VARIABLE ACTIVATION NETWORKS: A SIMPLE METHOD TO TRAIN DEEP FEED-FORWARD NETWORKS WITHOUT SKIP-CONNECTIONS

Anonymous authors
Paper under double-blind review

ABSTRACT

Novel architectures such as ResNets have enabled the training of very deep feed-forward networks via the introduction of skip-connections, leading to state-of-the-art results in many applications. Part of the success of ResNets has been attributed to improvements in the conditioning of the optimization problem (e.g., avoiding vanishing and shattered gradients). In this work we propose a simple method to extend these benefits to the context of deep networks without skip-connections. The proposed method poses the learning of weights in deep networks as a constrained optimization problem where the presence of skip-connections is penalized by Lagrange multipliers. This allows for skip-connections to be introduced during the early stages of training and subsequently phased out in a principled manner. We demonstrate the benefits of such an approach with experiments on MNIST, fashion-MNIST, CIFAR-10 and CIFAR-100 where the proposed method is shown to outperform many architectures without skip-connections and is often competitive with ResNets.

1 INTRODUCTION

The representation view of deep learning suggests that neural networks learn an increasingly abstract representation of input data in a hierarchical fashion (Zeiler & Fergus, 2014; Goodfellow et al., 2016; Greff et al., 2016). Such representations may then be exploited to perform various tasks such as image classification, machine translation and speech recognition.

A natural conclusion of the representation view is that deeper networks will learn more detailed and abstract representations as a result of their increased capacity. However, in the case of feed-forward networks it has been observed that performance deteriorates beyond a certain depth, even within training data. Recently, Residual Networks (ResNets; He et al., 2016a) and Highway Networks (Srivastava et al., 2015) have demonstrated that introducing various flavors of skip-connections or gating mechanisms makes it possible to train increasingly deep networks.

A widely held hypothesis explaining the success of ResNets is that the introduction of skip-connections serves to improve the conditioning of the optimization manifold as well as the statistical properties of gradients employed during training. Raiko et al. (2012) and Schraudolph (2012) show that the introduction of specially designed skip-connections serves to diagonalize the Fisher information matrix, thereby bringing standard gradient steps closer to the natural gradient. More recently, Baldazzi et al. (2017) demonstrated that the introduction of skip-connections helps retain the correlation structure across gradients. This is contrary to the gradients of deep feed-forward networks, which resemble white noise. More generally, the skip-connections are thought to reduce the effects of vanishing gradients by introducing a linear component to the activation function (He et al., 2016b).

Despite the success of ResNets and Highway Networks, some issues still remain. For instance, it is often not clear what the actual depth of the network is nor is the contribution of each layer to the final representation obvious. Greff et al. (2016) suggest that the level of representation remains constant across blocks of multiple layers and only varies when the dimensionality changes, typically through projections. A related issue is feature re-use, whereby higher layers may not necessarily learn useful representations. Indeed, Zagoruyko & Komodakis (2016) demonstrate that it is better to increase
the width of ResNets than the depth, suggesting that perhaps only a few layers are learning useful representations.

The goal of this work is to address some of the aforementioned issues by leveraging some of the desirable optimization properties of ResNets in the context of neural networks that lack skip-connections. We approach the task of learning parameters for a deep network under the framework of constrained optimization. This strategy allows us to introduce skip-connections penalized by Lagrange multipliers into the architecture of our network. In our setting, skip-connections play an important role during the initial training of the network and are subsequently removed in a principled manner. Throughout a series of experiments we demonstrate that such an approach leads to improvements in generalization error when compared to architectures without skip-connections and is competitive with ResNets in some cases.

The contributions of this work are as follows:

- We propose a novel alternative for training deep feed-forward networks without skip-connections. The proposed method introduces skip-connections which are penalized by Lagrange multipliers. This allows for the presence of skip-connections to be iteratively phased out during training in a principled manner. The proposed method is thereby able to enjoy the optimization benefits associated with skip-connections during the early stages of training.

- A number of benchmark datasets are used to demonstrate the empirical capabilities of the proposed method. In particular, the proposed method consistently out-performs deep networks trained using traditional back-propagation and is competitive with ResNets in some cases.

2 RELATED WORK

The hierarchical nature of many feed-forward networks is loosely inspired by the structure of the visual cortex where neurons in early layers capture simple features (e.g., edges) which are subsequently aggregated in deeper layers (Hubel & Wiesel, 1962). This interpretation of neural networks suggests that the depth of a network should be maximized, thereby allowing the network to learn more abstract (and hopefully useful) representations (Bengio et al., 2013). However, a widely reported phenomenon is that deeper networks are more difficult to train. This is the result of optimization challenges such as vanishing and shattered gradients (Hochreiter et al., 2001; Balduzzi et al., 2017).

In the past these challenges have been partially addressed via the use of supervised and unsupervised pre-training (Bengio et al., 2009) and more recently through careful parameter initialization (Glorot & Bengio, 2010; He et al., 2015) and batch normalization (Ioffe & Szegedy, 2015). In the past couple of years further improvements have been obtained via the introduction of skip-connections. ResNets (He et al., 2016a; b) replace traditional activation functions within each layer with a residual block consisting of a residual function $F$ together with a skip-connection. Formally, the residual block is defined as:

$$x_{l+1} = F_l(x_l, W_l) + W'_l x_l$$

where $F_l : \mathbb{R}^n \rightarrow \mathbb{R}^{n'}$ represents some combination of affine transformation, nonlinearity and batch normalization parameterized by $W_l$. The matrix $W'_l$ parameterizes a linear projection to ensure the dimensions are aligned. More generally, ResNets can be considered a special case of Highway Networks (Srivastava et al., 2015) where the output of each layer is defined as:

$$x_{l+1} = F_l(x_l, W_l) \cdot T(x_l, H_l) + x_l \cdot (1 - T(x_l, H_l)),$$

where $\cdot$ denotes element-wise multiplication. In Highway Networks the output of each layer is determined by a gating function $T(x_l, H_l) = \text{sigmoid}(H_l x_l)$ inspired from LSTMs. We note that both ResNets and Highway Networks were introduced with the explicit goal of training deeper networks. Inspired by the success of the these methods, many

\footnote{Unless stated otherwise we will assume $F$ retains the dimension of $x_l$ and set $W'_l$ to the identity.}
variations have been proposed. Huang et al. (2016a) propose DenseNet, where skip-connections are passed from all previous activations. Huang et al. (2016b) propose to shorten networks during training by randomly dropping entire layers, leading to better gradient flow and information propagation, while using the full network at test time. Recently, the goal of learning deep networks without skip-connections has begun to receive more attention. Zagoruyko & Komodakis (2017) propose a novel re-parameterization of weights in feed-forward networks which they call the Dirac parameterization. Instead of explicitly adding a skip-connection, they model the weights as a residual of the Dirac function, effectively moving the skip-connection inside the non-linearity. In related work, Baldazzi et al. (2017) propose to initialize weights in a CReLU activation function in order to preserve linearity during the initial phases of training. This is achieved by initializing the weights in a mirrored block structure. During training the weights are allowed to diverge, resulting in non-linear activations.

Finally, we note that while the aforementioned approaches have sought to train deeper networks via modifications to the network architecture (i.e., by adding skip-connections) success has also been obtained by modifying the non-linearities (Clevert et al., 2015; Klambauer et al., 2017).

3 VARIABLE ACTIVATION NETWORKS

The goal of this work is to train deep feed-forward networks without skip-connections. To set notation, we denote \( x_0 \) as the input and \( x_L \) as the output of a feed-forward network with \( L \) layers. Given training data \( \{y, x_0\} \) it is possible to learn parameters \( \{W_l\}_{l=1}^L \) by locally minimizing some objective function

\[
\{W_l\}_{l=1}^L = \arg\min C\left(y, x_L; \{W_l\}_{l=1}^L\right). \tag{3}
\]

First-order methods are typically employed due to the complexity of the objective function in equation (3). However, directly minimizing the objective is not practical in the context of deep networks: beyond a certain depth performance quickly deteriorates on both test and training data. Such a phenomenon, often termed the degradation problem (He et al., 2016b), does not occur in the presence of skip-connections. Accordingly, we take inspiration from ResNets and propose to modify equation (1) in the following manner:

\[
x_{l+1} = \alpha_l \cdot F_l(x_l, W_l) + (1 - \alpha_l) \cdot x_l \tag{4}
\]

where \( \alpha_l \in [0, 1]^n \) determines the convex combination between the non-linearity and the skip-connection. More specifically, \( \alpha_l \) is a vector where the entry \( i \) dictates the balance between non-linearity and skip-connection for the neuron \( i \) in layer \( l \). Due to the variable nature of parameters \( \alpha_l \) in equation (4), we refer to networks employing such an activation as Variable Activation Networks (VAN).

The objective of the proposed method is to train a feed-forward network under the constraint that \( \alpha_l = 1 \) for all layers, \( l \). When the constraint is satisfied all skip-connections are removed. The advantage of such a strategy is that we only require \( \alpha_l = 1 \) at the end of training. This allows us to initialize \( \alpha_l \) to some other value, thereby relaxing the optimization problem and obtaining the advantages associated with ResNets during the early stages of training. In particular, whenever \( \alpha_l \neq 1 \) information is allowed to flow through the skip-connections, alleviating issues associated with shattered and vanishing gradients.

As a result of the equality constraint on \( \alpha_l \), the proposed activation function effectively does not introduce any additional parameters. All remaining weights can be trained by solving the following constrained optimization problem:

\[
\{\hat{W}_l\}_{l=1}^L = \arg\min C\left(y, x_L; \{W_l, \alpha_l\}_{l=1}^L\right) \text{ such that } \alpha_l = 1 \text{ for } l = 1, \ldots, L. \tag{5}
\]

The associated Lagrangian takes the following simple form (Boyd & Vandenberghe, 2004):

\[
\mathcal{L} = C\left(y, x_L; \{W_l, \alpha_l\}_{l=1}^L\right) + \sum_{l=1}^L \lambda_l^T (\alpha_l - 1), \tag{6}
\]
where each $\lambda_l \in \mathbb{R}^n$ are the Lagrange multipliers associated with the constraints on $\alpha_l$. In practice, we iteratively update $\alpha_l$ via stochastic gradients descent (SGD) steps of the form:

$$\alpha_l \leftarrow \alpha_l - \eta \left( \frac{\partial C}{\partial \alpha_l} - \lambda_l \right) \quad \text{(7)}$$

where $\eta$ is the step-size parameter for SGD. Throughout the experiments we will often take the nonlinearity in $F_l$ to be ReLU in which case the activation function corresponds to the parametric ReLU, for which the term $\frac{\partial C}{\partial \alpha_l}$ can be efficiently computed (He et al., 2015). Although not strictly required, we clip the values $\alpha_l$ to ensure they remain in the interval $[0,1]^n$.

From equation (6), we have that the gradients with respect to Lagrange multipliers are of the form:

$$\lambda_l \leftarrow \lambda_l + \eta' (\alpha_l - 1) \quad \text{(8)}$$

We note that since we require $\alpha_l \in [0,1]^n$, the values of $\lambda_l$ are monotonically decreasing. As the value of Lagrange multiplier decreases, this in turn pushes $\alpha_l$ towards 1 in equation (7). We set the step-size for the Lagrange multipliers, $\eta'$, to be a fraction of $\eta$. The motivation behind such a choice is to allow the network to adjust as we enforce the constraint on $\alpha_l$.

### 3.1 Intuitions from a Toy Model

To provide concrete intuition for the proposed method we consider training a toy network consisting of a single sigmoidal unit. Consider a network mapping scalar $x$ to $F(x,w) = \tanh(wx)$ and propose to learn scalar $w$ via traditional gradient descent with the following cost:

$$C(w) = \frac{1}{2} \| y - F(x,w) \|^2_2.$$  

It is evident from the form of the $\tanh$ function that minimizing such a loss will suffer from vanishing gradients when $wx$ is large. Indeed,

$$\frac{dC}{dw} = -x(y - \tanh(wx)) \sech(wx)^2 \to 0 \quad \text{as} \quad wx \to \pm \infty,$$

hence gradients may vanish due to poor initialization\(^2\) of $w$ or large input $x$, leading to slow convergence. The motivation behind the proposed method follows from the fact that linear activation functions do not suffer from vanishing gradients. Hence we recast our sigmoidal activation $F(x,w)$ as one extreme of a convex combination of the linear and $\tanh$ activations, parameterized by the scalar $\alpha$:

$$F(x,w,\alpha) = \alpha F(x,w) + (1 - \alpha) wx, \quad \alpha \in [0,1].$$

Thus at the $\alpha = 0$ bound $F(x,w,\alpha)$ reduces to the linear activation function, while at $\alpha = 1$ we recover the $\tanh$ activation function. The proposed method therefore considers

$$\hat{C}(w,\alpha) = \frac{1}{2} \| y - F(x,w,\alpha) \|^2_2,$$

subject to the constraint that $\alpha = 1$ at the end of training. Such a constraint is enforced by formulating the associated Lagrangian as

$$\mathcal{L}(w,\alpha,\lambda) = \hat{C}(w,\alpha) + \lambda(1 - \alpha).$$

which can be solve by iteratively applying the following update rules:

$$w \leftarrow w - \eta \frac{\partial \hat{C}}{\partial w}, \quad \alpha \leftarrow \alpha - \eta \left( \frac{\partial \hat{C}}{\partial \alpha} - \lambda \right), \quad \lambda \leftarrow \lambda - \eta (\alpha - 1).$$

In the limit, these dynamics yield $\alpha = 1$, $w = w^*$ at their stationary point, just like ordinary gradient descent on $C(w)$. The intuition motivating our approach is that by employing a constrained optimization approach we allow $\alpha$ temporary freedom to assume values less than 1, for which the activation function is more linear. This leads to gradients whose magnitude is larger thereby speeding up the process of learning. Figure\(^1\) demonstrates that this is indeed the case on some toy data.

\(^2\)We note that typical initialization schemes for weights, such as Xavier or He initialization, would avoid such degeneracy in this toy network. However, this is not generally the case in deeper networks.
0.0
0.5
1.0
1.5
2.0
0
1
100
101
102
103
104
105
iterations
0
1
100
101
102
103
104
105
iterations
0
2
4
6
|w w * |

Figure 1: Training a single unit on a single example ($x = 1; y = \tanh(3x)$). **Left:** True loss i.e. for $\alpha = 1$. **Middle:** Time course of $\alpha$. **Right:** Time course of weight error. Ordinary gradient descent (GD, green) is slow due to vanishing gradients. VAN (blue) is faster because $\alpha$ is allowed to be non-zero, allowing it to take a larger initial steps. Note that $\alpha$ dips briefly back towards zero because of the contribution of the $-\frac{\partial C(w, \alpha)}{\partial \alpha}$ to its dynamics, but is pushed back to 1 by the Lagrange multipliers. Optimizing $\alpha$ to minimize the true loss $C(w)$ rather than $\hat{C}(w, \alpha)$ results in the fastest convergence (orange).

Furthermore, in the context of a simple network such as the one considered here, we can compute the optimal value of $\alpha$ at each iteration. If we update $w$ according to the gradient of $C$ such that

$$\Delta w = -\eta \frac{\partial C(w, \alpha)}{\partial w},$$

the resulting first-order approximation to the change in the cost function is

$$\Delta \hat{C}(w, \alpha) = \frac{\partial C(w)}{\partial w} \Delta w = -\eta \frac{\partial C(w)}{\partial w} \frac{\partial \hat{C}(w, \alpha)}{\partial w}.$$ 

We can then define $\alpha(w)$ as the $\alpha \in [0, 1]$ that minimizes this first-order change in the true loss for some fixed $w$:

$$\alpha^*(w) \triangleq \arg\min_{\alpha \in [0, 1]} \Delta \hat{C}(w, \alpha) = \arg\max_{\alpha \in [0, 1]} \frac{\partial C(w)}{\partial w} \frac{\partial \hat{C}(w, \alpha)}{\partial w}.$$ 

Notice that only the second term depends on $\alpha$. Because the domain of maximization of $\alpha$ contains 1 as a special case we have that $\Delta \hat{C}(w, \alpha^*(w)) \leq \Delta \hat{C}(w, 1)$. Hence if $\alpha^*(w)$ can be found, descent according to the gradient of $\hat{C}(w, \alpha^*)$ would (locally) reduce the loss at least as much standard gradient descent.

For the case of a single tanh unit considered here, finding $\alpha^*(w)$ requires minimizing a simple quadratic over $[0, 1]$. Using this value of $\alpha^*(w)$ yields a dramatic improvement in convergence time (see Figure 1). However, for networks with multiple layers, the optimal choice of $\alpha$ values across layers interact leading to an intractable optimization problem. The use of Lagrange multipliers constitutes a computationally efficient alternative in such settings.

4 Experiments

We first demonstrate the capabilities of the proposed method using a simple, non-convolutional architecture on the MNIST and Fashion-MNIST datasets (Xiao et al. 2017) in Section 4.1. More extensive comparisons are then considered on the CIFAR datasets (Krizhevsky & Hinton 2009) in Section 4.2.

4.1 MNIST AND FASHION-MNIST

Networks of varying depths were trained on both MNIST and Fashion-MNIST datasets. Following Srivastava et al. (2015) the networks employed in this section were thin, with each layer containing 50 hidden units. In all networks the first layer was a fully connected plain layer followed by $l$ layers and a final softmax layer. The proposed method is benchmarked against several popular architectures such as ResNets and Highway Networks as well as the recently proposed DiracNets (Zagoruyko & Komodakis 2017). Plain networks without skip-connections are also considered. For all architectures the ReLU activation function was employed. For both ResNets and VAN the residual function consisted of batch-normalization followed by ReLU.
The depth of the network varied from $l = 1$ to $l = 30$ hidden layers. All networks were trained using SGD with momentum. The learning rate is fixed at 0.001 and the momentum parameter at 0.9. Training consisted of 50 epochs with a batch-size of 128. In the case of VAN networks the $\alpha_l$ values were initialized to $0.5 \cdot 1$ for all layers. As such, during the initial stages of training VAN networks had the same activation function as ResNets.

RESULTS

The results are shown in Figure 2, where the test accuracy is shown as a function of the network depth for both the MNIST and Fashion-MNIST datasets. In both cases we find that the performance of plain networks deteriorates significantly once the network depth exceeds some critical value (approximately 10 layers). As would be expected, this is not the case for ResNets, Highway Networks and DiracNets as such architectures have been explicitly designed to avoid this behavior. We note that VAN networks obtain competitive results across all depths. This is particularly true in the context of shallow networks ($l \leq 10$ layers) where VAN networks obtain the highest test accuracies. In particular, we note that VAN networks outperform plain networks in this context, suggesting that the introduction of variable skip-connections may lead to convergence to more desirable local optima.

4.2 CIFAR

As a more realistic benchmark we consider the CIFAR-10 and CIFAR-100 datasets. These consist of 60000 32×32 pixel color images with 10 and 100 classes respectively. The datasets are divided into 50000 training images and 10000 test images.

We follow He et al. (2016a) and train deep convolutional networks consisting of four blocks each consisting of $n$ residual layers. The residual function is of the form conv-BN-ReLU-conv-BN-ReLU. This corresponds to the pre-activation function (He et al., 2016b). The convolutional layers consist of $3 \times 3$ filters with downsampling at the beginning of blocks 2, 3 and 4. The network ends with a fully connected softmax layer, resulting in a depth of $8n + 2$. The architecture is described in Table 1.

Networks were trained using SGD with momentum over 165 epochs. The learning rate was set to 0.1 and divided by 10 at the 82nd and 125th epoch. The momentum parameter was set to 0.9. Networks were trained using mini-batches of size 128. Data augmentation followed Lee et al. (2015): this involved random cropping and horizontal flips. Weights were initialized following He et al. (2015).

As in Section 4.1, we initialize $\alpha_l = 0.5$ for all layers. On CIFAR-10 we ran experiments with $n \in \{1, 2, 3, 4, 5, 6, 8, 10\}$ yielding networks with depths ranging from 10 to 82. For CIFAR-100 experiments were run with $n \in \{1, 2, 3, 4\}$.

Figure 2: Results on MNIST (left) and fashion-MNIST (right) for various different architectures as the depth of the network varies from 1 to 30. Mean average test accuracy over 10 independent training sessions is shown. We note that with the exception of plain networks, the performance of all remaining architectures is stable as the number of layers increases.
RESULTS

Results for experiments on CIFAR-10 are shown in Figure 3. The left panel shows the mean test accuracy over five independent training sessions for ResNets, VANs and plain networks. While plain networks provide competitive results for networks with fewer than 30 layers, their performance quickly deteriorates thereafter. We note that a similar phenomenon is observed in VAN networks but the effect is not as dramatic. In particular, the performance of VANs is similar to ResNets for networks with up to 40 layers. Beyond this depth, ResNets outperform VAN by an increasing margin. This may be a result of the ensemble nature of ResNets (Veit et al., 2016), which does not play a significant role until the depth of the network increases. A comparison of the performance of VAN networks in provided in Table 2. We note that while VAN networks do not outperform ResNets, they do outperform other alternatives such as Highway networks and FitNets (Romero et al., 2014) when networks of similar depths considered. In the case of both VANs and Highway Networks the best performance is obtained with networks of 18 and 19 layers respectively while ResNets continue to improve their performance as depth increases.

Training curves for a plain and VAN network are shown on the right hand side of Figure 3. The thick gold line indicates the mean residuals of the $\alpha_l$ parameters across all layers. This is defined as $\frac{1}{L} \sum_{l=1}^{L} (1 - \alpha_l)^2$ and is a measure of the extent to which skip-connections are present in the network. Recall that if all $\alpha_l$ values are set to one then all skip-connections are removed (see equation (4)). From Figure 3 it follows that skip-connections are effectively removed from the VAN network at approximately the 80th iteration.

Table 1: Architecture of varying depth employed on CIFAR-10 and CIFAR-100 datasets. Downsampling occurs at the beginning of block 2, 3 and 4 with a stride of two.

<table>
<thead>
<tr>
<th>Layer name</th>
<th>Output size</th>
<th>Convolution</th>
</tr>
</thead>
<tbody>
<tr>
<td>conv1</td>
<td>32 × 32</td>
<td>3 × 3, 64</td>
</tr>
<tr>
<td>Block 1</td>
<td>32 × 32</td>
<td>3 × 3, 64 × n</td>
</tr>
<tr>
<td>Block 2</td>
<td>16 × 16</td>
<td>3 × 3, 128 × n</td>
</tr>
<tr>
<td>Block 3</td>
<td>8 × 8</td>
<td>3 × 3, 256 × n</td>
</tr>
<tr>
<td>Block 4</td>
<td>4 × 4</td>
<td>3 × 3, 512 × n</td>
</tr>
</tbody>
</table>

Figure 4 provides results on the CIFAR-100 dataset. This dataset is considerably more challenging as it consists of a larger number of classes as well as fewer examples per class. As in the case of...
Figure 4: **Left:** Results on CIFAR-100 dataset are shown as the depth increases from 10 to 34 layers. We note that the performance of both VAN and plain networks deteriorates as the depth increases, but the effect is far less pronounced for plain networks. **Right:** Training and test error curves are shown for VAN and plain networks with 18 layers. The mean $\alpha$ residuals, $\frac{1}{L} \sum_{l=1}^{L} (1 - \alpha_l)^2$, are shown in gold along the right axis.

CIFAR-10, we observe a fall in the performance of both VAN and plain networks beyond a certain depth (in this case approximately 20 layers). Despite this drop in performance, Table 2 indicates that the performance of VAN networks with both 18 and 26 layers is competitive with many alternatives proposed in the literature such as Highway Networks and FitNets.

Training curves are shown on the right hand side of Figure 4. As in the equivalent plot for CIFAR-10, the introduction and subsequent removal of skip-connections during training leads to improvements in generalization error.

Table 2: Comparison of VAN networks to results of other convolutional networks on CIFAR-10 and CIFAR-100. For VAN networks we report the best value as well as the mean and standard deviation over five independent training runs (in brackets). Results are ordered by performance on CIFAR-10.

<table>
<thead>
<tr>
<th>Architecture</th>
<th># Layers</th>
<th>CIFAR-10 (test error %)</th>
<th>CIFAR-100 (test error %)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Highway Network</td>
<td>32</td>
<td>8.80</td>
<td>-</td>
</tr>
<tr>
<td>FitNet</td>
<td>19</td>
<td>8.39</td>
<td>35.04</td>
</tr>
<tr>
<td>Highway Network</td>
<td>19</td>
<td>7.54</td>
<td>32.39</td>
</tr>
<tr>
<td>DiracNet (width-1)</td>
<td>34</td>
<td>7.10</td>
<td>-</td>
</tr>
<tr>
<td>ELU</td>
<td>18</td>
<td>6.55</td>
<td>24.28</td>
</tr>
<tr>
<td>VAN (ours)</td>
<td>26</td>
<td>6.22 (6.55 ± 0.24)</td>
<td>28.31 (29.14 ± 0.48)</td>
</tr>
<tr>
<td>VAN (ours)</td>
<td>18</td>
<td>6.17 (6.35 ± 0.13)</td>
<td>27.15 (27.45 ± 0.41)</td>
</tr>
<tr>
<td>DiracNet (width-2)</td>
<td>34</td>
<td>5.60</td>
<td>26.72</td>
</tr>
<tr>
<td>ResNet</td>
<td>164</td>
<td>5.46</td>
<td>24.33</td>
</tr>
</tbody>
</table>

5 DISCUSSION

This manuscript presents a simple method for training deep feed-forward networks without skip-connections. The goal of the proposed method is to exploit many of the optimization benefits associated with skip-connections when training plain networks. This is achieved by posing network training as a constrained optimization problem where skip-connections are introduced during the early stages of training and subsequently phased out in a principled manner using Lagrange multipliers. Throughout a series of experiments we demonstrate that VAN networks obtain competitive results on both the CIFAR-10 and CIFAR-100 datasets.

The method proposed in this work has focused on networks without skip-connections due to the well-documented optimization challenges associated with training such networks, and to emphasize the ability of this approach to address such issues. However, the proposed method is quite general and can be readily extended to networks which do allow for skip-connections.
REFERENCES


