Neural Networks 126 (2020) 250–261

Contents lists available at ScienceDirect

Neural Networks

journal homepage: www.elsevier.com/locate/neunet

Multi-way backpropagation for training compact deep neural networks

Yong Guo [a,1], Jian Chen [a,1], Qing Du [a,1], Anton Van Den Hengel [b], Qinfeng Shi [b], Mingkui Tan [a,c,*]

[a] South China University of Technology, China
[b] University of Adelaide, Australia
[c] Guangzhou Laboratory, China

ARTICLE INFO

Article history:
Received 26 December 2019
Received in revised form 26 February 2020
Accepted 2 March 2020
Available online 26 March 2020

Keywords:
Backpropagation
Supervision vanishing
Compact model

ABSTRACT

Depth is one of the key factors behind the success of convolutional neural networks (CNNs). Since ResNet (He et al., 2016), we are able to train very deep CNNs as the gradient vanishing issue has been largely addressed by the introduction of skip connections. However, we observe that, when the depth is very large, the intermediate layers (especially shallow layers) may fail to receive sufficient supervision from the loss due to severe transformation through long backpropagation path. As a result, the representation power of intermediate layers can be very weak and the model becomes very redundant with limited performance. In this paper, we first investigate the supervision vanishing issue in existing backpropagation (BP) methods. And then, we propose to address it via an effective method, called Multi-way BP (MW-BP), which relies on multiple auxiliary losses added to the intermediate layers of the network. The proposed MW-BP method can be applied to most deep architectures with slight modifications, such as ResNet and MobileNet. Our method often gives rise to much more compact models (denoted by "Mw+Architecture") than existing methods. For example, MwResNet-44 with 44 layers performs better than ResNet-110 with 110 layers on CIFAR-10 and CIFAR-100. More critically, the resultant models even outperform the light models obtained by state-of-the-art model compression methods. Last, our method inherently produces multiple compact models with different depths at the same time, which is helpful for model selection. Extensive experiments on both image classification and face recognition demonstrate the superiority of the proposed method.

1. Introduction

Since 2012 when AlexNet won the first place in the ImageNet competition (Krizhevsky, Sutskever, & Hinton, 2012), convolutional neural networks (CNNs) (LeCun et al., 1989) have been producing state-of-the-art results in many of the most challenging vision tasks including image classification (Guo, Wu, Deng, Chen, & Tan, 2018; He, Zhang, Ren, & Sun, 2016a; Lee, Xie, Gallagher, Zhang, & Tu, 2015), face recognition (Chen, Liu, Gao, & Han, 2018), semantic segmentation (Ibtehaz & Rahman, 2020), and many other applications (Cao et al., 2018; Guo et al., 2019; Liu et al., 0000; Wang, Dai, Cai, Sun and Chen, 2018). Moreover, deep CNNs have also become the workhorse of many other tasks and real-world applications beyond computer vision, such as natural language understanding (Gonzalez-Dominguez, Lopez-Moreno, Moreno, & Gonzalez-Rodriguez, 2015) and speech recognition (LeCun, Bengio, & Hinton, 2015).

Recent studies (Srivastava, Greff, & Schmidhuber, 2015; Szegedy et al., 2015) have demonstrated the importance of depth to the representation power of neural networks. Recently, the training of very deep models becomes possible (*e.g.*, ResNet He et al., 2016a), since the gradient vanishing issue has been largely addressed by introducing skip (*i.e.*, shortcut) connections. However, when the depth becomes large, the model may incur training difficulties due to what we call *supervision vanishing* problem. Specifically, even with the skip connection or other advanced structures, the supervision from the loss tends to fade through a long backpropagation path (Pascanu, Mikolov, & Bengio, 0000; Shen, Lin, & Huang, 2016). As a result, the intermediate layers fail to receive sufficient information from the loss, which may lead to severe internal model redundancy. The existence of such internal redundancy often means more parameters, larger model size, higher inference cost, more energy consumption, and/or degraded performance (Ba & Caruana, 2014). Note that in real-world

* Corresponding author.
E-mail addresses: guo.yong@mail.scut.edu.cn (Y. Guo), ellachen@scut.edu.cn (J. Chen), duqing@scut.edu.cn (Q. Du), anton.vandenhengel@adelaide.edu.au (A. Van Den Hengel), javen.shi@adelaide.edu.au (Q. Shi), mingkuitan@scut.edu.cn (M. Tan).

[1] Yong Guo, Jian Chen, Qing Du contributed equally.

https://doi.org/10.1016/j.neunet.2020.03.001
0893-6080/

applications, we have an urgent demand for efficient models with smaller model size, less energy consumption, and promising performance. In this sense, how to reduce the internal model redundancy in CNNs while keeping/improving the performance is an important and urgent problem.

In this paper, we extensively study the supervision vanishing issue in existing BP methods and investigate why these methods would incur internal model redundancy even in carefully designed compact architectures. One can alleviate this issue by introducing auxiliary losses to the network (Szegedy et al., 2015). However, how to well exploit auxiliary losses to obtain more compact models still remains a question. Recent studies, such as Deeply Supervised Network (DSN) (Lee et al., 2015) and GoogLeNet (Szegedy et al., 2015), consider multiple losses as a joint loss and sum up the gradients from relevant losses into a joint one in backpropagation (BP). These methods have two major limitations. First, the multiple losses may have conflicts with each other due to their different positions in the network and simply summing them up may incur severe training difficulties. To alleviate this, one should carefully adjust the weights of losses during the training (Lee et al., 2015), which, however, limits its applicability to general cases. Second, these methods may still suffer from supervision vanishing and hence obtain only marginal improvement in the performance.

To address the supervision vanishing issue and thus reduce the internal redundancy of deep models, we propose a novel training method, called Multi-way BP (MW-BP), in which we let multiple losses share one forward propagation but conduct multiple separate backpropagations (one for each loss separately). In this way, it helps to alleviate the vanishing of supervision and obtain more compact models. Note that in this paper we do not attempt to design compact models (Howard et al., 0000; Zhang, Zhou, Lin, & Sun, 2018) or search for some compact architectures (Guo et al., 2019; Tan et al., 2019). Instead, we focus on improving the training of CNNs to obtain compact models. In practice, the proposed training paradigm can be applied to a variety of deep architectures, including both large models like ResNet (He et al., 2016a) and lightweight models like MobileNet (Sandler, Howard, Zhu, Zhmoginov, & Chen, 2018).

In the paper, we make the following contributions.

- We investigate the supervision vanishing issue when training deep models using existing BP methods. To address the issue, we exploit multiple auxiliary losses to provide additional supervision and propose an adaptive weighting scheme to alleviate the conflicts among multiple losses.
- We propose a simple but effective Multi-way BP (MW-BP) method to train deep models with multiple losses. During the training, we apply one shared forward propagation for all the losses but sequentially perform a backpropagation for each loss. In this way, the intermediate layers can receive sufficient information from each loss and hence their representation power can be significantly improved. Our MW-BP can be applied to various architectures, such as ResNet (He et al., 2016a), DenseNet (Huang, Liu, Van Der Maaten, & Weinberger, 2017), Inception network (Szegedy, Ioffe, Vanhoucke, & Alemi, 2017) and MobileNet (Sandler et al., 2018). We demonstrate the superiority of the proposed method with various architectures on both **image classification** and **face recognition** tasks.
- The proposed method can effectively reduce the internal model redundancy and often gives rise to more compact models than the models trained by existing BP methods. For example, MwResNet-44 of **44** layers outperforms ResNet-110 of **110** layers on several benchmark data sets. More critically, the models obtained by MW-BP even outperform the carefully compressed models obtained by state-of-the-art compression methods, in terms of both accuracy and model compactness (See Section 5.3).
- Equipped with MW-BP, we inherently produce multiple models of different depths at the same time. Surprisingly, these intermediate models often outperform their full-depth counterparts or even deeper ones trained by existing BP methods. In fact, we can choose an appropriate one as the final model. In this sense, the proposed method is helpful for model selection.

2. Related work

2.1. Deep models with multiple losses

Employing auxiliary classifiers to aid in the training has been investigated in many state-of-the-art methods. In GoogLeNet (Szegedy et al., 2015), two auxiliary classifiers are connected to the intermediate layers with very small weights for them to ensure the convergence (*i.e.*, 0.3 for the auxiliary losses). In DSN (Lee et al., 2015), each convolution layer is associated with a classifier. To avoid the training difficulty, DSN keeps the losses for a number of epochs and discard all but the final loss to finish the remaining epochs. Unlike these methods, the proposed MW-BP does not need to set such a small weight to auxiliary losses or discard any loss during the training, which helps to simultaneously produce multiple models with promising performance, with the ensuing benefits for model selection.

2.2. Backpropagation methods

Besides the standard BP method for handling a single loss, several BP variants have been proposed for dealing with multiple losses, including Joint BP (Lee et al., 2015; Szegedy et al., 2015) and Relay BP (Shen et al., 2016). Joint BP, that is used in GoogLeNet (Szegedy et al., 2015) and DSN (Lee et al., 2015), considers a weighted sum of multiple losses as a joint one and updates the model parameters with the joint gradients. Another variant, called Relay BP (Shen et al., 2016), discards the gradients from those losses with the long backpropagation paths to better preserve the supervision signal. In Joint BP and Relay BP, the multiple losses work jointly for the training and the gradients w.r.t. different losses are summed up in a single backpropagation. However, even with auxiliary losses, the supervision vanishing issue can still occur in these methods. Unlike Joint BP and Relay BP, in Drucker and Le Cun (1992), a double backpropagation method was proposed. Different from these methods, our MW-BP conducts a backpropagation for each loss separately. In this way, the intermediate layers can receive sufficient supervision from the nearest losses and the supervision vanishing issue can be alleviated.

2.3. Compact model design

Many attempts have been made to design compact models, such as ResNeXt (Xie, Girshick, Dollár, Tu, & He, 2017), MobileNet (Howard et al., 0000), ShuffleNet (Zhang et al., 2018), *etc.* Relying on ResNet (He et al., 2016a), ResNeXt (Xie et al., 2017) introduces group convolutions into the architecture to improve the model compactness. With the focus on mobile devices, MobileNet (Howard et al., 0000) employs depthwise separable convolution to build lightweight networks. ShuffleNet (Zhang et al., 2018) uses a channel shuffle operation to reduce the model size and inference complexity. Instead of designing compact architectures, we focus on devising an effective training method to obtain more compact models. Empirically, our MW-BP exhibits good compatibility with various architectures and can produce more compact models than the ones trained by existing BP methods.

2.4. Model compression methods

Recently, many efforts have been made to obtain compact models via model compression techniques. For example, one can prune unimportant channels based on a pretrained CNN and introduce sparsity into the filters of convolution (He, Liu, Wang, Hu, & Yang, 2019; He, Zhang, & Sun, 2017; Li, Kadav, Durdanovic, Samet, & Graf, 2017; Luo, Wu, & Lin, 2017; Zhuang et al., 2018). Li et al. utilize an ℓ_1-norm criterion to prune unimportant filters (Li et al., 2017). In He et al. (2019), He et al. propose to use the geometric median of the filters to perform channel pruning. Unlike these methods, we seek to develop an effective training algorithm to produce compact models. More critically, the resultant models trained by MW-BP even outperform the carefully compressed models obtained by state-of-the-art model compression methods (See results and comparisons in Section 5.3).

3. Supervision vanishing in deep networks

In this section, we study the supervision vanishing issue in training deep networks.

Without loss of generality, we consider an L-layers network that conducts *forward propagation* for any layer l by

$$\mathbf{y}_l = \lambda_l \mathbf{x}_l + \mathcal{F}_l(\mathbf{x}_l, \mathbf{W}_l), \quad \mathbf{x}_{l+1} = h(\mathbf{y}_l), \tag{1}$$

where $\lambda_l \in \{0, 1\}$. Here, $\mathbf{x}_l$ and $\mathbf{x}_{l+1}$ denote the input and output of the lth layer, respectively; $\mathbf{y}_l$ denotes the intermediate feature before activation; h is a nonlinear activation function (*e.g.*, Rectified Linear Unit (ReLU) (Nair & Hinton, 2010) or Sigmoid function); and $\mathcal{F}_l$ denotes a transformation function (*e.g.*, convolution operation) parameterized by $\mathbf{W}_l$. When $\lambda_l = 0$, Eq. (1) represents the forward propagation process of plain deep networks, such as AlexNet (Krizhevsky et al., 2012) and VGG (Simonyan & Zisserman, 2015). When $\lambda_l = 1$, there is a shortcut connection between the lth and $(l+1)$th layer. The shortcut connection, an effective technique to avoid the gradient vanishing issue in BP, enables us to train very deep models that are known as the residual networks (He et al., 2016a).

In fact, one can use stochastic gradient descent (SGD) (Wilson & Martinez, 2003) to update the parameters $\{\mathbf{W}_l\}_{l=0}^{L-1}$. Let ξ be the loss function, the gradient of ξ w.r.t. $\mathbf{W}_l$ can be computed by

$$\frac{\partial \xi}{\partial \mathbf{W}_l} = \frac{\partial \xi}{\partial \mathbf{x}_{l+1}} \frac{\partial \mathbf{x}_{l+1}}{\partial \mathbf{W}_l}, \tag{2}$$

where $\frac{\partial \xi}{\partial \mathbf{x}_{l+1}}$ denotes the gradient propagated from ξ to some intermediate layer. By applying the chain rule w.r.t. Eq. (1), such gradient for any layer l can be written as

$$\begin{aligned} \frac{\partial \xi}{\partial \mathbf{x}_l} &= \frac{\partial \xi}{\partial \mathbf{x}_L} \left(\frac{\partial \mathbf{x}_L}{\partial \mathbf{x}_{L-1}} \cdots \frac{\partial \mathbf{x}_{l+1}}{\partial \mathbf{x}_l} \right) \\ &= \frac{\partial \xi}{\partial \mathbf{x}_L} \prod_{j=l}^{L-1} \frac{\partial \mathbf{x}_{j+1}}{\partial \mathbf{y}_j} \frac{\partial \mathbf{y}_j}{\partial \mathbf{x}_j} \\ &= \frac{\partial \xi}{\partial \mathbf{x}_L} \prod_{j=l}^{L-1} \mathbf{T}_j(\mathbf{W}_j), \end{aligned} \tag{3}$$

where

$$\mathbf{T}_j(\mathbf{W}_j) = \frac{\partial \mathbf{x}_{j+1}}{\partial \mathbf{y}_j} \Big(\lambda_j \mathbf{I} + \partial_{\mathbf{x}_j} \mathcal{F}_j(\mathbf{x}_j, \mathbf{W}_j) \Big). \tag{4}$$

Definition 1 (*Supervision Information*)**.** We define $\frac{\partial \xi}{\partial \mathbf{x}_l}$, the partial gradient of ξ w.r.t. $\mathbf{x}_l$, as the *supervision information* obtained from the loss. From Eq. (3), the partial gradient $\frac{\partial \xi}{\partial \mathbf{x}_l}$ contains two parts, namely $\frac{\partial \xi}{\partial \mathbf{x}_L}$ and $\prod_{j=l}^{L-1} \mathbf{T}_j(\mathbf{W}_j)$, where the term $\frac{\partial \xi}{\partial \mathbf{x}_L}$ is directly related to the loss ξ.

Note that each $\mathbf{T}_j(\mathbf{W}_j)$ is a transformation matrix that transforms $\frac{\partial \xi}{\partial \mathbf{x}_L}$ a bit. Then, the term $\prod_{j=l}^{L-1} \mathbf{T}_j(\mathbf{W}_j)$ will transform the gradient $\frac{\partial \xi}{\partial \mathbf{x}_L}$ through a series of layers from the final layer to the lth layer. When $(L-l)$ is large, the transformation $\prod_{j=l}^{L-1} \mathbf{T}_j(\mathbf{W}_j)$ can be too severe and make the component $\frac{\partial \xi}{\partial \mathbf{x}_L}$ negligible in $\frac{\partial \xi}{\partial \mathbf{x}_l}$. As a result, the shallow layers may not receive sufficient supervision from the final loss ξ due to the severe transformation of the long-path backpropagation. We call this phenomenon the **supervision vanishing issue**. As a result, the intermediate layers (especially the shallow layers) may have limited representation power. In this sense, there would be a lot of redundant parameters in the intermediate layers, leading to severe **internal redundancy** in deep models. In practice, such redundancy would deteriorate the performance of deep models (See results in Table 6).

4. Multi-way backpropagation for deep models with auxiliary losses

4.1. Deep model with auxiliary losses

As mentioned in Section 3, the standard BP with a single loss may incur supervision vanishing issue and lead to severe internal model redundancy. To address this, it is natural to introduce auxiliary losses to the network to provide additional supervision for shallow layers, similar to DSN (Lee et al., 2015) and GoogLeNet (Szegedy et al., 2015). However, how to avoid the possible conflicts among different losses and well exploit the information from auxiliary losses to train compact models are still open questions.

Taking an L-layer ResNet for example, as shown in Fig. 1, we introduce K auxiliary losses to the network, with each being built on the top of an average pooling layer. Including the final loss ξ, we have $\widehat{K} = K + 1$ losses in total. We can either apply the same form of the final loss ξ_K to each auxiliary loss $\{\xi_i\}_{i=0}^{K-1}$ or exploit other forms of losses for them. We use L_i to indicate the layer to which the ith loss is connected. Note that each loss ξ_i is associated with a model of depth L_i. Thus, with multiple losses, we can inherently obtain multiple models of different depths (See Fig. 1).

4.1.1. Adaptive weighting scheme for auxiliary losses

The multiple losses may incur conflicts and thus hamper the training of deep networks. Specifically, given multiple losses with different forms, we have different objectives and the losses would naturally incur conflicts. Moreover, note that shallow layers often have less representation power than deep layers. If we add the losses to different layers, even with the same loss form, the shallow losses may produce inaccurate gradients and hence incur conflicts with the deep losses. To alleviate the loss conflict issue, we develop an adaptive weighting scheme for different losses.

Since the auxiliary losses are not equally important, we should impose different confidence, denoted by $\{\gamma_i\}_{i=0}^{K-1}$, over them. By default, we set $\gamma_K = 1$ for the final loss. For the auxiliary losses $\{\xi_i\}_{i=0}^{K-1}$, in general, the losses at deeper layers should be more important, since the features at deeper layers often have better representation power. Thus, we use $\gamma_i = (\frac{L_i}{L_{K-1}})^{\nu}$ to reflect such difference, where $\nu > 0$ is the decaying rate of γ_i. In practice, we observe that if $\gamma_i < 0.01$, the effect of ξ_i becomes negligible. We thus use the following rule to adjust the weights for different losses:

$$\gamma_i = \max\left(0.01, \left(\frac{L_i}{L_{K-1}}\right)^{\nu}\right), \forall i \in \{0, \ldots, K-1\}. \tag{5}$$

In practice, we suggest setting $\nu \in [1/2, 2]$. In practice, similar to the adjustment of the learning rate, we may apply an adaptive

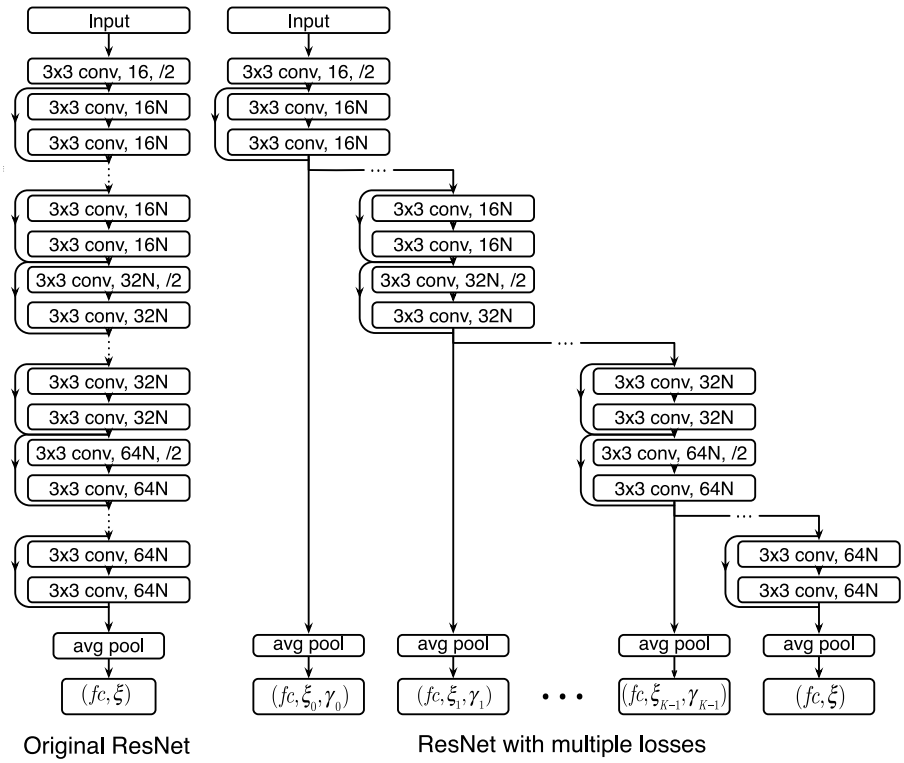


Fig. 1. Architecture with auxiliary losses. Taking ResNet for example, we put K additional losses $\{\xi_i\}_{i=0}^{K-1}$ evenly to the network, with each loss being built on top of an average pooling layer. Here, the K losses are weighted by $\{\gamma_i\}_{i=0}^{K-1}$, fc indicates the fully connected layer, ξ (also called ξ_K) denotes the final loss and N denotes the width of networks.

strategy to adjust ν during the training process (See discussions in Section 6.2).

4.1.2. Number of auxiliary outputs

There remains a question regarding how many auxiliary losses should be introduced. We observe that adding too many outputs would hamper the performance due to the conflicts of losses and also significantly increase the training complexity (See discussions of K in Section 6.3). Without loss of generality, given K auxiliary losses (*i.e.*, $\widehat{K}$=K+1 losses in total), we can introduce an auxiliary loss every $\tau = \lceil L/\widehat{K} \rceil$ layers, where $\tau \geq 5$.

4.2. Existing BP methods for multiple losses

Several BP methods have been proposed to train networks with auxiliary losses, *e.g.*, Joint BP (Lee et al., 2015; Szegedy et al., 2015) and Relay BP (Shen et al., 2016).

4.2.1. Joint BP

Joint BP considers minimizing a joint objective function of multiple losses (Lee et al., 2015; Szegedy et al., 2015):

$$\mathcal{L} = \sum_{i=0}^{K} \gamma_i \xi_i. \tag{6}$$

With the focus on the kth loss, the gradient of $\mathcal{L}$ w.r.t. $\mathbf{x}_l$ (where $L_{k-1} < l \leq L_k$) can be computed by

$$\frac{\partial \mathcal{L}}{\partial \mathbf{x}_l} = \sum_{i=k}^{K} \gamma_i \frac{\partial \xi_i}{\partial \mathbf{x}_{L_i}} \prod_{j=l}^{L_i-1} \mathbf{T}_j(\mathbf{W}_j). \tag{7}$$

From Eq. (7), Joint BP considers the information from all the losses by summing up the gradients. However, it has several limitations. First, the deep-layer losses (often with large weights) may dominate the gradients in Eq. (7) and the gradients from shallow-layer losses (often with very small weights) can be negligible. Thus, similar to the standard BP, the transformation $\prod_{j=l}^{L_i-1} \mathbf{T}_j(\mathbf{W}_j)$ for the deep-layer losses may cause information vanishing at intermediate layers (Shen et al., 2016), resulting in significant information loss in Eq. (7).

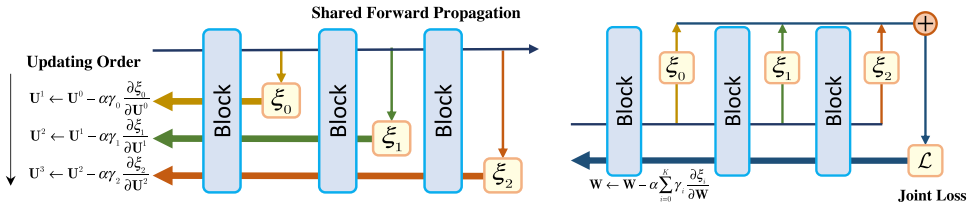
Second, due to the possible conflicts among losses, the gradients w.r.t. $\mathbf{x}_l$ from different losses may have different directions. As a result, the gradient in Eq. (7) can be inaccurate, which may incur training difficulties. To alleviate this, one should carefully adjust the weights γ_i for the auxiliary losses. For example, in DSN (Lee et al., 2015), the weights for auxiliary losses gradually decrease to zero during the training. However, decreasing the weights of auxiliary losses fails to fully exploit the auxiliary losses and thus hampers the overall performance (See results in Section 5.1).

4.2.2. Relay BP

To alleviate the possible information loss issue in Joint BP, Shen et al. proposed a Relay BP (Shen et al., 2016) method that discards the gradients propagated from deep-layer losses far away from a considered layer. With the focus on the kth loss, the gradient w.r.t. $\mathbf{x}_l$ (where $L_{k-1} < l \leq L_k$) becomes

$$\frac{\partial \mathcal{L}}{\partial \mathbf{x}_l} = \sum_{i=k}^{k+c} \gamma_i \frac{\partial \xi_i}{\partial \mathbf{x}_{L_i}} \prod_{j=l}^{L_i-1} \mathbf{T}_j(\mathbf{W}_j), \tag{8}$$

which means the lth layer only receives the gradients from $\{\xi_i\}_{i=k}^{k+c}$, where $c \geq 1$ is a constant. If $c \geq K-k$, Relay BP is reduced to Joint BP. When $c < K-k$, those gradients with long paths are discarded. However, without considering deep-layer losses, the shallow layers will tend to over-fit the nearby losses, which may deteriorate the representation power of the whole network (See results in Section 5.1).



(a) Multi-way Backpropagation (b) Joint Backpropagation

Fig. 2. A simple demonstration of Multi-way backpropagation (*left*) and Joint backpropagation (*right*) for training deep networks with auxiliary losses.

Algorithm 1: MW-BP for training deep networks.

Require: Model parameters: $\{\mathbf{W}_l\}_{l=0}^{L-1} = \{\mathbf{W}_0, \ldots, \mathbf{W}_{L-1}\}$;
Output positions: $\{L_i\}_{i=0}^{K} = \{L_0, \ldots, L_K\}$;
Weights for different losses $\{\gamma_i\}_{i=0}^{K} = \{\gamma_0, \ldots, \gamma_K\}$;
Input data: $\mathbf{x}_0$; Learning rate: α.

// **Shared Forward Propagation** for multiple outputs
for $l = 0$ to $L-1$ **do**
  Compute $\mathbf{x}_{l+1} = h(\lambda_l \mathbf{x}_l + \mathcal{F}_l(\mathbf{x}_l, \mathbf{W}_l))$;
  if $l \in \{L_i\}_{i=0}^{K}$ **then**
    Compute the loss ξ_i;
  end if
end for
// **Multi-way Backward Propagation**
Let $\mathbf{U}_l^0 \leftarrow \mathbf{W}_l$, $\forall l \in \{0, \ldots, L-1\}$;
for $i = 0$ to K **do**
  Conduct backpropagation w.r.t. ξ_i to compute $\{\frac{\partial \xi_i}{\partial \mathbf{U}_l^i}\}_{l=0}^{L_i}$;
  Update model parameters that are relevant to ξ_i by
    $\mathbf{U}_l^{i+1} \leftarrow \mathbf{U}_l^i - \alpha\gamma_i \frac{\partial \xi_i}{\partial \mathbf{U}_l^i}$, $\forall l \in \{0, \ldots, L_i\}$;
  Update model parameters that are irrelevant to ξ_i by
    $\mathbf{U}_l^{i+1} \leftarrow \mathbf{U}_l^i$, $\forall l \in \{L_i + 1, \ldots, L-1\}$;
end for
Let $\mathbf{W}_l \leftarrow \mathbf{U}_l^{K+1}$, $\forall l \in \{0, \ldots, L-1\}$;

4.3. Multi-way backpropagation

Using auxiliary losses is helpful for providing additional supervision information for intermediate layers. However, due to the possible conflicts among multiple losses, simply summing up the gradients propagated from multiple losses into a joint one (as done by Joint BP and Relay BP) may not achieve promising performance, as the conflicts are inherently ignored. Moreover, the supervision vanishing issue may still happen in Joint BP due to the long propagation path.

To address the above issues and well exploit the information from all the losses, we propose a simple yet effective method, called Multi-way BP (MW-BP), to train deep neural networks with multiple losses. The overall scheme is shown in Algorithm 1, which consists of **one shared forward propagation** and **multiple backward propagations** in each iteration. Note that, when performing the multiple backpropagations, we update the model parameters but keep the features and losses unchanged. As will be explained, this paradigm can effectively alleviate the possible conflicts among different losses and hence can address the supervision vanishing issue.

In Algorithm 1, similar to existing BP methods, in each iteration, we conduct a forward propagation to update the features and compute all the losses in $\{\xi_i\}_{i=0}^{K}$. However, when updating the model parameters, unlike Joint BP and Relay BP, we conduct multiple backpropagations (one for each loss in $\{\xi_i\}_{i=0}^{K}$) in a sequential manner. An illustrative comparison between Joint BP and MW-BP can be found in Fig. 2.

Taking the ith loss ξ_i for example, we compute the gradient of the lth layer ($\forall l \le L_i$) via backpropagation and update the model parameters via batch stochastic gradient descent (SGD). Let $\mathbf{x}_l$ be the input feature of the lth layer and $\mathbf{U}_l^i$ be the updated parameters after the $(i-1)$th backpropagation, with $\mathbf{U}_l^0$ being initialized by $\mathbf{W}_l$. For the ith loss ξ_i, we seek to update the parameters $\{\mathbf{U}_l^i\}_{l=0}^{L_i}$ that include both the updated parameters $\{\mathbf{U}_l^i\}_{l=0}^{L_{i-1}}$ by the $(i-1)$th loss and the unchanged parameters of the layers between L_{i-1} and L_i. Based on $\{\mathbf{U}_l^i\}_{l=0}^{L_i}$, we update the model parameters by

$$\mathbf{U}_l^{i+1} \leftarrow \mathbf{U}_l^i - \alpha \cdot \gamma_i \frac{\partial \xi_i}{\partial \mathbf{U}_l^i}, \tag{9}$$

where α denotes the learning rate and $\frac{\partial \xi_i}{\partial \mathbf{U}_l^i} = \frac{\partial \xi_i}{\partial \mathbf{x}_{l+1}} \frac{\partial \mathbf{x}_{l+1}}{\partial \mathbf{U}_l^i}$. By applying the chain rule, the gradient $\frac{\partial \xi_i}{\partial \mathbf{x}_l}$ for any layer l ($l \le L_i$) can be computed by

$$\frac{\partial \xi_i}{\partial \mathbf{x}_l} = \frac{\partial \xi_i}{\partial \mathbf{x}_{L_i}} \prod_{j=l}^{L_i - 1} \mathbf{T}_j(\mathbf{U}_j^i), \tag{10}$$

with $\mathbf{T}_j(\mathbf{U}_j^i) = \frac{\partial \mathbf{x}_{j+1}}{\partial \mathbf{y}_j}(\lambda_j \mathbf{I} + \partial_{\mathbf{x}_j} \mathcal{F}_j(\mathbf{x}_j, \mathbf{U}_j^i))$.

Remark 1. According to Eq. (9), we use $-\frac{\partial \xi_i}{\partial \mathbf{U}_l^i}$ as the search direction, which means the $(i+1)$th update is dependent on the ith update. As will be explained, this is very important for MW-BP (See Section 4.4.1). In fact, one may use $-\frac{\partial \xi_i}{\partial \mathbf{W}_l}$ as the search direction for the $(i+1)$th update, *i.e.*, each backpropagation is independent of each other. This strategy, however, is essentially the Joint BP method in Fig. 2(b).

Remark 2. Unlike existing BP methods in which the forward and backward propagations are often performed in pairs, in MW-BP, we apply a shared forward propagation for multiple updates, namely we do not update the features and losses after each backward propagation. As will be explained in Section 4.4.2, this is valid and also essential in boosting the performance and reducing the training complexity.

Differences from Joint BP. MW-BP is essentially different from Joint BP when computing gradients. As shown in Fig. 2, in Joint BP, since the gradient transformation of the backpropagation $\mathbf{T}(\mathbf{W})$ relies on the fixed model parameters $\mathbf{W}$, each backpropagation is independent of each other. Thus, conducting multiple backpropagations is equivalent to conducting a single backpropagation w.r.t. the joint loss. By contrast, in MW-BP, the gradient transformation of the ith backpropagation $\mathbf{T}(\mathbf{U}^i)$ depends on the updated parameters $\mathbf{U}^i$ by the $(i-1)$th backpropagation rather than $\mathbf{W}$ (See Fig. 2). Due to the dependence on previous updates, MW-BP gets different gradients from Joint BP and cannot be implemented by a single entire backpropagation.

4.4. Characteristics of MW-BP

Relying on the training paradigm in Algorithm 1, MW-BP has several characteristics over existing methods.

First, MW-BP can effectively address the supervision vanishing issue that occurred in the standard BP. In the standard BP, the long backpropagation path in Eq. (3) w.r.t. a single loss tends to incur the supervision vanishing issue. However, for MW-BP, we consider multiple backpropagations (one for each loss) to update model parameters in Eq. (9). Clearly, from Eq. (10), any intermediate layer l (especially the shallow layers) can receive sufficient supervision from their nearby losses (with $L_i \geq l$).

Second, MW-BP can also effectively avoid the loss conflict issue in Joint BP and Relay BP. As shown in Eqs. (7) and (8), Joint BP and Relay BP simply sum up the gradients of the related losses into a joint one. In this way, the conflicts among losses may affect the model performance (See Section 4.2 for details). Unlike these two methods, in MW-BP, we conduct the backpropagation for each loss. Thus, the summation process is avoided. As a result, the risk of loss conflict can be greatly reduced. However, to well address the supervision vanishing issue and the loss conflict issue, the importance of the **order of conducting backpropagations** and the **shared forward propagation** should be highlighted.

4.4.1. The order of conducting backpropagations

In MW-BP, we conduct multiple backpropagations from the loss ξ_0 to ξ_K in a sequential way. A primary reason is that a deeper loss, in general, is more important than a shallower loss (See Section 4.1 for details). Note that shallower models often have less representation power than deeper models. Thus, the attempt to fit shallower losses may introduce errors or distortions to the whole network (Lee et al., 2015). Fortunately, according to Eq. (9), the $(i+1)$th BP is built on the ith model update (See Remark 1). In this way, the errors brought by the model update w.r.t. a shallow loss can be corrected by the model update w.r.t. the deeper losses, which helps to obtain a good whole model and promising intermediate models. In other words, the order of backpropagations is essential for addressing the loss conflict issue.

4.4.2. The shared forward propagation

As stated in Remark 2, the shared forward propagation is one of the key features in MW-BP, which means that we do not update either features or losses after each backpropagation, even though a part of model parameters have been changed. In fact, upon the updating order in MW-BP, if updating the features and losses, the previous updates may highly affect the update w.r.t. deeper losses. For example, the deeper/final losses may decrease too quickly at the beginning epochs, which may incur gradient vanishing issue when updating deep layers and thus deteriorate the overall performance (See results and discussions in Section 5.1).

Moreover, as previously mentioned, the model update w.r.t. a shallow loss may bring in errors. However, if updating the features and losses, the corresponding forward propagation may propagate the errors to the deeper losses and hamper the correction effect of the model update w.r.t. them. Last, by avoiding multiple forward propagations, the shared forward propagation can significantly reduce the training complexity.

4.5. More discussions

To verify the above arguments and further analyze the proposed MW-BP method, in the following, we introduce several possible variants by considering the paired forward–backward propagation and/or different updating orders.

The first variant is referred to as **Naïve MW-BP**, in which a forward propagation is performed after each backpropagation. This method, however, may suffer from gradient vanishing issue since the deeper/final loss may decrease very quickly by updating the features and losses after each backpropagation. Therefore, the performance may be severely degraded (See Fig. 3 and discussions in Section 5.1). Moreover, multiple forward propagations will incur considerable training cost.

The second variant is referred to as **Reverse MW-BP**, in which we employ the same training paradigm of MW-BP but conduct multiple backpropagations in the reverse order of MW-BP (*i.e.*, from the last loss ξ_K to the first loss ξ_0). However, since the shallower models often have less representation power, the model update w.r.t. the shallower losses after the deeper losses may introduce representation errors into the whole network. Unlike this method, the errors incurred by shallower losses can be corrected by deeper losses in MW-BP that conducts backpropagations from the shallow losses to the deep ones.

The third variant is referred to as **Naïve Reverse MW-BP**. Based on Reverse MW-BP, we conduct a forward propagation after each backpropagation. However, in Naïve Reverse MW-BP, since the shallower losses will not affect the deeper/final losses, the gradient vanishing may not be as severe as Naïve MW-BP. Nevertheless, its performance is still limited since the model update w.r.t. shallower losses after deeper losses may bring in representation errors and hamper the overall performance. Moreover, the multiple forward propagations will incur considerable training cost.

4.6. Training and inference complexity

4.6.1. Training complexity

The training cost of MW-BP is approximately $(K/2+1)$ times of the standard BP method since it conducts one shared forward propagation and $(K+1)$ backpropagations at each iteration. If adding too many outputs to the model, it will greatly slow down the training process. In practice, introducing up to 4 auxiliary outputs is sufficient and is able to effectively improve the performance. Although MW-BP has larger training cost than the standard BP method, it simultaneously produces $(K+1)$ models. Therefore, the increased complexity is acceptable when considering the cost for model selection. Moreover, unlike the multiple forward propagations in Naïve MW-BP, the shared forward propagation can significantly reduce the computational cost (See more discussions in Section 6.4).

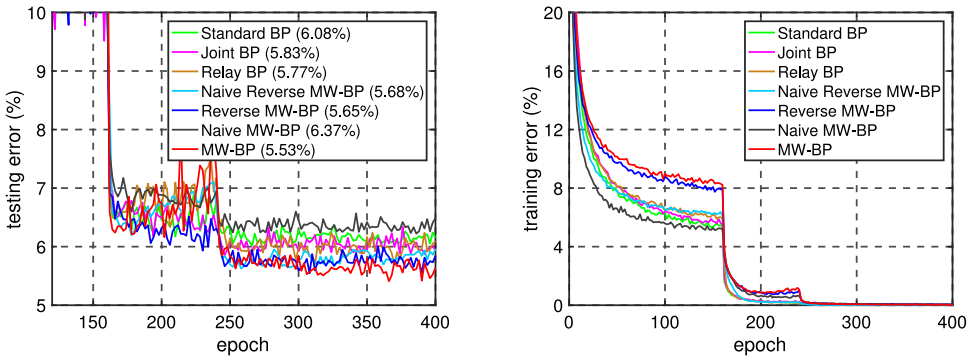
4.6.2. Inference complexity

During the inference, we do not need to consider the auxiliary losses. Moreover, MW-BP often produces very compact models (See results and discussions in Section 5.3). Thus, given the same architecture, the models trained by MW-BP have the same inference cost to the ones trained by existing BP methods, but often exhibit better prediction performance. In other words, under similar prediction performance, the inference cost for the models trained by MW-BP can be much lower than the ones trained with the standard BP method.

5. Experiments

To demonstrate the effectiveness of MW-BP, we apply the proposed MW-BP method to various architectures, including ResNet (He et al., 2016a), ResNeXt (Xie et al., 2017), DenseNet (Huang et al., 2017), Inception networks (Szegedy et al., 2017), and MobileNetV2 (Sandler et al., 2018). For convenience, we use "Mw+Architecture" to represent the model trained by MW-BP and "Architecture-L-$\widehat{K}$" to represent the model with L layers and $\widehat{K}$ outputs, *e.g.*, MwResNet-56-5. We conduct experiments on two tasks, namely **image classification** and **face recognition**. All implementations are based on PyTorch.[2]

[2] The source code of MW-BP and the pretrained models are available at https://github.com/tanmingkui/multiwaybp.



(a) Testing error of different BP methods. (b) Training error of different BP methods.

Fig. 3. Performance comparison of different BP methods on CIFAR-10. (a) Testing error of different BP methods on MwResNet-56-5. (b) Training error of different BP methods on MwResNet-56-5. All the curves come from the final outputs of deep models.

We organize the experiments as follows. First, we study and compare different BP methods in Section 5.1. Second, we extensively evaluate our MW-BP method on image classification tasks in Section 5.2. Third, we compare the resultant models obtained by MW-BP with the light models obtained by several compression methods in Section 5.3. Fourth, we evaluate our method on face recognition tasks in Section 5.4.

5.1. Comparison of various backpropagation methods

We compare MW-BP with 6 baseline BP methods, including the standard BP, Joint BP, Relay BP, Naïve MW-BP, Reverse MW-BP, and Naïve Reverse MW-BP. For a fair comparison, we apply the same weighting scheme of multiple losses to all the methods. Here, we first demonstrate the superiority of MW-BP over the standard BP. Then, we compare MW-BP with two existing BP methods that exploit auxiliary losses, namely Joint BP and Relay BP. Last, we compare MW-BP with its three variants.

5.1.1. Data sets and implementation details

We compare the performance of different BP methods on CIFAR-10 (Krizhevsky & Hinton, 2009). For Relay BP, we use the same setting in Shen et al. (2016) and set $c = 1$, *i.e.*, any intermediate layer only receives the gradients from the nearest two losses. For all the considered BP methods, we use SGD to train the models for 400 epochs with a mini-batch size of 128. We initially set the learning rate to 0.1 and divide it by 10 at 40% and 60% of the total epochs. In this experiment, we set the weighting scalar to $\nu = 2$ (See more discussions in Section 6.2).

5.1.2. Comparison with existing BP methods

We compare MW-BP with the standard BP and two existing BP methods that exploit auxiliary losses to train deep models. In this experiment, we use ResNet-56 as the baseline model. As for the BP methods trained with auxiliary losses, we evenly introduce 4 auxiliary losses at intermediate layers (at layer 15, 25, 35, 45, respectively). We compare the evolution of testing error and training error for different BP methods in Figs. 3(a) and 3(b), respectively.

From Fig. 3(a), since the intermediate layers can receive sufficient supervision from the nearby losses (See Section 4.3), MW-BP effectively addresses the supervision vanishing issue and yields significantly better performance than the model trained with the standard BP. Compared to Joint BP and Relay BP, the proposed MW-BP also greatly outperforms these methods and yields the best testing error of 5.53%. The main reason is that, unlike Joint BP and Relay BP, MW-BP conducts a backpropagation for each loss and does not sum up all the losses. In this way, MW-BP effectively avoids the loss conflict issue (See Section 4.4 for details). However, Joint BP and Relay BP sum up all the losses and would inevitably incur the loss conflict issue. As a result, the proposed MW-BP method is able to obtain significantly better results than these methods.

5.1.3. Comparison with MW-BP variants

We also compare MW-BP with its three variants to show the importance of the updating order of backpropagations and the shared forward propagation.

From Fig. 3(a), Naïve MW-BP yields the worst testing performance among all the considered methods. However, the training error decreases very quickly at the beginning epochs (See Fig. 3(b)). With the decreased training error/loss, the gradient vanishing issue can be very severe and hamper the performance. As a result, Naïve MW-BP yields severely degraded performance.

For Reverse MW-BP, when we reverse the updating order of MW-BP, it yields worse results than MW-BP. The main reason is that the model update w.r.t. the shallower losses after the deeper losses would introduce errors into the model (See discussions in Sections 4.4.1 and 4.5). As a result, the reverse updating order would hamper the overall performance (See Fig. 3(a)).

For Naïve Reverse MW-BP, it adopts the reverse updating order of backpropagations and updates the features and losses before each backpropagation. In this way, the shallow losses will not affect the deep losses and it significantly outperforms Naïve MW-BP (See Fig. 3(a)). However, unlike MW-BP, Naïve Reverse MW-BP with the reverse order cannot correct the errors incurred by shallow losses (See discussions in Section 4.5). As a result, Naïve Reverse MW-BP still yields slightly worse results than the proposed MW-BP method in Fig. 3.

5.2. Experiments on image classification

We apply the proposed MW-BP method to various architectures, including ResNet (He et al., 2016a), WideResNet (Zagoruyko & Komodakis, 2016), DenseNet (Huang et al., 2017), MobileNetV2 (Sandler et al., 2018) and Inception-ResNet (Szegedy et al., 2017). In this experiment, we evaluate our method on several image classification data sets.

5.2.1. Data sets and implementation details

We conduct comparisons on several benchmark data sets, including CIFAR-10 (Krizhevsky & Hinton, 2009), CIFAR-100 (Krizhevsky & Hinton, 2009), and ImageNet (Russakovsky et al., 2015). On CIFAR-10 and CIFAR-100, we perform SGD with a mini-batch size of 128 and train the model for 400 epochs. The learning

Table 1
Performance comparison on CIFAR-10 and CIFAR-100 data sets. Note that {·/N} indicates an N-fold increase in network width, *e.g.*, WideResNet-28-2/10 is 10 times wider than the baseline model. "–" denotes the results that are not reported.

Model	Error (%)					
	CIFAR-10			CIFAR-100		
ResNet-44 (He, Zhang, Ren, & Sun, 2016b)	6.37			28.85		
ResNet-56 (He et al., 2016b)	6.08			28.46		
ResNet-110 (He et al., 2016b)	5.86			27.41		
VGG-16 (Simonyan & Zisserman, 2015)	6.01			27.07		
DSN (Lee et al., 2015)	7.97			34.57		
GoogLeNet (Szegedy et al., 2015)	–			21.97		
MobileNetV2 (Sandler et al., 2018)	8.35			28.33		
WideResNet-28/10 (Zagoruyko & Komodakis, 2016)	4.17			20.50		
ResNeXt-29 (Xie et al., 2017)	4.25			21.02		
DenseNet-100 (Huang et al., 2017)	3.74			19.25		
Model-Depth-$\widehat{K}$	Joint BP	Relay BP	MW-BP	Joint BP	Relay BP	MW-BP
ResNet-44–2	6.21		**6.05**	28.03		**27.46**
ResNet-56–2	5.96		**5.77**	27.73		**26.83**
ResNet-44–3	6.07	6.03	**5.85**	27.86	27.77	**27.19**
ResNet-56–3	5.93	5.89	**5.68**	27.69	27.54	**26.77**
ResNet-56–5	5.83	5.77	**5.53**	27.37	27.33	**26.62**
ResNet-110–5	5.62	5.57	**5.41**	26.94	26.88	**26.48**
MobileNetV2–3	8.06	7.91	**7.63**	27.36	27.19	**26.77**
WideResNet-28-2/10	3.91	3.97	**3.77**	20.17	20.05	**19.69**
ResNeXt-29–3	3.98	3.84	**3.71**	19.23	19.17	**18.96**
DenseNet-100–4	3.65	3.61	**3.53**	19.21	19.24	**19.13**

rate starts from 0.1 and is divided by 10 at 40% and 60% of total epochs. On ImageNet, we use a mini-batch size of 256. For each model, we use the same number of epochs and the same learning rate strategy as the original paper. Specifically, we train ResNet models for 90 epochs and Inception models for 160 epochs. In all experiments, we empirically set the weighting scalar to $\nu = 2$ (See results and discussions in Section 6.2).

5.2.2. Comparison on CIFAR-10 and CIFAR-100

We conduct a comprehensive comparison between the proposed MW-BP method and existing BP methods based on various architectures, including ResNet (He et al., 2016a), Wide ResNet (Zagoruyko & Komodakis, 2016), ResNeXt (Xie et al., 2017), DenseNet (Huang et al., 2017), and MobileNetV2 (Sandler et al., 2018). We show the comparison results on CIFAR-10 and CIFAR-100 data sets in Table 1.

From Table 1, we have the following observations. First, the models trained by MW-BP significantly outperform the models trained by existing BP methods. For example, MwResNet-56-5 yields much better performance than the ResNet-56 baseline model trained by the standard BP and the ResNet-56-5 counterparts trained by Joint BP and Relay BP. Second, the proposed MW-BP is able to effectively reduce the internal model redundancy and produce compact models. To be specific, MwResNet-44 with 44 layers yields comparable or even better results than ResNet-110 with 110 layers on both CIFAR-10 and CIFAR-100 data sets. Third, when we introduce more auxiliary losses, we can further improve the performance. For example, MwResNet-56-5 with 5 losses yields better results than MwResNet-56-2 with 2 losses.

Besides the ResNet models, we also apply the proposed MW-BP to several state-of-the-art architectures, such as ResNeXt, DenseNet, and MobileNetV2. From Table 1, the resultant models trained by MW-BP consistently outperform the models trained by existing BP methods based on various architectures. For ResNeXt and DenseNet, MW-BP yields the best performance among all the considered BP methods. Even for a very compact model MobileNetV2, our MwMobileNetV2-3 also significantly outperforms the models trained by other BP methods. These results demonstrate that the proposed MW-BP method exhibits good compatibility with the considered deep architectures.

5.2.3. Comparison on ImageNet

We also evaluate the proposed method on a large-scale data set ImageNet (Krizhevsky et al., 2012). In this experiment, we apply the MW-BP method to several widely used models, *e.g.*, ResNet and Inception network. We show the comparison results in Table 2.

From Table 2, the proposed MW-BP method consistently outperforms existing BP methods based on various models. For example, MwResNet-18-2, MwResNet-34-2, and MwResNet-50-2 with two outputs yield better results than the models trained by the standard BP, Joint BP, and Relay BP. When adding more auxiliary losses, MW-BP is able to obtain better performance, *e.g.*, MwResNet-50-4. We further apply MW-BP on large models like ResNet-101 and Inception-ResNet. When we increase the number of losses up to 4, MwResNet-101-4 obtains a significant performance improvement of 0.8% in terms of Top-1 error. Moreover, equipped with MW-BP, MwInception-ResNet-4 yields the best performance among all the considered models in terms of both Top-1 error and Top-5 error. These results demonstrate the effectiveness of the proposed MW-BP method.

5.3. Comparison with light models

To demonstrate the superiority of our method in improving the compactness of deep models, we compare the resultant models obtained by MW-BP with the deeper models compressed by several channel pruning methods. In this sense, the compressed models have approximately the same computational complexity as the shallow models trained by MW-BP. We show the detailed comparison results in Tables 3 and 4.

For the considered channel pruning methods, we adopt ResNet-110 and ResNet-101 as the baseline models on CIFAR-10 and ImageNet, respectively. From Tables 3 and 4, most pruning methods yield similar or worse performance than the baseline models. Compared to the considered methods, the resultant model trained by MW-BP obtains the best or comparable results in terms of both accuracy and model compactness. For example, on CIFAR-10, MwResNet-56-5 with 56 layers and 5 outputs achieves a significant accuracy improvement of 0.33% compared to the baseline ResNet-110 and yields a great reduction of model size. On ImageNet, our MwResNet-50-4 yields the

Table 2
Comparison with different models on ImageNet in terms of validation error (10-crop).

Model	Top-1 error (%)			Top-5 error (%)		
VGG-16 (Simonyan & Zisserman, 2015)	28.07			9.33		
GoogLeNet (Szegedy et al., 2015)	–			9.15		
ResNet-18 (He et al., 2016a)	28.43			9.97		
ResNet-34 (He et al., 2016a)	24.76			7.35		
ResNet-50 (He et al., 2016a)	22.85			6.71		
ResNet-101 (He et al., 2016a)	21.75			6.05		
Inception-ResNet (Szegedy et al., 2017)	18.77			4.13		
Model-Depth-$\widehat{K}$	Joint BP	Relay BP	MW-BP	Joint BP	Relay BP	MW-BP
ResNet-18–2	28.13		**27.70**	9.93		**9.54**
ResNet-34–2	24.19		**23.76**	7.26		**7.03**
ResNet-50–2	22.78		**22.47**	6.65		**6.27**
ResNet-50–4	22.64	22.57	**22.15**	6.46	6.24	**6.07**
ResNet-101–4	21.54	21.43	**20.95**	5.71	5.97	**5.25**
Inception-ResNet-4	18.75	18.71	**18.61**	4.15	4.10	**4.05**

Table 3
Comparison between the resultant MwResNet models and the light models obtained by several state-of-the-art pruning methods on CIFAR-10. ResNet-110 is adopted as the baseline model.

Model		Baseline	PFEC (Li et al., 2017)	ThiNet (Luo et al., 2017)	CP (He et al., 2017)	SFP (He, Kang, Dong, Fu, & Yang, 2018)	PSFP (He, Dong, Kang, Fu, & Yang, 0000)	NISP (Yu et al., 2018)	MwResNet-56-5
ResNet-110 on CIFAR-10	#Params (M)	1.73	1.16	0.87	0.87	1.10	1.10	0.98	**0.85**
	#FLOPs (M)	253	155	127	127	150	150	143	**127**
	Error (%)	5.86	6.70	6.22	5.91	5.83	6.06	6.04	**5.53**

Table 4
Comparison between the resultant MwResNet models and the light models obtained by several state-of-the-art pruning methods on ImageNet (10-crop). ResNet-101 is adopted as the baseline model.

Model		Baseline	Rethinking (Ye, Lu, Lin, & Wang, 2018)	Taylor-FO-BN (Molchanov, Mallya, Tyree, Frosio, & Kautz, 2019)	SFP (He et al., 2018)	PSFP (He et al., 0000)	MwResNet-50-4
ResNet-101 on ImageNet	#Params (M)	44.55	**22.67**	26.75	25.23	25.23	25.55
	#FLOPs (M)	7260	3847	4790	4159	4159	**3530**
	Top-1 error (%)	21.75	23.85	23.74	22.49	22.72	**22.15**

best performance with the smallest accuracy drop compared to the baseline ResNet-101. These results show that the proposed MW-BP method effectively is able to reduce the internal model redundancy and thus produces very compact models. It is worth noting that, unlike most model compression methods, our MW-BP method is able to train the compact models from scratch and does not rely on pre-trained models.

5.4. Experiments on face recognition

In this experiment, we further apply MW-BP to a face recognition model MobileFaceNet (Chen et al., 2018). We add 2 auxiliary losses at the intermediate layers. For convenience, we term it MwMobileFaceNet-3 (*i.e.*, containing 3 losses in total). We compare the resultant models with the baseline model trained by the standard BP on several face recognition data sets.

5.4.1. Data sets and implementation details

We adopt the large-scale data set MS1M (Guo, Zhang, Hu, He, & Gao, 2016) as the training data and four benchmark data sets as the validation data, including LFW (Huang, Mattar, Berg, & Learned-Miller, 2008), CFP-FP (Sengupta et al., 2016), AgeDB-30 (Moschoglou et al., 2017), and MegaFace (Kemelmacher-Shlizerman, Seitz, Miller, & Brossard, 2016). We use the same setting as that in MobileFaceNet (Chen et al., 2018). Specifically, all face images are preprocessed to the size of 112×112. We set the momentum and weight decay to 0.9 and 4×10^{-5}, respectively. We train the models for 36 epochs with a mini-batch size of 200. The learning rate starts from 0.1 and is divided by 10 at the {15, 25, 31}th epoch, respectively. We set the weighting scalar $\nu = 2$ in the face recognition experiments.

5.4.2. Performance comparison

In this experiment, we compare the models trained with and without MW-BP in terms of the evolution of validation error. Fig. 4 shows the comparison on 3 data sets, including LFW, CFP-FP, and AgeDB-30. From Fig. 4, the proposed MW-BP method greatly accelerates the convergence and yields significantly better performance than the standard BP method.

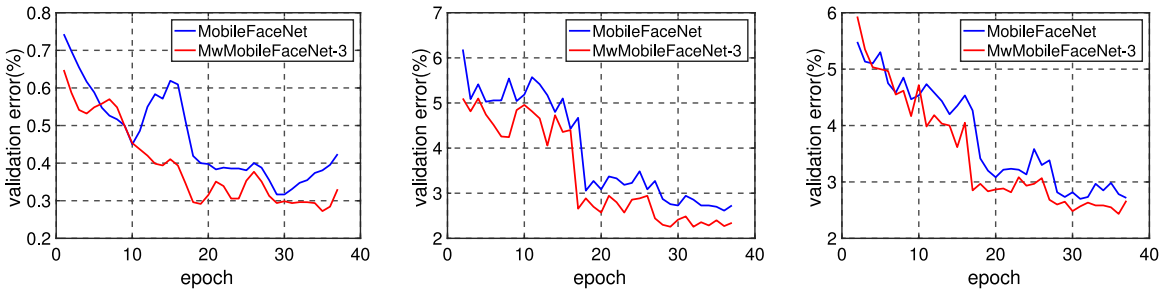
Besides the aforementioned 3 data sets, we also evaluate the models on a large-scale data set MegaFace and show more detailed results in Table 5. From Table 5, our MwMobileFaceNet-3 consistently outperforms the considered baseline models on 4 data sets. These results demonstrate the effectiveness of the proposed MW-BP method on face recognition models.

6. Further experiments

In this section, we investigate the prediction ability of intermediate models obtained by MW-BP and conduct ablation studies for the proposed method.

6.1. Prediction ability of intermediate models

In this section, we investigate the prediction ability of the intermediate models obtained during the training process of MW-BP. In Table 6, we compare the performance of intermediate



(a) Validation error on LFW. (b) Validation error on CFP-FP. (c) Validation error on AgeDB-30.

Fig. 4. Performance comparison of the MobileFaceNet models trained with and without MW-BP on 3 benchmark face recognition data sets.

Table 5
Performance comparison of different methods for face recognition. "VR" refers to Verification TAR (True Accepted Rate) and "FAR10^{-6}" refers to the False Accepted Rate at 10^{-6}. "–" denotes the results that are not reported.

Model	Validation Accuracy (%)			VR@FAR10^{-6} (%)
	LFW	CFP-FP	AgeDB-30	MegaFace
SphereFace (Liu et al., 2017)	99.42	–	–	85.56
CosFace (Wang et al., 2018)	99.33	–	–	89.88
MobileFaceNet (Chen et al., 2018)	99.67	97.30	97.02	93.57
MwMobileFaceNet-3	**99.72**	**97.60**	**97.45**	**93.94**

Table 6
Testing error of the intermediate models obtained by different BP methods on CIFAR-10 and CIFAR-100 data sets.

Model	#Layers	#Params	CIFAR-10 error (%)				CIFAR-100 error (%)			
			Standard BP	Joint BP	Relay BP	MW-BP	Standard BP	Joint BP	Relay BP	MW-BP
Model-15	15	0.03M	63.01	61.37	58.93	**50.35**	87.74	84.51	79.85	**71.73**
Model-25	25	0.09M	45.07	40.11	39.47	**18.94**	68.17	63.88	60.37	**51.21**
Model-35	35	0.18M	34.01	28.92	27.64	**9.23**	49.54	43.17	41.09	**35.88**
Model-45	45	0.48M	13.71	11.56	10.21	**5.67**	35.72	31.63	30.44	**27.35**
Model-56	56	0.85M	6.08	5.83	5.77	**5.53**	28.46	27.37	27.33	**26.62**

Table 7
Effect of the weighting scalar ν on CIFAR-10. MwResNet-56-5 is adopted as the baseline model.

Model	ν	Error (%)				
		Model-15	Model-25	Model-35	Model-45	Model-56
MwResNet-56-5 (ResNet-56 6.08%)	0	57.63	25.30	15.94	11.89	8.43
	1	49.83	18.77	9.71	5.97	5.90
	2	50.35	18.94	9.23	5.67	**5.53**
	5	52.46	19.97	10.93	6.15	6.03
	Adaptive-I	**41.47**	**16.05**	**8.65**	6.19	6.08
	Adaptive-II	44.89	16.98	9.07	**5.64**	5.61

Table 8
Effect of the number of losses on the performance of MwResNet-56 on CIFAR-10 data set.

Model	#Losses	Error (%)
ResNet-56 (He et al., 2016b)	1	6.08
MwResNet-56	2	5.77
	5	**5.53**
	10	7.36
	25	9.18

models generated by each loss of MwResNet-56-5 against the models trained by the standard BP,[3] Joint BP, and Relay BP.

From Table 6, each intermediate model of MwResNet-56-5 consistently outperforms its competitors (of the same depth) obtained by the standard BP, Joint BP, and Relay BP. By comparing these results with the results in Table 1, the intermediate model **model-45** even outperforms very deep models. For example, compared to ResNet-110 with 1.7M parameters (5.86% error on CIFAR-10 and 27.41% error on CIFAR-100), our **model-45** with 0.85M parameters yields better performance on both CIFAR-10 (5.67% error) and CIFAR-100 (27.35% error). These results demonstrate that the proposed method not only improves the representation ability of intermediate layers, but also provides the opportunity for a form of model selection.

6.2. Effect of the weighting scalar ν

We investigate the effect of the weighting scheme. Two kinds of weighting schemes are considered. First, we can set ν to a constant value to adjust the weights of different losses. Second, we also consider two adaptive strategies to dynamically increase or decrease the weights during the training, namely *Adaptive-I* and *Adaptive-II*. In *Adaptive-I*, we initially set $\nu = 1/2$ and multiply it by 2 when we change the learning rate. Just opposite to *Adaptive-I*, in *Adaptive-II*, we initially set $\nu = 2$ and divide it by

[3] To obtain the intermediate models of ResNet-56, We fix the parameters of all layers and only train the outputs added to the intermediate layers.

Table 9
Comparisons between the Naïve MW-BP with multiple forward propagations and MW-BP with the shared forward propagation based on MWResNet-56-5 on CIFAR-10. We measure the inference time for both methods using 128 images.

Method	Forward scheme	Inference time (ms)	MAdds (M)	Error (%)
Naïve MW-BP	multiple	51.78	415.88	6.37
MW-BP	shared	**17.10**	**125.75**	**5.53**

2 along with the change of learning rate. Based on MwResNet-56-5, we compare the performance of the models trained with different weighting strategies in Table 7.

We first compare the effect of ν with different constant values. When we set $\nu = 0$, all the losses are equally weighted. However, since multiple losses are not equally important (See Section 4.1.1), the equally weighted losses severely hamper the performance of MW-BP in Table 7. However, when we choose a large value of $\nu = 5$, the weights would decay so aggressively that the effects of auxiliary outputs are negligible. To avoid this issue, we empirically choose $\nu = 2$ and this setting yields the best performance in practice.

For the two adaptive strategies, from Table 7, they yield slightly worse results than the best setting of $\nu = 2$ at the final output. However, they significantly improve the performance of intermediate models. Therefore, we suggest that one can use the adaptive strategies to obtain better intermediate models.

6.3. Effect of the numbers of losses

We investigate the effect of the number of losses. We take ResNet-56 for example and insert different numbers of losses. From the results in Table 8, MwResNet-56-2 and MwResNet-56-5 perform significantly better than ResNet-56. However, adding too many losses does not necessarily improve the performance. For example, the MwResNet-56 models with 10 and 25 outputs yield severely degraded performance. The main reason is that the interval between losses is too small so that they may affect each other and eventually degrade the performance. Moreover, adding too many losses will also slow down the training. In practice, introducing up to 5 losses is sufficient to effectively improve the performance according to previous experimental results.

6.4. Effect of the shard forward propagation

In this section, we investigate the effect of the shared forward propagation in terms of both the computational cost and the learning performance. Specifically, we compare the shard forward propagation in MW-BP and the multiple forward propagations in Naïve MW-BP. Unlike Naïve MW-BP, the shared forward propagation in MW-BP only performs single forward propagation and thus greatly reduces the computational cost. To verify this, we take MwResNet-56-5 with 5 losses as an example. As shown in Table 9, MW-BP takes significantly lower cost in terms of inference time and the number of multiply-adds (MAdds) and yields better results than Naïve MW-BP. These results demonstrate the effectness of the proposed shared forward propagation.

7. Conclusion

In this paper, we have investigated the supervision vanishing issue in existing backpropagation (BP) methods for training deep networks. When the network is very deep, shallow layers tend to receive insufficient supervision due to the severe transformation through long backpropagation path, resulting in severe internal model redundancy. To address these issues, we introduced auxiliary losses into deep models and proposed an effective training method, called Multi-way BP (MW-BP). Based on various architectures, our method consistently obtains significant performance improvement and produces more compact models than existing BP methods. More critically, with approximately the same model complexity, the resultant models also outperform the light models obtained by state-of-the-art model compression methods. Extensive experiments on both image classification and face recognition tasks demonstrate the effectiveness of the proposed method.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This work was partially supported by Guangdong Provincial Scientific and Technological Funds under Grants 2018B010107001 and 2019B010155002, National Natural Science Foundation of China (NSFC) 61836003 (key project), Fundamental Research Funds for the Central Universities D2191240, Program for Guangdong Introducing Innovative and Enterpreneurial Teams 2017ZT07X183, Tencent AI Lab Rhino-Bird Focused Research Program JR201902, Guangdong Special Branch Plans Young Talent with Scientific and Technological Innovation 2016TQ03X445, Guangzhou Science and Technology Planning Project 201904010197, and Microsoft Research Asia (MSRA Collaborative Research Program).

References

Ba, J., & Caruana, R. (2014). Do deep nets really need to be deep? In *NeurIPS* (pp. 2654–2662).
Cao, J., Guo, Y., Wu, Q., Shen, C., Huang, J., & Tan, M. (2018). Adversarial learning with local coordinate coding. In *ICML*.
Chen, S., Liu, Y., Gao, X., & Han, Z. (2018). Mobilefacenets: Efficient cnns for accurate real-time face verification on mobile devices. In *CCBR* (pp. 428–438).
Drucker, H., & Le Cun, Y. (1992). Improving generalization performance using double backpropagation. *IEEE Transactions on Neural Networks*, *3*(6), 991–997.
Gonzalez-Dominguez, J., Lopez-Moreno, I., Moreno, P. J., & Gonzalez-Rodriguez, J. (2015). Frame-by-frame language identification in short utterances using deep neural networks. *Neural Networks*, *64*, 49–58.
Guo, Y., Chen, Q., Chen, J., Wu, Q., Shi, Q., & Tan, M. (2019). Auto-embedding generative adversarial networks for high resolution image synthesis. *TMM*, *21*(11), 2726–2737.
Guo, Y., Wu, Q., Deng, C., Chen, J., & Tan, M. (2018). Double forward propagation for memorized batch normalization. In *AAAI* (pp. 3134–3141).
Guo, Y., Zhang, L., Hu, Y., He, X., & Gao, J. (2016). Ms-celeb-1m: A dataset and benchmark for large-scale face recognition. In *ECCV* (pp. 87–102).
Guo, Y., Zheng, Y., Tan, M., Chen, Q., Chen, J., Zhao, P., et al. (2019). Nat: Neural architecture transformer for accurate and compact architectures. In *NeurIPS* (pp. 735–747).
He, Y., Dong, X., Kang, G., Fu, Y., & Yang, Y. Progressive deep neural networks acceleration via soft filter pruning, arXiv preprint abs/1808.07471.
He, Y., Kang, G., Dong, X., Fu, Y., & Yang, Y. (2018). Soft filter pruning for accelerating deep convolutional neural networks. In *IJCAI* (pp. 2234–2240).
He, Y., Liu, P., Wang, Z., Hu, Z., & Yang, Y. (2019). Filter pruning via geometric median for deep convolutional neural networks acceleration. In *CVPR* (pp. 4340–4349).
He, K., Zhang, X., Ren, S., & Sun, J. (2016a). Deep residual learning for image recognition. In *CVPR* (pp. 770–778).

He, K., Zhang, X., Ren, S., & Sun, J. (2016b). Identity mappings in deep residual networks. In *ECCV* (pp. 630–645).
He, Y., Zhang, X., & Sun, J. (2017). Channel pruning for accelerating very deep neural networks. In *ICCV* (pp. 1398–1406).
Howard, A. G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., et al. Mobilenets: Efficient convolutional neural networks for mobile vision applications, arXiv preprint abs/1704.04861.
Huang, G., Liu, Z., Van Der Maaten, L., & Weinberger, K. Q. (2017). Densely connected convolutional networks. In *CVPR* (pp. 4700–4708).
Huang, G. B., Mattar, M., Berg, T., & Learned-Miller, E. (2008). Labeled faces in the wild: A database forstudying face recognition in unconstrained environments.
Ibtehaz, N., & Rahman, M. S. (2020). Multiresunet: rethinking the u-net architecture for multimodal biomedical image segmentation. *Neural Networks*, *121*, 74–87.
Kemelmacher-Shlizerman, I., Seitz, S. M., Miller, D., & Brossard, E. (2016). The megaface benchmark: 1 million faces for recognition at scale. In *CVPR* (pp. 4873–4882).
Krizhevsky, A., & Hinton, G. (2009). Learning multiple layers of features from tiny images.
Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In *NeurIPS* (pp. 1097–1105).
LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *Nature*, *521*(7553), 436–444.
LeCun, Y., Boser, B., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W., et al. (1989). Backpropagation applied to handwritten zip code recognition. *Neural Computation*, *1*(4), 541–551.
Lee, C.-Y., Xie, S., Gallagher, P., Zhang, Z., & Tu, Z. (2015). Deeply-supervised nets. In *AISTATS*.
Li, H., Kadav, A., Durdanovic, I., Samet, H., & Graf, H. P. (2017). Pruning filters for efficient convnets. In *ICLR*.
Liu, W., Wen, Y., Yu, Z., Li, M., Raj, B., & Song, L. (2017). Sphereface: Deep hypersphere embedding for face recognition. In *CVPR* (pp. 212–220).
Liu, J., Zhuang, B., Zhuang, Z., Guo, Y., Huang, J., Zhu, J., et al. Discrimination-aware network pruning for deep model compression, arXiv preprint arXiv: 2001.01050.
Luo, J.-H., Wu, J., & Lin, W. (2017). Thinet: A filter level pruning method for deep neural network compression. In *ICCV* (pp. 5068–5076).
Molchanov, P., Mallya, A., Tyree, S., Frosio, I., & Kautz, J. (2019). Importance estimation for neural network pruning. In *CVPR* (pp. 11264–11272).
Moschoglou, S., Papaioannou, A., Sagonas, C., Deng, J., Kotsia, I., & Zafeiriou, S. (2017). Agedb: the first manually collected, in-the-wild age database. In *CVPR workshops* (pp. 51–59).
Nair, V., & Hinton, G. E. (2010). Rectified linear units improve restricted Boltzmann machines. In *ICML* (pp. 807–814).
Pascanu, R., Mikolov, T., & Bengio, Y. Understanding the exploding gradient problem, CoRR, abs/1211.50632.
Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., et al. (2015). Imagenet large scale visual recognition challenge. *IJCV*, *115*(3), 211–252.
Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., & Chen, L.-C. (2018). Mobilenetv2: Inverted residuals and linear bottlenecks. In *CVPR* (pp. 4510–4520).
Sengupta, S., Chen, J.-C., Castillo, C., Patel, V. M., Chellappa, R., & Jacobs, D. W. (2016). Frontal to profile face verification in the wild. In *WACV* (pp. 1–9).
Shen, L., Lin, Z., & Huang, Q. (2016). Relay backpropagation for effective learning of deep convolutional neural networks. In *ECCV* (pp. 467–482).
Simonyan, K., & Zisserman, A. (2015). Very deep convolutional networks for large-scale image recognition. In *ICLR*.
Srivastava, R. K., Greff, K., & Schmidhuber, J. (2015). Training very deep networks. In *NeurIPS* (pp. 2377–2385).
Szegedy, C., Ioffe, S., Vanhoucke, V., & Alemi, A. A. (2017). Inception-v4, inception-resnet and the impact of residual connections on learning. In *AAAI* (pp. 4278–4284).
Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., et al. (2015). Going deeper with convolutions. In *CVPR* (pp. 1–9).
Tan, M., Chen, B., Pang, R., Vasudevan, V., Sandler, M., Howard, A., et al. (2019). Mnasnet: Platform-aware neural architecture search for mobile. In *CVPR* (pp. 2820–2828).
Wang, H., Dai, L., Cai, Y., Sun, X., & Chen, L. (2018). Salient object detection based on multi-scale contrast. *Neural Networks*, *101*, 47–56.
Wang, H., Wang, Y., Zhou, Z., Ji, X., Gong, D., Zhou, J., et al. (2018). Cosface: Large margin cosine loss for deep face recognition. In *CVPR*.
Wilson, D. R., & Martinez, T. R. (2003). The general inefficiency of batch training for gradient descent learning. *Neural Networks*, *16*(10), 1429.
Xie, S., Girshick, R., Dollár, P., Tu, Z., & He, K. (2017). Aggregated residual transformations for deep neural networks. In *CVPR* (pp. 1492–1500).
Ye, J., Lu, X., Lin, Z., & Wang, J. Z. (2018). Rethinking the smaller-norm-less-informative assumption in channel pruning of convolution layers. In *ICLR*.
Yu, R., Li, A., Chen, C.-F., Lai, J.-H., Morariu, V. I., Han, X., et al. (2018). Nisp: Pruning networks using neuron importance score propagation. In *CVPR* (pp. 9194–9203).
Zagoruyko, S., & Komodakis, N. (2016). Wide residual networks. In *BMVC*.
Zhang, X., Zhou, X., Lin, M., & Sun, J. (2018). Shufflenet: An extremely efficient convolutional neural network for mobile devices. In *CVPR* (pp. 6848–6856).
Zhuang, Z., Tan, M., Zhuang, B., Liu, J., Guo, Y., Wu, Q., et al. (2018). Discrimination-aware channel pruning for deep neural networks. In *NeurIPS* (pp. 875–886).