Residual Gates: A Simple Mechanism for Improved Network Optimization

Pedro Savarese [1] **Daniel Figueiredo** [1]

Abstract

A fundamental idea behind all state-of-the-art deep network models are shortcut connections which allow layers to perform identity mappings, passing their input almost unaltered to their output. Although this functionality allows information to flow more freely through the network, the details of the shortcut connection have profound impact on the capacity of learning identity mappings. This paper proposes *Residual Gates*, a novel and simple gating mechanism for shortcut connections that can be applied to any network model. The proposed design leverages two aspects: a single scalar parameter controls the gate; the new output is a weighted sum of the input and the original layer's output. A diverse set of experimental results show the advantages of *Residual Gates*, including superior performance in all models and all datasets evaluated (yielding competitive results for CIFAR-10 and CIFAR-100), robustness to layer removal on the trained model (only 8% accuracy decrease after removing 60% of layers), competitive performance without adopting batch normalization. The paramount success of *Residual Gates* stems from the simplicity in learning identity mappings, allowing the optimizer to push information much more freely through the network and, thus learn better representations.

1. Introduction

Recent advancements in neural networks have showcased the unprecedented success in increasing the network's depth. Deep network models such as Residual Networks (He et al., 2016a) and Highway Neural Networks (Srivastava et al., 2015) allow designing networks with hundreds of layers, and have led to breakthroughs in several computer vision applications.

The potential benefits of increasing network depth seems unquestionable, supported by both theoretical and practical findings. On the theoretical side, increasing a network's depth contributes exponentially more to its representational capacity in comparison to increasing its width (Eldan & Shamir, 2016; Telgarsky, 2016; Bianchini & Scarselli, 2014; Montúfar et al., 2014). Moreover, most models allow layers to learn the identity function, making it possible to increase depth without compromising performance (He et al., 2016a). In practice, deep networks have been highly successful in many Machine Learning applications, and are the leading architectures on competitions such as ILSVRC (He et al., 2016a; Szegedy et al., 2015).

These observations may suggest that stacking more layers to a network will always lead to performance improvements. However, this behavior is not observed in practice, even for recently proposed models. The problem lies in the fundamental challenge of training deep networks. There are many obstacles to an effective training, including vanishing/exploding gradients (Bengio et al., 1994) due to saturating activation functions and poor weight initialization (Glorot & Bengio, 2010a). Techniques such as unsupervised pre-training (Bengio et al., 2016), non-saturating activation functions (Nair & Hinton, 2010) and normalization (Ioffe & Szegedy, 2015) target these issues, but alone are insufficient for training networks with over a hundred layers.

A key idea behind recent deep network models, such as Residual Networks (He et al., 2016a) and Highway Neural Networks (Srivastava et al., 2015), is to allow information to flow more freely through the network. Usually achieved by using a shortcut connection between layers, this clever layer design greatly facilitates optimization due to shorter paths between the lower layers and the network's error function, providing stronger supervision (Huang et al.). In particular, these models can more easily learn *identity mappings*, converting the layer's input into an output that is almost unaltered. As a consequence, these much deeper networks can learn more abstract representations, and have been highly successful in many computer vision tasks (Feichtenhofer et al., 2016; Dai et al., 2016).

Although present in such recent models, the operational details of shortcut connections vary significantly among them. For example, in Highway Neural Networks the short-

[1]Federal University of Rio de Janeiro (UFRJ), Brazil. Correspondence to: Pedro Savarese <savarese@land.ufrj.br>.

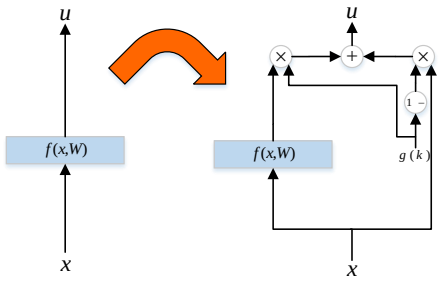

Figure 1. Adding *Residual Gates* to a plain network. The key difference with Highway Neural Networks is that only a scalar (k) is used to regulate the gates instead of a tensor.

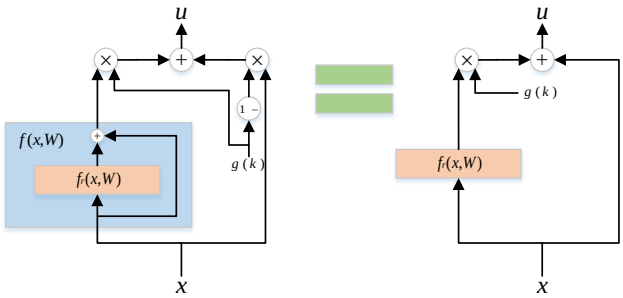

Figure 2. Adding *Residual Gates* to a ResNet. Note that the resultant layers have unaltered shortcut connections, while $g(k)$ modulates the residuals $f_r(x, W)$.

cut connection suffers a non-linear transformation that depends on additional parameters and incoming data, while in pre-activation Residual Networks the input traverses the shortcut connection unaltered. In particular, the details of shortcut connections has profound impact on learning identity mappings. In Highway Neural Networks, learning a true identity mapping is non-trivial since the gating mechanism depends on the incoming data, while for Residual Networks it suffices to learn an all-zero tensor.

Intuitively, shortcut connections should facilitate learning identity mappings without hindering network performance. This work approaches this problem by proposing *Residual Gates*, a novel mechanism to handle shortcut connections. It consist of a simple linear gating mechanism which is added to a network layer, as illustrated in Figure 1. It introduces a single scalar parameter, k, and can be applied to any network model, including Residual Networks. Note that both the shortcut and residual connections are controlled by gates parameterized by a scalar k. When $g(k) = 0$ we have a true identity mapping, while when $g(k) = 1$ the shortcut connection does not contribute to the output.

Figure 2 illustrates Residual Gates used on ResNets. Note that the residual layer simply becomes $u = g(k)f_r(x, W) + x$, where f_r denotes the layer's residual function. Thus, the shortcut connection allows the input to flow freely through the layer without any interference from $g(k)$. However, the residual function f_r is modulated by $g(k)$ which can amplify or diminish its signal relative to the shortcut.

In what follows we provide empirical and theoretical support for the advantages of *Residual Gates*. In particular, we show:

- Easier learning of identity mappings: in comparison to Highway Neural Networks and Residual Networks, *Residual Gates* can more easily learn identity mappings. We provide some theoretical support for this intuitive argument in Section 2.
- Superior performance: adding *Residual Gates* improves the performance of network models. This was observed in all different models (Plain Networks, Residual Networks, and Wide Residual Networks), different network depths, and different datasets (MNIST, CIFAR-10, and CIFAR-100) evaluated (see Section 3).
- Robustness to layer removal: *Residual Gates* increase the robustness of networks to layer removal. In a particular example with 50 residual blocks (100 layers), randomly removing 30 blocks (60 layers) of the trained network only leads to a relative decrease of 8% accuracy (see Section 3.1).
- Insensitivity to batch normalization: *Residual Gates* diminish the role of batch normalization and shows competitive performance in models where absolutely no normalization is used (see Section 3.2). Its simple and flexible design allows the optimizer to learn the adequate modulation for residuals.

We conclude the paper in Section 4 with a brief discussion on the importance of learning identity mappings and effective designs for shortcut connection mechanisms.

2. Residual Gates

As previously discussed, Residual Gates leverages the idea of shortcut connections but with a simple weighted linear combination between the original layer's output and input, as illustrated in Figure 1. More formally, adding Residual Gates to a layer results in the following formulation:

$$u = g(k)f(x, W) + (1 - g(k))x \tag{1}$$

where $f(x, W)$ is the output of the original layer, $g(k)$ is the gating function, and k is the gate parameter, a scalar

value. Note that the new output u is a simple linear combination between the previous output and the input, weighted by the gating function $g(\cdot)$.

With the proposed mechanism, a layer can degenerate and perform an identity mapping when $g(k) = 0$. Using the ReLU activation function as g, it suffices that $k \leq 0$ for $g(k) = 0$. Of course, the best value of k for each layer will also be learned during the optimization procedure, along with the layers' other original parameters.

Note that adding a single scalar parameter to a layer increases the dimensionality of the cost surface by one. This new dimension, however, can be easily understood due to the specific nature of the proposed mechanism. The original surface is maintained on the $g(k) = 1$ slice, since models with *Residual Gates* become equivalent to original ones. On the $g(k) = 0$ slice we have an identity mapping since $u = x$. Under this last condition, the associated cost for all points in this slice is the same as for the point $\{g(k) = 1, W = I\}$: this follows since both parameter configurations correspond to identity mappings, therefore being equivalent. Last, assuming $g(k) = k$ for $0 \leq k \leq 1$, such as with ReLU, we have a linear combination with weight k, and consequently, all other slices $0 < k < 1$ will be a linear combination of the slices $k = 0$ and $k = 1$. Thus, the addition of the parameter k has a very specific impact on the cost surface of the original model: it adds a new axis that corresponds to a linear removal of its corresponding layer.

Also as previously mentioned, the proposed mechanism can be applied to Residual Networks, as illustrated in Figure 2. Although it may appear counterintuitive to add a shortcut connection with a gating mechanism to a layer that already has a shortcut connection, such augmentation provides a simple means to regulate the residuals, as we now discuss. Let the original design of a residual layer be:

$$u = f(x, W) = f_r(x, W) + x$$

where $f_r(x, W)$ is the layer's residual function – in our case, **BN-ReLU-Conv-BN-ReLU-Conv**. Adding *Residual Gates* to this layer yields:

$$\begin{aligned} u &= g(k)f(x, W) + (1 - g(k))x \\ &= g(k)(f_r(x, W) + x) + (1 - g(k))x \\ &= g(k)f_r(x, W) + x \end{aligned}$$

The resulting layer maintains the shortcut connection unaltered, granting free gradient flow through the network, a much desired property when designing network models (He et al., 2016b). As $(1 - g(k))$ vanishes from the formulation, $g(k)$ stops acting as a gating mechanism and can be interpreted as a flow regulator. Note that $g(k)$ modulates the residual signal $f_r(x, W)$ and can promote or demote it with respect to the shortcut connection, x. This intuitively provides better and easier control for each residual block, that now relies on a single scalar parameter, k, to regulate the relative importance between the shortcut and the residual. As we will see, this simple feature yields significant benefits in practice.

2.1. Theoretical observations

As briefly discussed, shortcut connections on the layer design of both Highway Neural Networks and Residual Networks allows much deeper models to be trained. In particular, shortcut connections facilitate the learning of (near) identity mappings but strongly depend on their specific mechanics. For Residual Networks a true identity mapping in a layer is learned when the corresponding $W = 0$ (and not $W = I$), since there is no gating. For Highway Networks, identity mappings are learned when the gating term $T(x, W_T) = 0$ for all input x. However, this condition strongly depends on the choice of the transform function T, and is non-trivial when T is the sigmoid function, since $T^{-1}(0)$ is not defined. Thus, intuitively, learning identity mappings on Highway Nets seems reasonably harder than in Residual Nets.

We thus focus on Residual Networks and analyze the difficulty for them to learn (true) identity mappings. Considering a fully-connected residual layer $u = ReLU(\langle x, W\rangle) + x$, we have (for $W = 0$):

$$u = ReLU(\langle x, 0\rangle) + x = ReLU(0) + x = x$$

Intuitively, residual layers can degenerate into identity mappings more effectively (in comparison to Plain Networks) since learning an all-zero matrix is easier than learning the identity matrix (required for identity mappings in Plain Networks). To support this argument, consider weight parameters randomly initialized with zero mean but non-zero variance. Hence, the point $W = 0$ is the expected value for the random weights used in the initialization, and thus closer to the actual values than the point $W = I$. Moreover, recent work ((Zhang et al.)) suggests that the L2 norm of a critical point is an important factor regarding the difficulty for an optimizer to reach it. Thus, $W = 0$ is more accessible for the optimizer than $W = I$ due its smaller L2 norm. In a nutshell, by having the identity mapping occur at $W = 0$ as opposed to $W = I$, Residual Networks have facilitated the learning of identity mappings.

However, assuming that residual layers can trivially learn $W = 0$ implies ignoring the randomness in the weight initialization scheme. We demonstrate this caveat by calculating the expected distance between W_o and the origin ($W = 0$), where W_o denotes the weight tensor right after initialization and prior to any optimization. The expected distance between W_o and the origin captures the effort for

a network to learn identity mappings. Let μ_o and σ_o^2 denote the mean and variance of the random variable W_o. The expected distance to the origin is given by:

$$E\Big[(W_o - 0)^2\Big] = E\Big[W_o^2\Big] = \sigma_o^2 + \mu_o^2$$

However, there is no reason to assume that the variance (and mean) are negligible. In particular, recent initialization schemes propose using $\mu_o = 0$ and $\sigma_o^2 = \Theta(\frac{1}{n})$ ((Glorot & Bengio, 2010b), (He et al.)), where n is the size of the layer's input (e.g. number of feature maps for constitutional layers). However this represents the variance for each individual element of W_o, as each component is initialized independently from one another. Thus, for tensors with $\Theta(n^2)$ parameters (e.g. weight matrices in fully-connected networks), the component wise sum of the expected distance and the expected Euclidean distance between W_o and the origin are $\Theta(n)$, and thus not negligible.

In contrast. *Residual Gates* allows identity mappings to be learned by adjusting just a single scalar parameter, k. In particular, when $g(\cdot) = 0$ the output is equal to the input. Assuming that $g(\cdot) = 0$ for $k = 0$, such as with ReLU or $g(k) = k$, we immediately observe that the effort in learning identity mappings (as measured by the distance between the initialized value for k and $k = 0$) does not depend on any model parameters, such as the size of the layer's input.

Initializing k either randomly (as long as the mean and variance do not depend on model parameters) or with a constant value leads to a distance to the origin ($k = 0$) of $\Theta(1)$. This observation is in sharp contrast with prior models, including Residual Networks. In particular, it indicates that learning identity mappings is much easier under the proposed gating mechanism, an observation soon supported by empirical results.

3. Experiments

All models were implemented on Keras (Chollet, 2015) or Torch (Collobert et al., 2011), and were executed on a Geforce GTX 1070. Larger models or more complex datasets, such as the ImageNet (Russakovsky et al., 2015), were not explored due to hardware limitations.

3.1. MNIST

The MNIST dataset (Lecun et al., 1998) is composed of $60,000$ greyscale images with 28×28 pixels. For preprocessing, we divided each pixel value by 255, normalizing their values to $[0, 1]$. Images represent handwritten digits, resulting in a total of 10 classes. We explore *Residual Gates* by applying it to two fully-connected models: classic Plain Networks (PlainNets) and ResNets. We compare the performance of the original models with the models augmented with *Residual Gates*, naming them G-PlainNets and G-ResNets.

Table 1. Test error (%) on the MNIST dataset for fully-connected networks for different network depths (d). Values in parenthesis show relative gains of gated models over their original counterpart (Plain and Res denote PlainNets and ResNets, respectively).

d	PLAIN	RES	G-PLAINNETS	G-RESNETS
2	2.29	2.20	2.04 (11%)	2.17 (1.4%)
10	2.22	1.64	1.78 (20%)	1.60 (2.4%)
20	2.21	1.61	1.59 (28%)	1.57 (2.4%)
50	60.4	1.62	1.36 (∞)	1.48 (8.6%)
100	90.2	1.50	1.29 (∞)	1.26 (16%)

The networks consist of a linear layer with 50 neurons, followed by d layers with 50 neurons each, and lastly a softmax layer for classification. Only the d middle layers differ between the four models – the first linear layer and the softmax layer are the same in all experiments.

For PlainNets, each layer performs dot product, followed by Batch Normalization and a ReLU activation function. A for ResNets, initial tests with pre-activations (He et al., 2016b) resulted in poor performance on the validation set, therefore we opted for the traditional **Dot-BN-ReLU** layer structure for ResNets. Each residual block consists of two layers, as conventional.

All networks were trained using Adam (Kingma & Ba) with Nesterov momentum (Dozat) for a total of 100 epochs using mini-batches of size 128. No learning rate decay was used: we kept the learning rate and momentum fixed to 0.002 and 0.9 during the entire training.

3.1.1. IMPACT OF DEPTH

The training curves for PlainNets, G-PlainNets, ResNets and G-ResNets with varying network depths are shown in Figure 3. Note that the distance between the curves increase with network depth, indicating that *Residual Gates* is more effective in deeper models. Moreover, PlainNets fail to converge for $d = 50$ and $d = 100$ (not shown in the figure) while its gated counterpart (G-PlainNets) has competitive performance.

Table 1 shows the test error for each depth and model. Note that the gated models outperform their original counterparts in all experiments. Moreover, performance gains increase with network depth, again indicating that the benefits of *Residual Gates* are larger for deeper networks. Interestingly, G-PlainNets outperformed ResNets on networks with more depth ($d = 20$ or higher), suggesting that the simple regulatory nature of the residual in *Residual Gates* is effective.

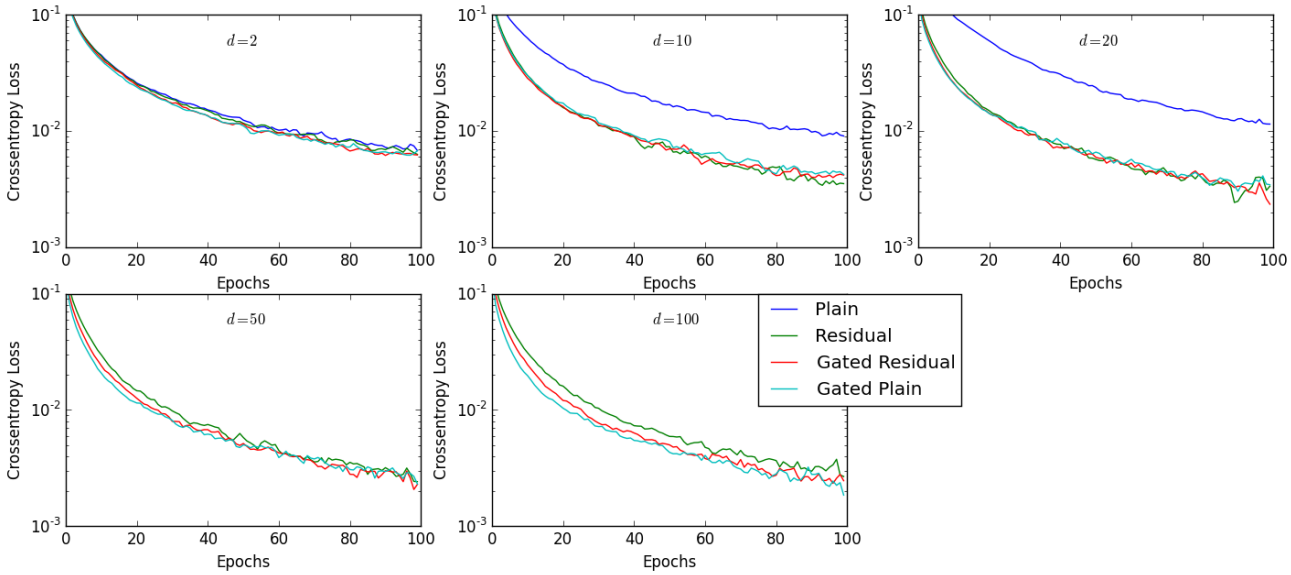

Figure 3. Train loss for PlainNets and ResNets, along with their gated counterparts, with $d = \{2, 10, 20, 50, 100\}$.

Table 2. Average k across all layers for G-PlainNets and G-ResNets for different depths.

DEPTH	G-PLAINNETS	G-RESNETS
$d = 2$	10.57	5.58
$d = 10$	1.19	2.54
$d = 20$	0.64	1.73
$d = 50$	0.46	1.04
$d = 100$	0.41	0.67

An important consideration are the values for parameter k learned for each network layer. Table 4 shows the average k value (across all layers) for different network depths. Note that for both models, the average k decreases as the network becomes deeper. This important observation suggests that in order for deep networks to be effective, higher layers should receive enough information from lower layers. Indeed, learning smaller k allows information to flow more freely to through the network, providing more stability to gradient signals during backpropagation. In fact, this is consistent with the intuition for why ResNets outperform PlainNets as the depth increases (namely, the shortcut connections allow information to flow more freely).

Last, the significant difference between average k values for G-PlainNets and G-ResNets has an intuitive explanation: in order to suppress the residual signal against the shortcut connection, G-PlainNets require that $k < 0.5$ (otherwise the residual signal will be enhanced). Conversely, G-ResNets suppresses the residual signal when $k < 1.0$, and enhance it otherwise.

Beyond average values, it is interesting to consider the structure induced by the learned values for k across the layers of a deep network. Figure 4 shows the final values for k for our deepest network, $d = 100$ (50 different values, as each block has two layers), for each layer. Note that k first tends to decrease and then increase towards the end of the network. Interestingly, the middle layers tend to have the smallest k values, and is thus the region where information flows more freely in the network.

3.1.2. ROBUSTNESS TO LAYER REMOVAL

We consider the impact on performance when removing layers in ResNets and G-ResNets after the model has been trained. Intuitively, a well-trained deep network should learn representations in small refinements (Greff et al.), thus removing layers should not significantly impact performance.

In order to assess this intuition, we consider the test performance of our deepest network ($d = 100$) as residual blocks are completely removed from the trained network. We consider two strategies for block removal: greedy and random. In the greedy strategy blocks are removed in increasing order of k, while in the random strategy a block is chosen uniformly at random to be removed. Thus, the greedy strategy removes first blocks that are intuitively less important, since they are closer to performing a true identity mappings. Note that since ResNets lacks the parameter k, we consider only the random strategy for this model.

Results are shown in Figure 5, comparing both models and both strategies (for G-ResNets). Note that G-ResNets is extremely robust to layer removal. In particular, accuracy

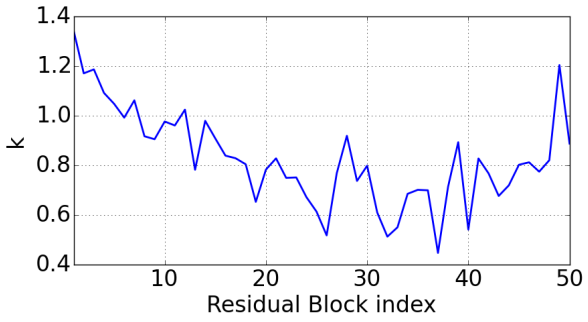

Figure 4. Values for k according to ascending order of residual blocks. The first block, consisted of the first two layers of the network, has index 1, while the last block – right before the softmax layer – has index 50.

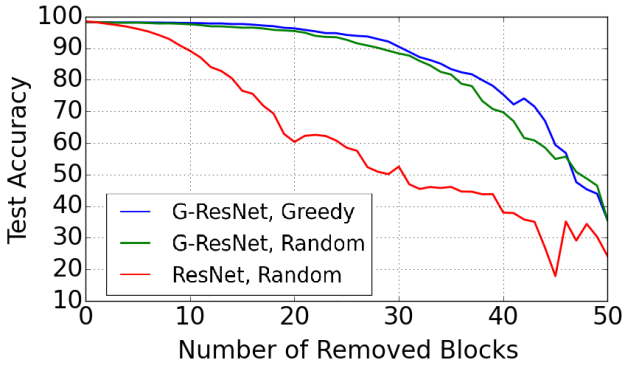

Figure 5. Test accuracy (%) according to the number of removed layers. Gated Residual Networks are more robust to layer removal, and maintain decent results even after half of the layers have been removed.

is above 96% even after removing 40% of the layers. More surprisingly, there is very little difference between greedy and random removal for G-ResNets when removing half of the layers. This shows that G-ResNets has learned redundant representations across the network and not just in particular layers, and can thus operate under severe layer removal. The findings are quite different for ResNets, as accuracy decays much faster with random layer removal. Note that accuracy is only around 60% when removing 40% of the layers (20 blocks).

These findings indicate that *Residual Gates* generate networks that are not only robust to layer removal but can deliver good performance even when a significant fraction of the layers of the trained network is removed. Thus, a trained deep network can be pruned to make much faster predictions (say, in 60% of the time after removing 40% of the layers) while still delivering good performance.

3.2. CIFAR

The CIFAR datasets (Krizhevsky, 2009) consists of $60,000$ color images with 32×32 pixels each. CIFAR-10 has a total of 10 classes, including pictures of cats, birds and airplanes. The CIFAR-100 dataset is composed of the same number of images, however with a total of 100 classes. ResNets are currently the state-of-the-art methodology for classifying the CIFAR dataset. We apply the proposed gating mechanism to ResNets and Wide ResNets (Zagoruyko & Komodakis, 2016) and compare the performance with their original counterpart.

For pre-activation ResNets, as described in (He et al., 2016b), we follow the original implementation details. We set an initial learning rate of 0.1, and decrease it by a factor of 10 after 50% and 75% epochs. SGD with Nesterov momentum of 0.9 is used for optimization, and the only pre-processing consists of mean subtraction. Weight decay of 0.0001 is used for regularization, and Batch Normalization's momentum is set to 0.9.

Table 3. Test error (%) on the CIFAR-10 dataset for ResNets, Wide ResNets and their gated counterparts (value in parenthesis indicates relative gains with respect to original counterpart). k decay is when weight decay is also applied to the k parameters in the gated network. Results for ResNets and Wide ResNets are as reported in the original paper (He et al., 2016a; Zagoruyko & Komodakis, 2016).

MODEL	ORIGINAL	GATED	k DECAY
RESNETS 5	7.16	**6.67** (6.8%)	7.04
WIDE RESNETS (4,10) + DROPOUT	3.89	**3.65** (6.2%)	3.74

We follow the implementation from (Zagoruyko & Komodakis, 2016) for Wide ResNets. The learning rate is initialized as 0.1, and decreases by a factor of 5 after 30%, 60% and 80% epochs. Images are mean/std normalized, and a weight decay of 0.0005 is used for regularization. We also apply 0.3 dropout (Srivastava et al., 2014) between convolutions, whenever specified. All other details are the same as for ResNets.

We use moderate data augmentation for both models: images are padded with 4 pixels, and we take random crops of size 32×32 during training. Additionally, each image is horizontally flipped with 50% probability. We use a mini-batch size 128 for all experiments.

For all gated networks, we initialize k with a constant value of 1. An important consideration is whether weight decay should also be applied to the parameter k. We will explore weight decay for parameter k using the same magnitude as for the rest of the model parameters: 0.0001 for G-ResNet (Gated ResNets) and 0.0005 for G-WResNet (Gated Wide ResNets).

Table 3 shows the test error for the two models: ResNets

Table 4. Test error (%) on the CIFAR-10 dataset for Wide ResNets and its gated counterpart. Results for Wide ResNets are from the original paper (Zagoruyko & Komodakis, 2016).

MODEL	ORIGINAL	GATED
WIDE RESNETS (2,4)	5.02	**4.66**
WIDE RESNETS (4,10)	4.00	**3.82**
WIDE RESNETS (4,10) + DROPOUT	3.89	**3.65**
WIDE RESNETS (8,1)	6.43	**6.10**
WIDE RESNETS (6,10) + DROPOUT	3.80	**3.63**

Table 5. Test error (%) on the CIFAR-100 dataset for Wide ResNets and its gated counterpart. Results for Wide ResNets are from the original paper (Zagoruyko & Komodakis, 2016).

MODEL	ORIGINAL	GATED
WIDE RESNETS (2,4)	24.03	**23.29**
WIDE RESNETS (4,10)	19.25	**18.89**
WIDE RESNETS (4,10) + DROPOUT	18.85	**18.27**
WIDE RESNETS (8,1)	29.89	**28.20**

with $n = 5$ and Wide ResNets with $n = 4$ and widening factor of 10, along with their gated counterparts. Note that for both models, their gated counterparts exhibited superior performance, surpassing by around 6%. Interestingly, the gated counterparts with weight decay for parameter k also exhibited superior performance, but not as high as without regularization. This indicates that regularization of parameter k should be more subtle, as a strong regularization is promoting identity mappings at the layers, which can harm the model. Thus, for the remainder of the experiments we do not consider any regularization for parameter k. Last, note that the gated model adds only 15 and 12 parameters (a different k for each block) to their respective original counterparts which have millions of parameters.

We proceed to evaluate other models – having different depths and widening factors – to evaluate the effectiveness and robustness of *Residual Gates*. Tables 4 and 5 show that G-WResNets outperforms the original counterpart in all scenarios without changing any hyperparameter, for both CIFAR-10 and CIFAR-100 datasets. This is a strong evidence of the advantages in the workings of *Residual Gates*.

Figure 6 shows the loss curves for G-WResNets(4,10) + Dropout, both on CIFAR-10 and CIFAR-100. The optimization behaves similarly to the original model (Zagoruyko & Komodakis, 2016), suggesting that *Residual Gates* does not have any significant side effects on the network's training, beyond yielding an overall superior performance.

We again consider the final k values for each block of the network to understand the behavior of the gating mechanism. Figure 7 shows the result for G-WResNets (4,10) on CIFAR-10. Note that k values follow an intriguing pattern: the lowest values are for the blocks 1, 5 and 9, which are exactly the ones that decrease the spatial size of feature maps within the network. This indicates that the projection in the shortcut connection is enhanced relative to the non-linear transformations in the residual path. Interestingly, this indicates that pure projections (without non-linearities) might be better suited for spatial size reduction. Last, the peak value for the last residual block suggests that its shortcut connection is of very little importance. Indeed, this block generates the final representations which are used by the softmax layer.

As a final comparison, Table 6 shows the results of prior models in increasing order of performance for both the CIFAR-10 and CIFAR-100 datasets. Note that *Residual Gates* applied to Wide ResNet (4,10) + Dropout yields the best performance among these very recent works and thus competitive with state-of-the-art. The corresponding training and test error for this network is shown in Figure 6.

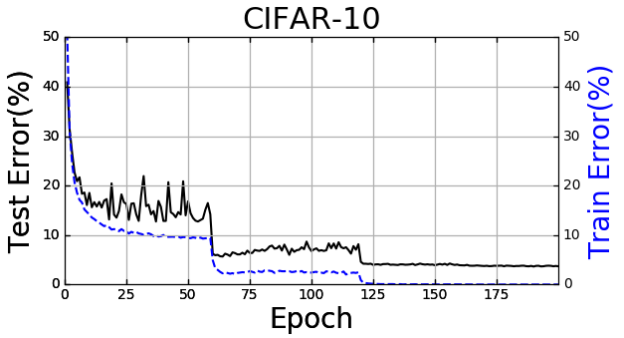

Figure 6. Training and test curves for G-WResNets (4,10) with 0.3 Dropout, showing error (%) on training (dashed lines) and test (solid lines) sets for CIFAR-10.

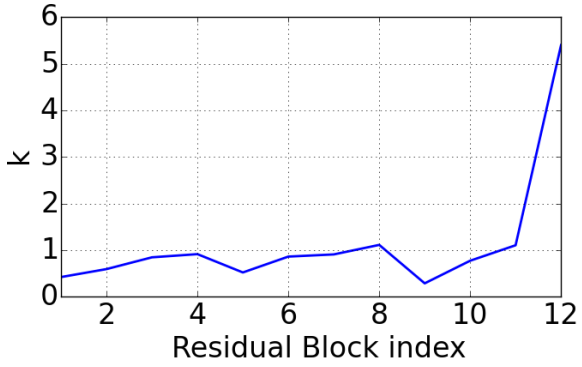

Figure 7. Learned values for k for the various network layers (blocks) for the G-WResNets (4,10) model on CIFAR-10.

Table 6. Test error (%) for various networks on the CIFAR-10 and CIFAR-100 datasets. All results are with standard data augmentation (crops and flips), as described in the text.

METHOD	C10+	C100+
NETWORK IN NETWORK (CHEN ET AL., 2016)	8.81	-
FITNET (ROMERO ET AL.)	8.39	35.04
HIGHWAY NEURAL NETWORK (SRIVASTAVA ET AL., 2015)	7.76	32.39
ALL-CNN (SPRINGENBERG ET AL.)	7.25	33.71
RESNET-110 (HE ET AL., 2016A)	6.61	-
RESNET IN RESNET (TARG ET AL.)	5.01	22.90
STOCHASTIC DEPTH (HUANG ET AL., 2016)	4.91	-
RESNET-1001 (HE ET AL., 2016B)	4.62	22.71
WIDE RESNET (4,10) (ZAGORUYKO & KOMODAKIS, 2016)	3.89	18.85
DENSENET (HUANG ET AL.)	3.74	19.25
G-WIDE RESNET (4,10) + DROPOUT	**3.65**	**18.27**

3.2.1. IMPACT OF NORMALIZATION

In order to further assess the optimization advantages of *Residual Gates*, we investigate the model without using any layer normalization. Particularly, batch normalization combats the internal co-variate shift in deep networks, and has been widely used to train networks with many layers.

With the removal of batch normalization, we adopt the original residual block composed of **Conv-ReLU-Conv-ReLU**, with ReLU before addition (He et al., 2016a). To each model considered, we apply *Residual Gates*, introducing a single parameter k per block.

Table 7 shows results for three different Wide ResNets models on CIFAR-10. Original models fail to converge, yielding the highest possible error even with He initialization (He et al.). In sharp contrast, the gated models achieve excellent performance, comparable to their counterparts with batch normalization (see Table 4). Surprisingly, our gating mechanism seems to diminish the need for normalization layers, even though it does not directly address the internal co-variate shift problem. Unlike batch normalization, our technique does not require the computation of moving averages, nor adds computational complexity to the running time.

Table 7. Test error (%) on the CIFAR-10 dataset for Wide ResNets models with no batch normalization and their gated counterparts.

MODEL	ORIGINAL	GATED
WIDE RESNETS (2,4)	90.0	**5.36**
WIDE RESNETS (4,10)	90.0	**4.48**
WIDE RESNETS (8,1)	90.0	**6.57**

4. Conclusion and discussion

Shortcut connections have shown to be instrumental to deep networks as they facilitate training by allowing layers to more easily learn identity mappings. The key idea is that shortcut connections allow information to flow more freely through the network, thus allowing the backpropagated error to play a more pronounced role in lower layers. However, there are various mechanisms to establish shortcut connections, such as the proposed by Residual Networks and Highway Neural Networks. These mechanisms have a significant impact on the learning ability of the network, as they regulate information flow.

This paper introduced *Residual Gates*, a simple gating mechanism that can be applied to any network model, including Residual Networks. The simplicity of the proposed mechanisms leverages two key ideas: (1) single scalar parameter to regulate each gate; (2) output is a weighted sum of the shortcut connection with the corresponding residual.

A diverse set of experimental results indicate the various and significant advantages of *Residual Gates*. In all scenarios, introducing *Residual Gates* to the network model led to superior performance, including for different models (PlainNets, ResNets and Wide ResNets) and datasets (MNIST, CIFAR-10, CIFAR-100). For CIFAR-10 and CIFAR-100, performance was competitive with the state-of-the-art. Networks trained with the gating mechanism are also much more robust to layer removal, with performance decaying slowly even if layers are randomly removed. Last, the proposed gating mechanism challenges the need of batch normalization, as gated networks with no normalization layers exhibited performance comparable to their original non-gated, batch normalized counterparts.

Why is this so? The key to understanding the benefits observed with *Residual Gates* lies behind the difficulty in learning identity mappings, we believe. If learning identity mappings is easier, then the optimization can more easily push signals from lower layers to higher ones, with benefits increasing with the network depth. Thus, it does not suffice that a network model enables the learning of identity mappings – they should also be easy to learn.

References

Bengio, Yoshua, Simard, Patrice Y., and Frasconi, Paolo. Learning long-term dependencies with gradient descent is difficult. *IEEE Trans. Neural Networks*, 5(2), 1994.

Bengio, Yoshua, Lamblin, Pascal, Popovici, Dan, and Larochelle, Hugo. Greedy layer-wise training of deep networks. In *NIPS*, 2016.

Bianchini, Monica and Scarselli, Franco. On the complexity of shallow and deep neural network classifiers. In *ESANN*, 2014.

Chen, Yan, Yang, Xiangnan, Zhong, Bineng, Zhang, Huizhen, and Lin, Changlong. Network in network based weakly supervised learning for visual tracking. *JVCIR*, 37, 2016.

Chollet, Franois. keras. `https://github.com/fchollet/keras`, 2015.

Collobert, Ronan, Kavukcuoglu, Koray, and Farabet, Clément. Torch7: A matlab-like environment for machine learning. In *BigLearn, NIPS Workshop*, 2011.

Dai, Jifeng, He, Kaiming, and Sun, Jian. Instance-aware semantic segmentation via multi-task network cascades. In *CVPR*, 2016.

Dozat, Timothy. Incorporating nesterov momentum into adam.

Eldan, Ronen and Shamir, Ohad. The power of depth for feedforward neural networks. In *COLT*, 2016.

Feichtenhofer, Christoph, Pinz, Axel, and Wildes, Richard P. Spatiotemporal residual networks for video action recognition. In *NIPS*, 2016.

Glorot, Xavier and Bengio, Yoshua. Understanding the difficulty of training deep feedforward neural networks. In *AISTATS*, 2010a.

Glorot, Xavier and Bengio, Yoshua. Understanding the difficulty of training deep feedforward neural networks. In *AISTATS*, 2010b.

Greff, K., Srivastava, R. K., and Schmidhuber, J. Highway and Residual Networks learn Unrolled Iterative Estimation. *ArXiv e-prints*.

He, Kaiming, Zhang, Xiangyu, Ren, Shaoqing, and Sun, Jian. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *ICCV 2015*.

He, Kaiming, Zhang, Xiangyu, Ren, Shaoqing, and Sun, Jian. Deep residual learning for image recognition. In *CVPR*, 2016a.

He, Kaiming, Zhang, Xiangyu, Ren, Shaoqing, and Sun, Jian. Identity mappings in deep residual networks. In *ECCV*, 2016b.

Huang, G., Liu, Z., Weinberger, K. Q., and van der Maaten, L. Densely Connected Convolutional Networks. *ArXiv e-prints*.

Huang, Gao, Sun, Yu, Liu, Zhuang, Sedra, Daniel, and Weinberger, Kilian Q. Deep networks with stochastic depth. In *ECCV*, 2016.

Ioffe, Sergey and Szegedy, Christian. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *ICML*, 2015.

Kingma, D. P. and Ba, J. Adam: A Method for Stochastic Optimization. *ArXiv e-prints*.

Krizhevsky, Alex. Learning multiple layers of features from tiny images. Technical report, 2009.

Lecun, Yann, Bottou, Lon, Bengio, Yoshua, and Haffner, Patrick. Gradient-based learning applied to document recognition. In *Proceedings of the IEEE*, 1998.

Montúfar, Guido F., Pascanu, Razvan, Cho, KyungHyun, and Bengio, Yoshua. On the number of linear regions of deep neural networks. In *NIPS*, 2014.

Nair, Vinod and Hinton, Geoffrey E. Rectified linear units improve restricted boltzmann machines. In *ICML*, 2010.

Romero, A., Ballas, N., Ebrahimi Kahou, S., Chassang, A., Gatta, C., and Bengio, Y. FitNets: Hints for Thin Deep Nets. *ArXiv e-prints*.

Russakovsky, Olga, Deng, Jia, Su, Hao, Krause, Jonathan, Satheesh, Sanjeev, Ma, Sean, Huang, Zhiheng, Karpathy, Andrej, Khosla, Aditya, Bernstein, Michael S., Berg, Alexander C., and Li, Fei-Fei. Imagenet large scale visual recognition challenge. *IJCV*, 115(3), 2015.

Springenberg, J. T., Dosovitskiy, A., Brox, T., and Riedmiller, M. Striving for Simplicity: The All Convolutional Net. *ArXiv e-prints*.

Srivastava, Nitish, Hinton, Geoffrey E., Krizhevsky, Alex, Sutskever, Ilya, and Salakhutdinov, Ruslan. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1), 2014.

Srivastava, Rupesh Kumar, Greff, Klaus, and Schmidhuber, Jürgen. Training very deep networks. In *NIPS*, 2015.

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

Targ, S., Almeida, D., and Lyman, K. Resnet in Resnet: Generalizing Residual Architectures. *ArXiv e-prints*.

Telgarsky, Matus. benefits of depth in neural networks. In *COLT*, 2016.

Zagoruyko, Sergey and Komodakis, Nikos. Wide residual networks. In *BMVC*, 2016.

Zhang, C., Bengio, S., Hardt, M., Recht, B., and Vinyals, O. Understanding deep learning requires rethinking generalization. *ArXiv e-prints*.