNet, we use a larger batch size of 1024 by default instead of 256, following the common setting suggested by (Goyal et al. 2017).

Anytime Prediction

In this setting, the model progressively updates its predicted results and can be forced to output its most recent prediction at any time. The classification accuracies of individual classifiers are reported for each method.

ImageNet Results In Table 1, we summarize the top-5 accuracy of each exit classifier on ImageNet for different compared models. DBT and Impr neglects the rich beneficial supervision information of dense distillation and the balance of the joint classification and distillation loss. Through incorporating dense distillation into the multi-exit learning, NaiveDKD achieves much better performance for the early exits, but the performance of late exits stagnates due to the conflicting between losses. Via employing the harmonized weighting scheme to balance the competing multi-objectives during the optimization, HDBT outstrips DBT, however, each exit still can not fully exploit the useful information from all later exits. For the late exis, HDKD and HDBT outstrip MSDNet, while DBT and NaiveDKD behave similarly to MSDNet. It demonstrates that the appropriate choice of loss weights also benefits the teacher exits and helps the overall performance of multi-exit learning. Consequently, from Table 1, we observe that HDKD out-performs all other compared models, and the accuracy scores achieved by HDKD are much higher than those achieved by the competing models in most cases. For example, compared with Impr and DBT, HDKD achieves 8.56% and 2.5% gains respectively in the exit-1 classifier on ImageNet(100).

CIFAR100 Results We observe analogous results on CIFAR100 in Table 2. By allowing each exit to learn from the last exit, DBT greatly improves the early exits, however, the performance of late exits keeps roughly the same with the MSDNet baseline. With dense knowledge distillation, NaiveDS further improves the early exits but the performance late exits still keep stagnating. H-MSDT and HDBT achieve better performance than MSDT and DBT in most cases, which demonstrates the effectiveness of the harmonized loss weights learning.

Compared with DBT, HDKD leverages dense knowledge distillation with a harmonized loss weighting scheme and achieves better performance in most cases, especially for the late exits. For example, HDKD improves the performance of exit-8 over DBT by 1.73%, 1.35%, 1.35%, and 1.12% respectively on CIFAR(150/250/350/500). This demonstrates that the dynamically changed configuration γ guides the model converge to a better local optimum, where the early exit benefits the late exit.

Admittedly, for some early exits, HDKD performs slightly worse than NaiveDKD, which might illustrate that some early exits require uniform supervision from the later exits. Since the harmonized weighting scheme considers the overall performance of all exits, it may be not optimal for some early exits. However, HDKD still consistently performs better than the competing models in most cases. Future work should continue to explore how to further resolve the conflicts and improve the early and late exits more harmoniously.

		HDKD with Different γ Initializaiton			Fixed γ		α of HDKD				
	MSDNet	DBT-like	DS-like	Rand	Fixed-1	Fixed-2	0	0.1	0.5	0.9	1
Exit-1	72.45	79.02	78.72	79.22	78.47	78.49	77.51	76.64	78.72	77.07	**79.55**
Exit-2	79.95	81.78	82.58	82.30	82.61	82.21	81.99	**83.26**	82.58	82.98	81.84
Exit-3	83.12	83.67	85.13	85.51	84.23	83.69	**85.65**	85.34	85.13	84.83	84.94
Exit-4	83.61	84.49	85.87	85.95	84.85	83.87	86.17	**86.26**	85.87	85.92	85.54
Exit-5	84.66	85.65	87.37	87.70	85.91	83.33	**87.94**	87.54	87.37	87.31	86.11

Table 3: Ablation study on ImageNet(300) to verify the effect of γ initialization, fixed configuration γ, and α in HDKD.

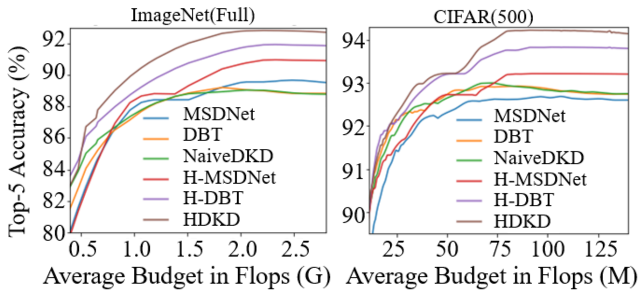

Figure 2: Accuracy (top-5) v.s. flops per image in budgeted batch classification on the ImageNet(Full) and CIFAR(500).

Budgeted Batch Classification

In this setting, the model receives a batch of instances and a computational budget. Constrained by the budget, "easy" examples could exit at early classifiers while "hard" examples propagate throughout the network. For each method, several corresponding models with different depths are trained, each of which covers a different range of computational budgets. At the inference stage, the best model is selected for each budget based on its accuracy on the validation set, and the corresponding accuracy is plotted.

The performance of budgeted batch classification on ImageNet(Full) and CIFAR(500) are shown in Fig. 2. On ImageNet(Full), HDKD improves the top-5 accuracy by 3.5% compared to DBT at high-flops region around 2.5G and by 1.5% at low-flops region around 0.5G, which shows the advantage of the dense distillation and harmonized weighting scheme. On CIFAR(500), with an average budget of 25M, the improvement of HDKD over the baseline MSDNet is slightly better than DBT, i.e., 0.83% versus 0.71%, and at the high-flops region of 125M, HDKD exceeds the DBT with a significant margin of 1.66%.

Ablation Study

In this section, we examine the sensitivity of parameter settings for the proposed HDKD training method.

Sensitivity Analysis of γ Initialization As a benefit of the HDKD, the exhausted tuning of γ is avoided, and the initialization of γ should not severely affect performance. To test the sensitivity, we initialize γ with the following configurations: 1). *DBT-like*: the last exit teaches all the others exits; 2). *DS-like*: each exit equally learns from all later exits; 3). *Rand*: γ is sampled randomly from a uniform distribution.

The comparison among different γ initializations is shown in Table 3. We observe that the impact of different initializations is wiped out in the learning process and the performance is not sensitive to the γ initialization.

The Importance of Evolving γ One may argue that the advantage of HDKD is simply hyper-parameter searching, i.e., seeking for a final fixed best γ configuration. However, we observe that the function of harmonized weighting scheme is guiding the model to a better local optimum and the final searched configuration γ is not necessarily beneficial to the whole training process.

To understand this, we conduct experiments by extracting the final γ in a converged model trained by HDKD, using it as fixed γ parameters to train new models from scratch. The performance is shown in Column Fixed-1 and Fixed-2 in Table 3. We see that both Fixed-1 and Fixed-2 perform worse than Rand, which demonstrates that prematurely applying the final γ causes severe performance degradation. Thus, we conclude that HDKD is not merely searching for a final best γ. The highly dynamic training process expects a continuously evolving γ and HDKD is responsible for that.

Sensitivity Analysis of α The validation module of HDKD in Eq. 10 is composed of *cross-entropy loss term* (weighted by α) and *ratio regularizer term* (weighted by $1-\alpha$). As observed in Table 3, When $\alpha = 1$, only the cross-entropy loss term on the validation data is leveraged, and the early exit performs better with the overall performance decreasing. It is because that the early exit loss is usually larger than the later ones, and the bilevel optimizer may over-focus on decreasing the early ones and hurts the later ones. Therefore, the ratio regularizer that involves the current cross-entropy loss and the estimation of expectation loss is an important design for alleviating this influence. The performance is not sensitive to the choice of α in a wide range, and by default, we set $\alpha = 0.5$ without tuning.

Conclusion

In this paper, we introduce a harmonized dense knowledge distillation training technique for multi-exit architectures, where a harmonized weighting scheme is designed for the multi-objective optimization problem consisting of multi-exit classification loss and dense distillation loss. A bilevel optimization algorithm is introduced for alternatively updating the weights of multiple objectives and the multi-exit network parameters. Extensive experiments show that the HDKD strategy consistently improves the performance of the state-of-the-art multi-exit neural networks.

Acknowledgments

This work is supported in part by National Key R&D Program of China (No. 2018YFB1403600), NSFC (No. U20A201548, 61702448), Artificial Intelligence Research Foundation of Baidu Inc., Horizon Robotics, and Cloud TPUs from Google's TensorFlow Research Cloud (TFRC).

References

Abadi, M.; Barham, P.; Chen, J.; Chen, Z.; Davis, A.; Dean, J.; Devin, M.; Ghemawat, S.; Irving, G.; Isard, M.; et al. 2016. Tensorflow: A system for large-scale machine learning. In *12th USENIX Symposium on Operating Systems Design and Implementation*, 265–283.

Bolukbasi, T.; Wang, J.; Dekel, O.; and Saligrama, V. 2017. Adaptive neural networks for efficient inference. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, 527–536.

Courbariaux, M.; and Bengio, Y. 2016. BinaryNet: Training Deep Neural Networks with Weights and Activations Constrained to +1 or -1. *CoRR* abs/1602.02830.

Finn, C.; Abbeel, P.; and Levine, S. 2017. Model-agnostic meta-learning for fast adaptation of deep networks. *arXiv preprint arXiv:1703.03400* .

Goyal, P.; Dollár, P.; Girshick, R.; Noordhuis, P.; Wesolowski, L.; Kyrola, A.; Tulloch, A.; Jia, Y.; and He, K. 2017. Accurate, large minibatch sgd: Training imagenet in 1 hour. *arXiv preprint arXiv:1706.02677* .

Han, S.; Pool, J.; Tran, J.; and Dally, W. J. 2015. Learning both Weights and Connections for Efficient Neural Network. In *Advances in Neural Information Processing Systems*, 1135–1143.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 770–778.

Hinton, G.; Vinyals, O.; and Dean, J. 2015. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531* .

Howard, A. G.; Zhu, M.; Chen, B.; Kalenichenko, D.; Wang, W.; Weyand, T.; Andreetto, M.; and Adam, H. 2017. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861* .

Huang, G.; Chen, D.; Li, T.; Wu, F.; van der Maaten, L.; and Weinberger, K. Q. 2017a. Multi-scale dense networks for resource efficient image classification. *arXiv preprint arXiv:1703.09844* .

Huang, G.; Liu, Z.; Maaten, L. v. d.; and Weinberger, K. Q. 2017b. Densely Connected Convolutional Networks. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* .

Ioffe, S.; and Szegedy, C. 2015. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167* .

Ke, L.; Chang, M.-C.; Qi, H.; and Lyu, S. 2018. Multi-scale structure-aware network for human pose estimation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 713–728.

Krizhevsky, A.; Nair, V.; and Hinton, G. 2009. Cifar-10 and cifar-100 datasets. *URl: https://www. cs. toronto. edu/kriz/cifar. html* 6.

Li, B.; Liu, Y.; and Wang, X. 2019. Gradient Harmonized Single-Stage Detector. *Proceedings of the AAAI Conference on Artificial Intelligence* 33: 8577–8584.

Li, H.; Zhang, H.; Qi, X.; Ruigang, Y.; and Huang, G. 2019. Improved Techniques for Training Adaptive Deep Networks. *IEEE International Conference on Computer Vision (ICCV)* .

Lin, J.; Rao, Y.; Lu, J.; and Zhou, J. 2017. Runtime neural pruning. In *Advances in Neural Information Processing Systems*, 2181–2191.

Liu, L.; and Deng, J. 2017. Dynamic deep neural networks: Optimizing accuracy-efficiency trade-offs by selective execution. *arXiv preprint arXiv:1701.00299* .

Lorraine, J.; and Duvenaud, D. 2018. Stochastic hyperparameter optimization through hypernetworks. *arXiv preprint arXiv:1802.09419* .

Luan, Y.; Zhao, H.; Yang, Z.; and Dai, Y. 2019a. MSD: Multi-Self-Distillation Learning via Multi-classifiers within Deep Neural Networks. *arXiv preprint arXiv:1911.09418* .

Luan, Y.; Zhao, H.; Yang, Z.; and Dai, Y. 2019b. MSD: Multi-Self-Distillation Learning via Multi-classifiers within Deep Neural Networks. *arXiv preprint arXiv:1911.09418* .

McIntosh, L.; Maheswaranathan, N.; Sussillo, D.; and Shlens, J. 2018. Recurrent Segmentation for Variable Computational Budgets. *IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)* .

Nichol, A.; Achiam, J.; and Schulman, J. 2018. On first-order meta-learning algorithms. *arXiv preprint arXiv:1803.02999* .

Passalis, N.; Raitoharju, J.; Tefas, A.; and Gabbouj, M. 2020. Efficient adaptive inference for deep convolutional neural networks using hierarchical early exits. *Pattern Recognition* 105: 107346.

Phuong, M.; and Lampert, C. H. 2019. Distillation-Based Training for Multi-Exit Architectures. In *Proceedings of the IEEE International Conference on Computer Vision*, 1355–1364.

Rajeswaran, A.; Finn, C.; Kakade, S. M.; and Levine, S. 2019. Meta-learning with implicit gradients. In *Advances in Neural Information Processing Systems*, 113–124.

Rastegari, M.; Ordonez, V.; Redmon, J.; and Farhadi, A. 2016. XNOR-Net: ImageNet Classification Using Binary Convolutional Neural Networks. In Leibe, B.; Matas, J.; Sebe, N.; and Welling, M., eds., *ECCV*, volume 9908, 525–542.

Ravi, S.; and Larochelle, H. 2016. Optimization as a model for few-shot learning. In *International Conference on Learning Representations*.

Romero, A.; Ballas, N.; Kahou, S. E.; Chassang, A.; Gatta, C.; and Bengio, Y. 2014. Fitnets: Hints for thin deep nets. *arXiv preprint arXiv:1412.6550* .

Russakovsky, O.; Deng, J.; Su, H.; Krause, J.; Satheesh, S.; Ma, S.; Huang, Z.; Karpathy, A.; Khosla, A.; Bernstein, M.; et al. 2015. Imagenet large scale visual recognition challenge. *International journal of computer vision* 115(3): 211–252.

Sun, Q.; Liu, Y.; Chua, T.-S.; and Schiele, B. 2019. Meta-Transfer Learning for Few-Shot Learning. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* .

Teerapittayanon, S.; McDanel, B.; and Kung, H. T. 2016. BranchyNet: Fast inference via early exiting from deep neural networks. In *23rd International Conference on Pattern Recognition, ICPR 2016, Cancún, Mexico, December 4-8, 2016*, 2464–2469. IEEE.

Veit, A.; and Belongie, S. 2018. Convolutional networks with adaptive inference graphs. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 3–18.

Wang, X.; and Li, Y. 2020. Gradient Deconfliction-Based Training For Multi-Exit Architectures. In *IEEE International Conference on Image Processing, ICIP 2020, Abu Dhabi, United Arab Emirates, October 25-28, 2020*, 1866–1870. IEEE.

Wang, X.; Yu, F.; Dou, Z.-Y.; Darrell, T.; and Gonzalez, J. E. 2018a. Skipnet: Learning dynamic routing in convolutional networks. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 409–424.

Wang, Y.; Wang, L.; You, Y.; Zou, X.; Chen, V.; Li, S.; Huang, G.; Hariharan, B.; and Weinberger, K. Q. 2018b. Resource Aware Person Re-identification Across Multiple Resolutions. *IEEE Conference on Computer Vision and Pattern Recognition* .

Wu, Z.; Nagarajan, T.; Kumar, A.; Rennie, S.; Davis, L. S.; Grauman, K.; and Feris, R. 2018. Blockdrop: Dynamic inference paths in residual networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 8817–8826.

You, C.; Yang, Q.; Gjesteby, L.; Li, G.; Ju, S.; Zhang, Z.; Zhao, Z.; Zhang, Y.; Cong, W.; Wang, G.; et al. 2018. Structurally-sensitive multi-scale deep neural network for low-dose CT denoising. *IEEE Access* 6: 41839–41855.

Yu, T.; Kumar, S.; Gupta, A.; Levine, S.; Hausman, K.; and Finn, C. 2020. Gradient Surgery for Multi-Task Learning. *arXiv preprint arXiv:2001.06782* .

Zhang, C.; Ren, M.; and Urtasun, R. 2018. Graph hypernetworks for neural architecture search. *arXiv preprint arXiv:1810.05749* .

Zhang, L.; Song, J.; Gao, A.; Chen, J.; Bao, C.; and Ma, K. 2019. Be Your Own Teacher: Improve the Performance of Convolutional Neural Networks via Self Distillation. *IEEE International Conference on Computer Vision (ICCV)* .

Zhu, M.; and Gupta, S. 2017. To prune, or not to prune: exploring the efficacy of pruning for model compression. *arXiv preprint arXiv:1710.01878* .