The Shallow End: Empowering Shallower Deep-Convolutional Networks through Auxiliary Outputs

Yong Guo[1] Mingkui Tan[1] Qingyao Wu[1] Jian Chen[1] Anton Van Den Hengel[2] Qinfeng Shi[2]
[1]South China University of Technology, [2]The University of Adelaide
{guoyongcs, tanmingkui}@gmail.com, {qyw, ellachen}@scut.edu.cn
{anton.vandenhengel, javen.shi}@adelaide.edu.au

Abstract

Convolutional neural networks (CNNs) with very deep architectures, such as the residual network (ResNet) [6], have shown encouraging results in various tasks in computer vision and machine learning. Their depth has been one of the key factors behind the great success of CNNs, with the gradient vanishing issue having been largely addressed by ResNet. However, there are other issues associated with increased depth. First, when networks get very deep, the supervision information may vanish due to the associated long backpropagation path. This means that intermediate layers receive less training information, which results in redundancy in models. Second, when the model becomes more complex and redundant, inference becomes more expensive. Third, very deep models require larger volumes of training data. We propose here instead an AuxNet and a new training method to propagate not only gradients but also supervision information from multiple auxiliary outputs at intermediate layers. The proposed AuxNet gives rise to a more compact network which outperforms its very deep equivalent (i.e. ResNet). For example, AuxNet with 44 layers performs better than the original ResNet with 110 layers on several benchmark data sets, i.e. CIFAR-10, CIFAR-100 and SVHN.

1. Introduction

Since 2012 when AlexNet won the first place in the ImageNet competition [10], convolutional neural networks (CNNs) [12], have been producing state-of-the-art results in many of the most challenging vision tasks including image classification [10, 13, 6], face recognition [21, 26], semantic segmentation [14] and object detection [18, 32]. CNNs have become the workhorse of many learning tasks and real-world applications beyond computer vision, such as natural language understanding and speech recognition [11].

Recent studies have demonstrated both theoretically [15, 2] and empirically [22, 24, 27, 13] that the depth of neural networks is crucial for their representation ability with limited units [24]. However, in practice, a deeper network often does not necessarily lead to better performance mainly due to the implied training difficulties. It is shown in [6] that a plain network with large depth often *under-fits*, instead of over-fitting, the training data, leading to a poor test accuracy. More critically, when the number of layers L is very large, the network performance may be severely degraded, largely due to the training difficulty caused by gradient vanishing or exploding [3, 19, 23, 6].

Recently, He *et al.* proposed a new network architecture called the deep residual network (ResNet) [6], which conducts forward inference by

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

with $\mathbf{x}_0$ being the input data, h being the nonlinear activation function (e.g., the Rectified Linear Unit (ReLU): $h(z) = \max(0, z)$ [16]), and $\mathcal{F}$ being the so called residual, a transformation function (e.g., the convolution operation [19]) parameterized by $\boldsymbol{\mathcal{W}}_l$. The term $\lambda_l \mathbf{x}_l$ in Eqn. (1) introduces what are called *shortcut connections*, and the model is reduced to a plain network when $\lambda_l = 0$. The shortcut connections enable the features of any layer l to be propagated to deeper layers, which helps to address the gradient vanishing issue. In fact, assuming $\lambda_l = 1$ and $h(\mathbf{x}) = \mathbf{x}$ (i.e., $h(\mathbf{x})$ is an identity activation function), and applying the recursive rule in Eqn.(1), we achieve a simplified relation: $\mathbf{x}_L = \mathbf{x}_l + \sum_{i=l}^{L-1} \mathcal{F}(\mathbf{x}_i, \boldsymbol{\mathcal{W}}_i)$ [7].

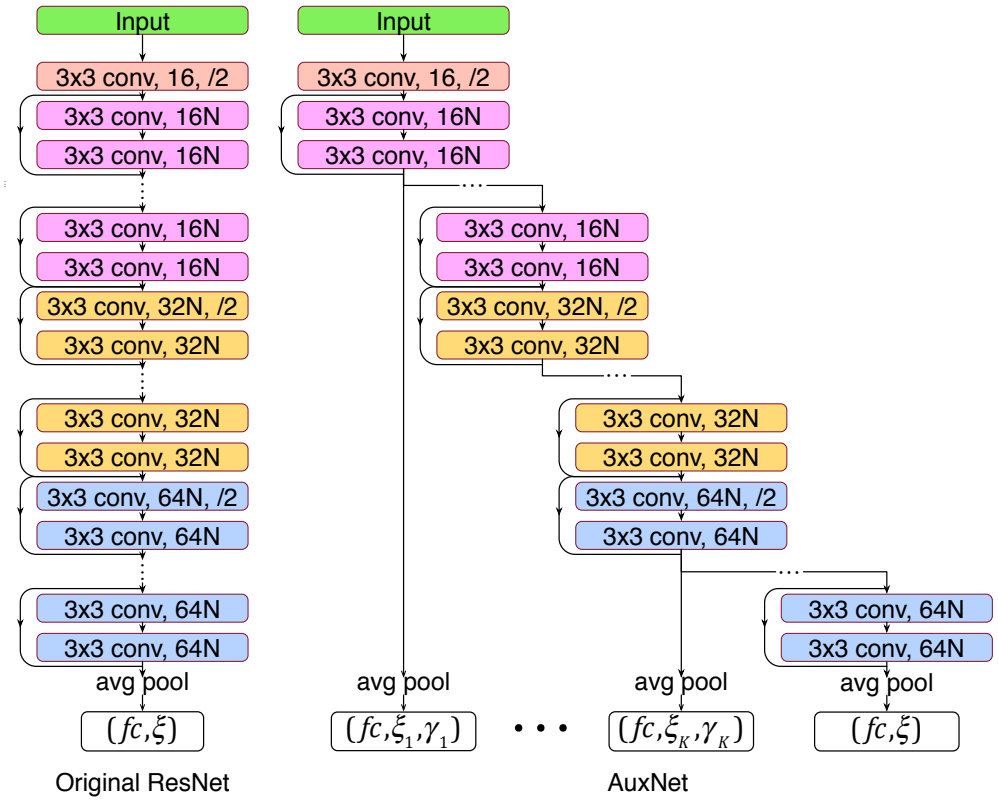


Figure 1. Architecture of the proposed AuxNet. fc indicates a fully connected layer, $\{\xi_k\}_{k=1}^K$ denotes softmax losses of auxiliary outputs, ξ denotes softmax loss of the final output, N denotes the width of network and $\{\gamma_k\}_{k=1}^K$ denotes the weights on the auxiliary cost functions.

Letting ξ be the loss of the final output, then for any shallow layer l, the gradient of ξ w.r.t. $\mathbf{x}_l$ can be written as

$$\partial_{\mathbf{x}_l}\xi = \partial_{\mathbf{x}_L}\xi + \partial_{\mathbf{x}_L}\xi \cdot \left(\sum_{i=l}^{L-1} \partial_{\mathbf{x}_l}\mathcal{F}(\mathbf{x}_i, \boldsymbol{\mathcal{W}}_i)\right). \tag{2}$$

According to Eqn.(2), the gradient of a layer $\partial_{\mathbf{x}_l}\xi$ is unlikely to vanish even when the weights are arbitrarily small, which makes the training of very deep networks with hundreds or thousands of layers possible [7]. With such depth, ResNet has achieved state-of-the-art accuracy for several challenging tasks on ImageNet [20] and MS COCO competitions [15]. However, although gradient is maintained, the supervision signal may none-the-less vanish for very deep networks. In Eqn.(2), the term $\partial_{\mathbf{x}_L}\xi$ reflects the supervision information obtained from the loss; while $(\sum_{i=l}^{L-1} \partial_{\mathbf{x}_l}\mathcal{F}(\mathbf{x}_i, \boldsymbol{\mathcal{W}}_i))$ indicates the transformation of the network over $\partial_{\mathbf{x}_L}\xi$. Given a large L, $\sum_{i=l}^{L-1} \partial_{\mathbf{x}_l}\mathcal{F}(\mathbf{x}_i, \boldsymbol{\mathcal{W}}_i)$ for a small l can be very large and hence may dominate the resulting gradient. The supervision information reaching the shallower layers may thus not be sufficient to drive the desired behavior. This may result in difficulties in training the intermediate layers, and lead to model redundancy as network depth increases.

Very deep CNNs can also have other limitations in practice. First, given a very large number of layers, the model becomes more complex and the generalization ability may also suffer [6, 8]. Especially, when there is insufficient training data available, over-fitting is inevitable [6]. Second, the number of model parameters increases dramatically, with depth, however, leading to a dramatic increase in inference cost. In real-world applications, there is a demand for more computationally efficient models, particularly for models capable of fast inference [1]. Third, when the network is sufficiently deep, adding even a large number of layers effects only a marginal improvement in performance, which implies that it would be promising to train a compact network to achieve the same performance as the very deep model.

In this paper, we propose to address the above issues by devising a so called AuxNet in which adaptively weighted auxiliary losses (or outputs) are introduced (see Figure 1 and Section 2 for more details). Specifically, rooted in the ResNet, each loss is connected to an intermediate layer in order to reduce model redundancy and improve the discriminative power

of hidden layers. In this way, we can learn a much more compact model to represent images. While adding auxiliary outputs has been investigated by some previous work [13, 27, 21], we draw the following new conclusions in our study.

1. We propose three methods to train the network with multiple losses. In particular, we develop a multi-way training method which applies one forward propagation for all losses but conducts the gradient backpropagation for each loss separately and in series. This method effectively avoids the issue of supervision information vanishing, and yields the best result in terms of generalization performance.

2. By providing sufficient supervision information from the nearby loss for intermediate layers, the proposed AuxNet reduces the internal model redundancy and gives rise to a more compact model (i.e., with fewer layers but better performance).

3. The proposed approach inherently produces multiple models of different depths, making model selection very simple. Surprisingly, the intermediate models often outperform much deeper models, including the full-depth ResNet equivalent.

2. Related Studies

Employing auxiliary classifiers has been investigated in DeepID models for face recognition [26, 25], GoogLeNet [27], and deep supervised nets (DSN) [13, 30]. In [26], the supervisory signals are connected to each convolution layer; while in [25] they are added to each pooling layer.

In GoogLeNet, two auxiliary classifiers (with weights equal to 0.3) are connected to intermediate layers to address the problem of vanishing gradients. In DSN [13], each convolution layer, followed by a *mlpconv* layer, is associated with a classifier. In the training, a joint objective function is constructed. To avoid the over-fitting issue, they keep the losses for a number of epochs, and discard all but the final loss to finish the rest epochs. These networks cannot be very deep (less than 25 layers). Nevertheless, even with auxiliary classifiers, the issue of supervision information vanishing can still happen for these methods. Moreover, unlike DSN, our method does not discard any loss, which helps to generate multiple models at the same time, with the ensuing benefits for model selection. In BranchyNet [28], auxiliary outputs are regarded as side branches for early exit. These outputs, not for training, but allow test samples to exit from shallow layers to achieve fast inference.

3. Auxiliary Residual Networks

We see that the supervision information vanishing of long path propagation can cause model redundancy in even well trained very deep networks. We thus seek to add auxiliary outputs, (each associated with a loss function), to some intermediate layers to improve the discriminative power of all the shallower layers. Given its heritage, we have labelled the model AuxNet.

3.1. General Architecture of AuxNet

Rooted in ResNet, the general architecture of AuxNet is shown in Figure 1. The original ResNet is composed of a set of residual blocks, each containing two or more convolutional layers, followed by a batch normalization layer and an activation function (see [6] for more details). Very recently, He *et al.* have stated that the pre-activation function could further improve the ResNet and make the training of 1000-layers model possible [7]. Compared with the ResNet, one may extend the width of network by a factor of N to improve the representation power of each block (see wide ResNets [31]). In AuxNet, in addition to the final output, we introduce additional K losses weighted by γ_k to the intermediate residual blocks. Thus it contains $K+1$ outputs in total, each for the same classification task. For convenience, hereafter we use AuxNet-56-5 to denote AuxNet with 56 layers and 5 outputs (4 auxiliary outputs).

The positions of the auxiliary outputs are important. In general, very shallow layers represent low-level structures with little discriminative power even for a well trained shallow model [33]. Therefore, the outputs should not be added to very shallow layers. For example, the first layer, in fact, with low-level feature, has very little discriminative power. If adding an output to this layer, the loss and the gradients are always very large, meaning that the training may not converge. Moreover, adding too many losses may cause severe over-fitting, which seriously influences the generalization ability of intermediate layers and results in great drop in terms of testing error (see details in Section 5.1). Therefore, we do not add outputs to each block or even each layer, which is very different from the approach proposed in [13].

Setting $\{\gamma_k\}$. The values of the $\{\gamma_k\}$ are critical to the performance of the proposed method. First, since features at deep layers have more discriminative power, the losses at deeper layers should be more important, and hence weighted more highly. The output of the final layer L is, of course, the most important, and hence we set its weight to 1 by default. For convenience, let L_k be the index of layer to which the k-th loss is applied. For the auxiliary losses, we suggest choosing $\gamma_k = \max(0.01, (\frac{L_k}{L_K})^\nu)$, where $\nu \geq 0$. In fact, if $\gamma_k < 0.01$, its effect can be negligible, so we set $\gamma_k \geq 0.01$. The parameter ν reflects the decay rate of $\{\gamma_k\}$ w.r.t. k. 1) If setting $\nu = 0$, it follows $\gamma_k = 1, \forall k$, and that every loss is equally

weighted. However, this setting is too aggressive since shallow layers which are often not very discriminative may hamper the performance. 2) If setting $\nu > 0$, the weights of losses at shallower layers will be smaller, which improves overall performance. 3) For a larger ν, the weights of the losses at shallow layers will decay faster to be negligible. In practice, we suggest choosing $1 \leq \nu \leq 2$, which implies a fast decay for the weights of losses at shallower layers, but does not eliminate their effect entirely.

3.2. Training Methods for AuxNet

AuxNet has the same forward propagation method as ResNet, but the gradient backpropagation (BP) method must be modified to exploit the additional losses. We here develop three approaches to gradient backpropagation for AuxNet.

3.2.1 Joint Gradient Backpropagation

Let ξ_k be the loss of the k-th output and ξ be the loss of the final output. The training process of AuxNet with auxiliary outputs can be considered to minimize the following objective function as in [13]

$$\mathcal{L} = \xi + \sum_{k=1}^{K} \gamma_k \xi_k. \tag{3}$$

In backpropagation, the supervision information from intermediate auxiliary outputs is jointly propagated to the shallower layers. Let $\mathbf{g}(\mathbf{x}_l)$ be the gradient of all $K+1$ losses w.r.t. $\mathbf{x}_l$. For any shallow layer l (where $L_{k-1} < l \leq L_k$), the gradient $\mathbf{g}(\mathbf{x}_l)$ of $\mathcal{L}$ w.r.t. $\mathbf{x}_l$ is computed by

$$\mathbf{g}(\mathbf{x}_l) = \frac{\partial \xi}{\partial \mathbf{x}_l} + \sum_{j=k}^{K} \gamma_j \frac{\partial \xi_j}{\partial \mathbf{x}_l}, k = 1, ..., K. \tag{4}$$

It is worth mentioning that when $\sum_{j=k}^{K} \gamma_k$ is very large, the second term in Eqn.(4) can be very large. Thus, the step size η in the original ResNet could be too large. To amend this, when $\sum_{k=1}^{K} \gamma_k > 1$, we can adjust the step size by $\eta_k := \eta / \sum_{j=k}^{K} \gamma_j$ to avoid convergence issues.

The joint gradient backpropagation can alleviate the issue of supervision information vanishing to some extent. Assuming $h(\mathbf{x}) = \mathbf{x}$, based on Eqn.(2), the gradient can be simplified as

$$\mathbf{g}(\mathbf{x}_l) = \mathbf{g}_1 + \mathbf{g}_2 + \mathbf{g}_3, \tag{5}$$

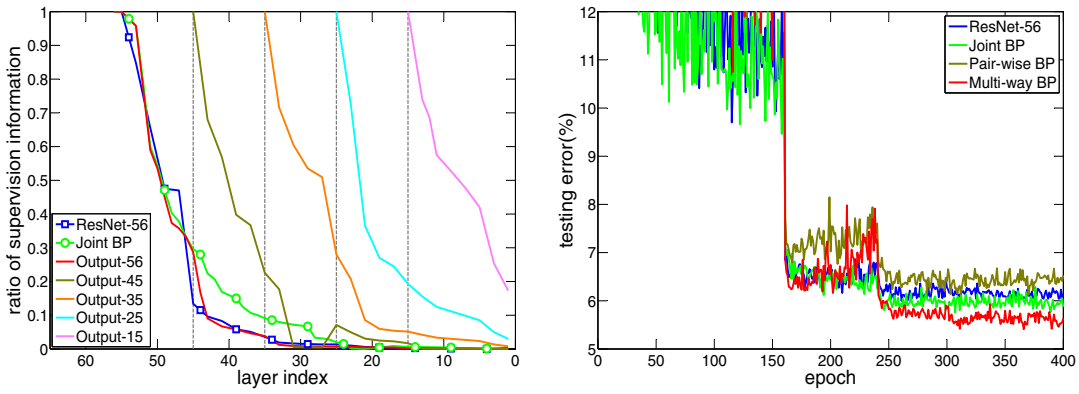
where $\mathbf{g}_1 = \partial_{\mathbf{x}_L} \xi$, $\mathbf{g}_2 = \sum_{j=k}^{K} \gamma_j \partial_{\mathbf{x}_{L_j}} \xi_j$ and $\mathbf{g}_3 = \partial_{\mathbf{x}_{L_j}} \xi_j \cdot (\sum_{j=k}^{K} \gamma_j \sum_{i=l}^{L_j - 1} \partial_{\mathbf{x}_l} \mathcal{F}(\mathbf{x}_i, \boldsymbol{\mathcal{W}}_i))$. The term $\mathbf{g}_2$ represents the original supervision information obtained from all the auxiliary losses. However, when L is very large, since ResNet can be considered as an ensemble of exponentially many shallow networks [29], the term $(\sum_{j=k}^{K} \gamma_j \sum_{i=l}^{L_j - 1} \partial_{\mathbf{x}_l} \mathcal{F}(\mathbf{x}_i, \boldsymbol{\mathcal{W}}_i))$ in $\mathbf{g}_3$ can be exponentially large and dominate the gradient $\mathbf{g}(\mathbf{x}_l)$. Therefore, the supervision information may still vanish when the network goes deeper. Similarly, the supervision vanishing issue may also happen in DSN [13].

3.2.2 Pair-wise Gradient Backpropagation

To address the above issue, a straightforward pair-wise gradient backpropagation method can be applied. In this method, we sequentially conduct forward propagation and backpropagation for each output. For any epoch t in SGD, suppose we deal with the outputs in an order of $k = 1, 2, ..., K+1$, where $K+1$ denotes the index of the final output. For the k-th output, we first compute the features $\{\mathbf{x}_l^k\}$ and the loss ξ_k regarding this output based on the model updated by the $(k-1)$-th output, denoted by $\mathbf{W}_t^{k-1}$ for simplicity. Here, when $k = 1$, we have $\mathbf{W}_t^0 = \mathbf{W}_{t-1}^{K+1}$. After that, we compute the gradient $\mathbf{g}_k(x_l)$ for the k-th output at any layer l that is lower than or equal to L_k by

$$\mathbf{g}_k(x_l) = \gamma_k \frac{\partial \xi_k}{\partial \mathbf{x}_{L_k}^k} \cdot \frac{\partial \mathbf{x}_{L_k}^k}{\partial \mathbf{x}_{L_k - 1}^k} ... \frac{\partial \mathbf{x}_{l+1}^k}{\partial \mathbf{x}_l^k}, \text{ for } l \leq L_k, \tag{6}$$

According to Eqn.(6), for any layer l, the supervision information of the nearest output can be successfully propagated to this layer, making the network easier to train.



(a) (b)

Figure 2. Performance comparison of AuxNet-56-5 on CIFAR-10 with different BP methods. (a) Supervision information changes w.r.t. layers; (b) Evolution of testing error.

3.2.3 Multi-way Gradient Backpropagation

In the pair-wise gradient backpropagation, for each output, the features and loss will be updated before doing the gradient backpropagation, which has two limitations. First, after backpropagation of the auxiliary output, the loss and the magnitude of the gradients for the next output shall decrease with very high probability, and this will affect the gradient propagation for the layers between the two outputs. Second, the forward propagation for each output will take additional cost. To amend these, we propose a more aggressive backpropagation method, called multi-way gradient backpropagation. In this method, we apply only one forward propagation, in which we update the features for all layers and compute losses for all the outputs. Keeping the features and losses fixed, we conduct the gradient backpropagation for each output separately in a sequential way. Let $\{\mathbf{x}_l\}$ be the features of layer l which are fixed for all outputs. In this case, for the k-th output, the gradient $\partial\xi_k/\partial\mathbf{x}_l$ for any layer l that is lower or equal to L_k shall be computed by

$$\frac{\partial \xi_k}{\partial \mathbf{x}_l} = \frac{\partial \xi_k}{\partial \mathbf{x}_L} \cdot \frac{\partial \mathbf{x}_L}{\partial \mathbf{x}_{L-1}} ... \frac{\partial \mathbf{x}_{l+1}}{\partial \mathbf{x}_l}, \quad \text{for } \; l \leq L_k. \tag{7}$$

Assuming $h(\mathbf{x}) = \mathbf{x}$, we will have

$$\mathbf{g}_k(x_l) = \gamma_k \partial_{\mathbf{x}_{L_k}} \xi_k + \gamma_k \partial_{\mathbf{x}_{L_k}} \xi_k \cdot (\sum_{i=l}^{L_k - 1} \partial_{\mathbf{x}_l} \mathcal{F}(\mathbf{x}_i, \boldsymbol{\mathcal{W}}_i)), \tag{8}$$

which is similar to Eqn.(2) for ResNet.

The major characteristic of the multi-way method is the performance of multiple independent backpropagations, where only one forward propagation is applied. Due to Eqns.(7) and (8), intermediate layers can obtain sufficient supervision signal from independent backpropagations, and thus the issue of supervision information vanishing can be trivially addressed.

3.3. An Experimental Demonstration

To better understand the effect on ResNet of having supervision information vanishing, as well as the three gradient backpropagation methods, we compare ResNet with 56 layers and AuxNet with 56 layers and 4 auxiliary outputs (at layers 15, 25, 35, 45). Note that the latter two backpropagation methods are similar, here we only study multi-way backpropagation on the issue of supervision information vanishing. For simplicity, based on Eqn.(8), for the k-th output of AuxNet, we measure the supervision component of the whole gradient by $||\gamma_k \partial_{\mathbf{x}_{L_k}} \xi_k||/||\mathbf{g}_k(\mathbf{x}_l)||$ at the final training epoch of SGD. Similarly, based on Eqns.(2) and (5), we measure the component of the gradients by $||\partial_{\mathbf{x}_L} \xi||/||\mathbf{g}(\mathbf{x}_l)||$ for ResNet and $||\mathbf{g}_1 + \mathbf{g}_2||/||\mathbf{g}(\mathbf{x}_l)||$ for AuxNet with joint gradient backpropagation, respectively. The ratios are shown in Figure 2(a). From the figure, the supervision information for ResNet and joint gradient backpropagation indeed diminishes very fast in shallower layers. However, for AuxNet with multi-way gradient backpropagation, because of multiple individual backpropagations, the supervision component remains stable.

Table 1. Effects of different training methods for networks with auxiliary outputs on CIFAR-10.

training method	network	outputs position	error(%)
Original BP	ResNet-44 [7]	{44}	6.37
	ResNet-56 [7]	{56}	6.08
Joint BP	AuxNet-44-3	{44, 35, 25}	6.00
	AuxNet-56-5	{56, 45, 35, 25, 15}	5.89
Pair-wise BP	AuxNet-44-3	{44, 35, 25}	6.45
	AuxNet-56-5	{56, 45, 35, 25, 15}	6.22
Multi-way BP	AuxNet-44-3	{44, 35, 25}	**5.85**
	AuxNet-56-5	{56, 45, 35, 25, 15}	**5.53**

It is worth mentioning that, unlike the joint gradient backpropagation, the latter two approaches do not minimize a joint objective function as in Eqn.(3). However, as is visible from Figure 2(b) and Table 1, except the pair-wise gradient backpropagation, the other two approaches with auxiliary outputs work better than the ResNet with 56 layers and only one output in terms of testing accuracy. Moreover, the multi-way gradient backpropagation shows a significant improvement over the other two approaches. To keep the length of paper within reasonable limits, in our later experiments, we only study AuxNet with multi-way gradient backpropagation.

4. Experiments

We implement AuxNet based on a Torch implementation of ResNet [7][1], and only the multi-way backpropagation is studied since it yields consistently better performance than the other two methods. Several state-of-the-art deep learning models are adopted as baselines, including FitNet [19], DSN [13], Frac.Pool [4], Highway Network [23], Stochastic Depth Network (StochResNet) [8], Residual Networks (ResNet) [7] and Wide Residual Network(Wide-ResNet) [31]. We conducted comparisons on following benchmark data sets: CIFAR-10, CIFAR-100, SVHN and ImageNet.

All experimental settings for AuxNet remain the same as in [6]. Networks are trained with SGD using a mini-batch size of 128 on 2 GPUs (64 each). The learning rate starts from 0.1 and is divided by 10 at 40% and 60% of total epochs. The weight decay in the objective function is 0.0001, the momentum is 0.9 and the weights are initialized as in [5]. By training AuxNet with multi-way gradient backpropagation, we use the final output to obtain the final prediction. Our pretrained models are available at https://github.com/guoyongcs/auxnet.

All the experiments are performed on a GPU Server on which a 64-bit Ubuntu Server 14.04 LTS is installed, with two Intel(R) Xeon E5-2620 v3 CPUs (2.40GHz), 128GB memory and two Titan X GPUs.

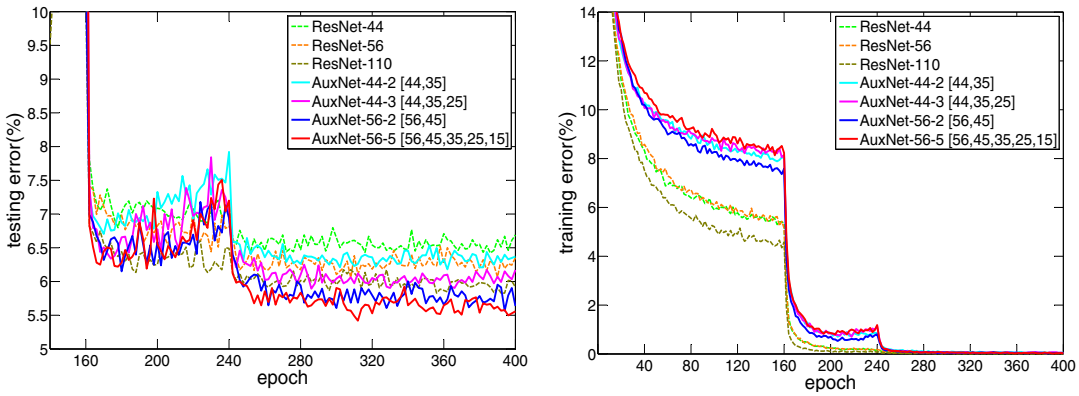
4.1. Comparison with State-of-the-art Methods

CIFAR-10 and ***CIFAR-100***. CIFAR-10 [9] contains 10 classes of 32x32 natural color images, each with 5,000 training samples and 1,000 testing samples. CIFAR-100 [9] contains 100 classes. Each class has 500 training samples and 100 testing samples. We perform 400 SGD epochs for the training, and use simple data augmentation such as translation and horizontal flipping as in [13].

Note that AuxNet is rooted in ResNet. In the first experiment, we thus first compare the testing error evolution of 3 ResNets of (44, 56, 110)-layers and 4 AuxNets of different settings, as shown in Figure 3(a). In the figure, AuxNet-44-2 denotes a AuxNet network with 44 layers and 1 auxiliary output (i.e., 2 outputs in total). The numbers in brackets indicate the indices of layers with added outputs.

From Figure 3(a), we have the following observations. First, the deeper ResNet shows better performance than its shallower counterparts, and ResNet-110 demonstrates the best testing performance with an error rate of 5.86%. Second, AuxNet-44-2, AuxNet-44-3, AuxNet-56-2, and AuxNet-56-5 all perform significantly better than their ResNet counterparts with the same depth. Third, with much fewer layers, AuxNet-56-2 and AuxNet-56-5 perform better than the deeper ResNet with 110 layers. Forth, AuxNet-56-5 and AuxNet-44-3 with more auxiliary outputs outperform AuxNet-56-2 and AuxNet-44-2, respectively. All these observations strongly prove the effectiveness of AuxNet trained by multi-way backpropagation. Specifically, due to the auxiliary outputs and the proposed gradient propagation method, the supervision information can be successfully propagated to intermediate layers, which improves the discriminative power of intermediate layers and hence helps to learn a much more compact model. This explains why AuxNet-44-3 with only 44 layers performs better than ResNet-110 with 110 layers. On the other hand, as we can see from Figure 3(b), the training error of AuxNet is competitive and often

[1] https://github.com/facebook/fb.resnet.torch.



(a) Testing error (b) Training error

Figure 3. Testing error and training error evolution of ResNet and different AuxNet on CIFAR-10.

Table 2. Classification error on CIFAR-10 and CIFAR-100. Note that $\{\cdot/N\}$ indicates a N-fold increase in network width, e.g. AuxNet-26-2/10 is 10 times wider than the baseline.

network	depth	outputs position	error (%)	
			CIFAR-10	CIFAR-100
FitNet [19]	19	{19}	8.39	35.04
DSN [13]	11	{11, 9, 5, 1}	7.97	34.57
Frac.Pool, 1 test [4]	15	{15}	4.50	31.20
Highway Network [23]	19	{19}	7.60	32.24
StochResNet-110 [8]	110	{110}	5.23	24.58
ResNet-20 [7]	20	{20}	7.76	31.12
ResNet-32 [7]	32	{32}	6.81	29.74
ResNet-44 [7]	44	{44}	6.37	28.85
ResNet-56 [7]	56	{56}	6.08	28.46
ResNet-110 [7]	110	{110}	5.86	27.41
ResNet-1001 [7]	1001	{1001}	4.62	22.71
AuxNet-20-2	20	{20, 17}	7.78	30.44
AuxNet-32-2	32	{32, 25}	6.83	28.24
AuxNet-44-2	44	{44, 35}	6.12	27.46
AuxNet-44-3	44	{44, 35, 25}	5.85	27.19
AuxNet-56-2	56	{56, 45}	5.77	26.83
AuxNet-56-5	56	{56, 45, 35, 25, 15}	5.53	26.62
AuxNet-26-2/2	56	{26, 19}	5.48	24.36
AuxNet-56-2/2	56	{56, 45}	4.77	22.95
AuxNet-56-5/2	56	{56, 45, 35, 25, 15}	4.51	22.57
Wide-ResNet-26/10 [31]	26	{26}	4.17	20.50
AuxNet-26-2/10	26	{26, 19}	**3.77**	**19.69**

slightly larger than those of the ResNet. As we have described, ResNet uses backpropagation to learn the networks. For very deep models, the discriminative power of upper layers may become limited due to long propagation paths in networks. Keeping these hidden layers which provide little representative contribution can degenerate the ability of the network to generalize to new data. However, AuxNet can learn a more sophisticated representation because it allows intermediate layers to capture the supervision information from nearby output. This is the reason why the training error of AuxNet in some cases is larger than ResNet.

Now we report a comprehensive comparison of AuxNet with other baselines on both CIFAR-10 and CIFAR-100 data sets in terms of testing errors. Note that as shown in Figure 1, we can increase the network width by a factor of N to improve the representation ability of each layer. The settings of AuxNet and results can be found in the Table 2.

From Table 2, we have the following observations. First, our AuxNet with auxiliary outputs achieves better or comparable performance with ResNet and other baselines. Specifically, AuxNet-56-5 yields 5.53% and 26.62% error on CIFAR-10 and CIFAR-100, respectively, which are much better than its ResNet-56 counterpart and ResNet-110 with far more layers. This proves that AuxNet indeed is capable of reducing the model redundancy. Second, by increasing the layer width, AuxNet-

Table 3. Testing error on SVHN. † denotes results with dropout.

network	outputs position	error (%)
FitNet [19]	{19}	2.42
DSN [13]	{21, 16, 9, 2}	1.92
StochResNet [8]	{152}	1.75
ResNet-44 [7]	{44}	2.15
ResNet-56 [7]	{56}	2.06
ResNet-110 [7]	{110}	1.97
ResNet-44† [7]	{44}	1.91
ResNet-56† [7]	{56}	1.70
ResNet-110† [7]	{110}	1.65
AuxNet-44-2	{44, 35}	1.96
AuxNet-56-2	{56, 45}	1.84
AuxNet-44-2†	{44, 35}	1.75
AuxNet-56-2†	{56, 45}	1.63
AuxNet-56-3†	{56, 45, 35}	**1.58**

26-2/10 (10x wider) achieves the best performance simultaneously on CIFAR-10 (3.77%) and CIFAR-100 (19.69%). These observations demonstrate that, by leveraging the power of auxiliary outputs, AuxNet trained by multi-way backpropagation improves the performance of ResNet by a large margin.

SVHN. The Street View House Number (SVHN) [17] data contains house number images of 32x32 pixels, which includes 73237 digits for training, 26032 digits for testing, and 531131 digits as extra training data. The number of epochs is set to 200, and the images are divided by 255 for scaling without data augmentation. From Table 3, AuxNet-56-2 with 56 layers improves the performance of ResNet-110 with a deeper structure significantly (yielding 1.84% in error rate). When using dropout, AuxNet-56-3 results in 1.58% error rate, which is the best published result on SVHN to our knowledge.

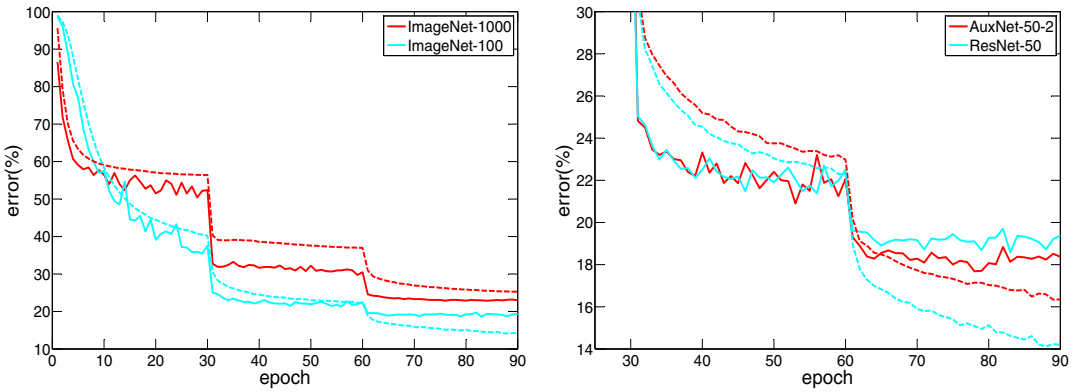
ImageNet-1000. ImageNet-1000 [20] contains 1,000 classes with 1.28 million training images and 50k testing images. The experiment setting is the same as that given in [6]. We scale the learning rate by a factor of 0.5 to train AuxNet with 90 epochs. Table 4 shows the results of top-1 and top-5 errors on the ImageNet-1000.

Table 4. Validation error on ImageNet-1000 (**10-crop** testing).

network	outputs position	error (%)	
		top-1	top-5
VGG-16 [22]	{16}	28.07	9.33
GoogLeNet [27]	{22}	-	9.15
PReLU-net [5]	{22}	24.27	7.38
ResNet-34 [6]	{34}	24.76	7.35
ResNet-50 [6]	{50}	22.85	6.71
AuxNet-34-2	{34, 29}	24.61	7.18
AuxNet-50-2	{50, 40}	22.77	6.27
AuxNet-50-4	{50, 40, 31, 22}	22.34	6.07

From the results in Table 4, AuxNet achieves better performance than ResNet. For example, AuxNet-34-2 with 34 layers is superior to ResNet-34, yielding 24.61% on top-1 error and 7.18% on top-5 error. The AuxNet-50-2 with 50 layers also outperforms the ResNet-50, resulting in 22.77% on top-1 error and 6.27% on top-5 error. When adding more auxiliary outputs, the AuxNet-50-4 with 4 outputs in total further improve the performance to 22.43% and 6.07% on top-1 error and top-5 error respectively, which outperforms the ResNet-50 and AuxNet-50-2 with 2 outputs. On ImageNet-1000, the improvement of AuxNet is not as significant as that on the previous small scale data sets. One possible reason is that the number of classes for ImageNet-1000 is too large, making it very challenging to solve with limited data. To demonstrate this, we conducted an experiment on a subset of ImageNet-1000, a data set containing a randomly selected 100 classes from ImageNet-1000. This testing shows that AuxNet achieves significant improvement on this subset.

ImageNet-100. To show the influence of the number of classes, we conduct experiment on the ImageNet-100 data set, by randomly selecting 100 classes from the ImageNet-1000 data set with 1,000 classes. The ImageNet-100 data set consists of 128k training images and 5k testing images.



(a) Validation error and training error evolution of ResNet-56 on ImageNet-1000 and ImageNet-100

(b) Validation error and training error evolution of ResNet-56 and AuxNet-56-2 on ImageNet-100

Figure 4. Training on ImageNet-1000 and ImageNet-100. Dash curves denote training error, and solid curves denote validation error of 10-crop testing.

We compare the ResNet-50 with 50 layers on ImageNet-1000 and ImageNet-100 data sets respectively. From Figure 4(a), in contrast to the ImageNet-100 data set, the training curve is always above the testing curve on ImageNet-1000 due to the training difficulty of large scale data set. The result indicates that the ResNet-50 with 50 layers on ImageNet-1000 is not very redundant. As a result, the improvement of AuxNet over ResNet is not significant in this case. On the smaller data set ImageNet-100, the AuxNet-50-2 (with 50 layers and 1 auxiliary output at layer 40) performs better than the baseline ResNet-50 with the same depth (see Figure 4(b)). This result shows that introducing auxiliary outputs is able to reduce the model redundancy and improve the performance.

Table 5. Classification error on the ImageNet-100 (**10-crop** testing) validation set.

network	outputs position	error (%)	
		top-1	top-5
ResNet-34 [6]	{34}	19.22	4.65
ResNet-50 [6]	{50}	18.57	4.39
ResNet-101 [6]	{101}	19.68	4.87
AuxNet-34-2	{34, 29}	19.02	4.33
AuxNet-50-2	{50, 40}	17.93	3.98
AuxNet-101-2	{101, 88}	**17.87**	**3.84**

In Table 5, we report both top-1 and top-5 error rates for the compared approaches on the ImageNet-100 data set. The ResNet achieves 19.22% and 18.57% on top-1 error with 34 layers and 50 layers, respectively. However, the ResNet-101 with 101 layers shows a reduction in the classification accuracy (19.68% error), which implies that a deeper network architecture does not necessarily produce better performance on relatively simpler tasks. This is also in concordance with the conclusion in [6], which reports a performance decay of a very deep network with 1,202 layers. A possible reason is that, without auxiliary outputs, a very deep ResNet network may not be necessarily better for simpler tasks (100-classes compared to 1000-classes).

The proposed AuxNet consistently yields better performance in different settings. For example, AuxNet-50-2 of 50 layers performs much better than ResNet-50, yielding 3.4% and 9.3% relative improvement on top-1 error and top-5 error, respectively. In contrast to ResNet, AuxNet-101 with 101 layers shows better performance than AuxNet-50 and also much better performance than ResNet-101. Specifically, AuxNet-101 shows 9.2% and 21.1% relative improvement on top-1 error and top-5 error over ResNet-101, respectively.

4.2. Prediction Ability of Intermediate Models

The AuxNet training process inherently generates multiple models of differing depths. We show here that these shallower intermediate models often outperform their full-depth ResNet counterparts. We also show that they outperform the corresponding layers of ResNet, and therefore the intermediate layers of AuxNet are more discriminative. Table 6 thus compares

the prediction performance of intermediate models generated by each loss of AuxNet-56-5 against that of the corresponding layer of ResNet-56.

From Table 6 we see that each intermediate model of AuxNet-56-5 outperforms its ResNet comparator. By comparing these results with Table 2, we see that the AuxNet-56-5 intermediate model obtained at output-45 shows much better performance (5.67% error) than ResNet-44 (6.37%), ResNet-56 (6.08%), ResNet-110 (5.86%), as well as AuxNet-44-3 (5.85%). This demonstrates not only that AuxNet outperforms ResNet both in full depth, and at the intermediate layers, but also that it provides the opportunity for a form of model selection, and a means of avoiding over-fitting.

In Table 6, the third column records the number of model parameters. Fewer model parameters implies faster inference, which is important for real-world applications. For example, the AuxNet-56-5 model at **output-45** has 0.48M parameters, which is nearly half the number in ResNet-56 (0.85M), and 1/3 of that in ResNet-110 (1.7M).

Table 6. Testing error of intermediate models of AuxNet-56-5 and ResNet-56 on CIFAR-10.

model	#layers	#params	error (%)	
			AuxNet	ResNet
output-56	56	0.85M	**5.53**	6.56
output-45	45	0.48M	**5.67**	13.71
output-35	35	0.18M	9.23	34.01
output-25	25	0.09M	18.94	45.07
output-15	15	0.03M	50.35	63.01

5. More Discussions

The performance of AuxNet is affected by different factors and we study them in the following aspects:

1. Adding too many losses may cause severe over-fitting (see Section 5.1).
2. Adding losses at very shallow layers severely hampers the performance (see Section 5.2).
3. Hyperparameters, such as ν and λ, can be set properly with simple principle (see Section 5.3 and 5.4).

5.1. Effects of Varying Numbers of Outputs

In this experiment, we test the performance of AuxNet for various numbers of intermediate outputs on the CIFAR-10 data set. The results are given in Table 7. We see that AuxNet-56-2 and AuxNet-56-5 perform significantly better than ResNet-56. However, adding too many outputs does not necessarily improve the performance. For example, AuxNet-56-10 with 10 outputs performs much worse than AuxNet-56-2, AuxNet-56-5 and ResNet-56.

Table 7. Testing error for varying number of outputs on CIFAR-10.

network	outputs position	error (%)
ResNet-56 [7]	{56}	6.08
AuxNet-56-2	{56, 45}	5.77
AuxNet-56-5	{56, 45, 35, 25, 15}	**5.53**
AuxNet-56-10	{56, 45, 41, 37, 33, 29, 25, 21, 17, 13}	7.96

5.2. Outputs at Very Shallow Layers

While auxiliary outputs can help a simpler model to achieve the same accuracy as a deeper ResNet model, adding auxiliary outputs to very shallow layers may severely hampers the prediction performance since the features of shallow layers may not be sufficiently discriminative. To demonstrate this, we investigate two AuxNet models by adding single output to very shallow layers, and record the results in Table 8.

From Table 8, adding outputs at very shallow layer, e.g. 5 out of ResNet-20 and ResNet-44, severely influences the prediction performance. That means, in practice, it is necessary to avoid adding auxiliary outputs at very shallow layers.

Table 8. Effects of auxiliary outputs at very shallow layers on CIFAR-10 test set.

network	outputs position	error (%)
ResNet-20 [7]	{20}	7.76
AuxNet-20-2	{20, 5}	9.64
AuxNet-20-2	{20,17}	7.58
ResNet-44 [7]	{44}	6.37
AuxNet-44-2	{44, 5}	7.73
AuxNet-44-2	{44,15}	6.75
AuxNet-44-2	{44,35}	6.12

5.3. Effect of Weighting Scalar ν

We conducted experiments to investigate how to weight outputs to achieve better performance. Under the setting of $\{\gamma_k\}$ described in Section 3.1, we compare different value of ν to adjust weights. From Table 9, the equally weighted outputs ($\nu = 0$) severely hampers the performance. If $\nu \geq 1$, the weights of shallower outputs become smaller due to their decreasing discriminative powers. Especially for $\nu = 1$ or $\nu = 2$, the n-layer AuxNet behaves better than the original n-layer ResNet. For a larger $\nu = 5$, the weights decay so aggressively that the effects of auxiliary outputs are negligible.

Table 9. Effect of weighting scalar ν on CIFAR-10 test set.

network	ν	error (%)
ResNet-56 [7]	-	6.08
AuxNet-56-5	0	8.43
	1	5.90
	2	5.53
	5	6.03

5.4. Effects of the Parameter λ

The choice of the parameter λ in the shortcut mapping is critical to the performance of residual networks. When $\lambda = 0$, ResNet is reduced to the plain network. We study the influence of λ on shortcuts within a residual unit of ResNet and AuxNet. The values of λ considered are $\{0.5, 0.95, 1\}$. From Table 10, the AuxNet consistently outperforms ResNet with different λ settings. We also observe that the performance of AuxNet is robust across different values of λ, and the default parameter setting $\lambda = 1.0$ gives good results in our experiments, whilst ResNet is more sensitive.

Table 10. Classification error on CIFAR-10 test set based on ResNet-56, with different λ applied to $\mathbf{x}_l$.

λ	network	# outputs	error (%)
0.5	ResNet-56 [7]	1	9.25
	AuxNet-56-5	5	6.87
0.95	ResNet-56 [7]	1	6.32
	AuxNet-56-5	5	6.14
1	ResNet-56 [7]	1	6.08
	AuxNet-56-5	5	5.53

6. Conclusion

Besides gradient vanishing, very deep networks also suffer from a vanishing supervision information issue. In the course of the investigation we developed a novel model, which we labelled AuxNet. This model addresses the vanishing supervision information issue through the use of additional auxiliary outputs. We proposed three gradient backpropagation methods to train the network with multiple losses. In particular, we developed a multi-way gradient backpropagation method, which gives rise to models which outperform ResNet. The proposed method reduces the internal redundancy and thus makes the models significantly more compact. Because of the fact that it optimizes multiple intermediate models, AuxNet also offers the opportunity for a form of model selection.

References

[1] J. Ba and R. Caruana. Do deep nets really need to be deep? In *Advances in Neural Information Processing Systems*, pages 2654–2662, 2014.

[2] N. Cohen, O. Sharir, and A. Shashua. On the expressive power of deep learning: A tensor analysis. In *Conference on Learning Theory*, 2016.

[3] X. Glorot and Y. Bengio. Understanding the difficulty of training deep feedforward neural networks. In *International Conference on Artificial Intelligence and Statistics*, volume 9, pages 249–256, 2010.

[4] B. Graham. Fractional max-pooling. *arXiv preprint arXiv:1412.6071*, 2014.

[5] K. He, X. Zhang, S. Ren, and J. Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1026–1034, 2015.

[6] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016.

[7] K. He, X. Zhang, S. Ren, and J. Sun. Identity mappings in deep residual networks. *European Conference on Computer Vision*, 2016.

[8] G. Huang, Y. Sun, Z. Liu, D. Sedra, and K. Q. Weinberger. Deep networks with stochastic depth. *CoRR*, abs/1603.09382, 2016.

[9] A. Krizhevsky and G. Hinton. Learning multiple layers of features from tiny images, 2009.

[10] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems*, pages 1097–1105, 2012.

[11] Y. LeCun, Y. Bengio, and G. Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015.

[12] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. Backpropagation applied to handwritten zip code recognition. *Neural Computation*, 1(4):541–551, 1989.

[13] C.-Y. Lee, S. Xie, P. Gallagher, Z. Zhang, and Z. Tu. Deeply-supervised nets. In *International Conference on Artificial Intelligence and Statistics*, 2015.

[14] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3431–3440, 2015.

[15] G. F. Montufar, R. Pascanu, K. Cho, and Y. Bengio. On the number of linear regions of deep neural networks. In *Advances in Neural Information Processing Systems*, pages 2924–2932, 2014.

[16] V. Nair and G. E. Hinton. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, pages 807–814, 2010.

[17] Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, and A. Y. Ng. Reading digits in natural images with unsupervised feature learning. 2011.

[18] S. Ren, K. He, R. Girshick, and J. Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in Neural Information Processing Systems*, pages 91–99, 2015.

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

[20] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, et al. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 115(3):211–252, 2015.

[21] F. Schroff, D. Kalenichenko, and J. Philbin. Facenet: A unified embedding for face recognition and clustering. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 815–823, 2015.

[22] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[23] R. K. Srivastava, K. Greff, and J. Schmidhuber. Highway networks. *International Conference on Machine Learning*, 2015.

[24] R. K. Srivastava, K. Greff, and J. Schmidhuber. Training very deep networks. In *Advances in Neural Information Processing Systems*, pages 2377–2385, 2015.

[25] Y. Sun, D. Liang, X. Wang, and X. Tang. DeepID3: Face recognition with very deep neural networks. *arXiv preprint arXiv:1502.00873*, 2015.

[26] Y. Sun, X. Wang, and X. Tang. Deeply learned face representations are sparse, selective, and robust. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2892–2900, 2015.

[27] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–9, 2015.

[28] S. Teerapittayanon, B. McDanel, and H. Kung. Branchynet: Fast inference via early exiting from deep neural networks, 2016.

[29] A. Veit, M. Wilber, and S. Belongie. Residual networks are exponential ensembles of relatively shallow networks. *Advances in Neural Information Processing Systems*, 2016.

[30] L. Wang, C.-Y. Lee, Z. Tu, and S. Lazebnik. Training deeper convolutional networks with deep supervision. *arXiv preprint arXiv:1505.02496*, 2015.

[31] S. Zagoruyko and N. Komodakis. Wide residual networks. *arXiv preprint arXiv:1605.07146*, 2016.

[32] S. Zagoruyko, A. Lerer, T.-Y. Lin, P. O. Pinheiro, S. Gross, S. Chintala, and P. Dollár. A multipath network for object detection. Technical report, Facebook AI Research, 2016.

[33] M. D. Zeiler and R. Fergus. Visualizing and understanding convolutional networks. In *European Conference on Computer Vision*, pages 818–833. Springer, 2014.