IJIRCCE

e-ISSN: 2320-9801 | p-ISSN: 2320-9798

# INTERNATIONAL JOURNAL OF INNOVATIVE RESEARCH

## IN COMPUTER & COMMUNICATION ENGINEERING

Volume 11, Issue 9, Septmber 2023

INTERNATIONAL STANDARD SERIAL NUMBER INDIA

Impact Factor: 8.379

International Journal of Innovative Research in Computer and Communication Engineering

| e-ISSN: 2320-9801, p-ISSN: 2320-9798| www.ijircce.com | |Impact Factor: 8.379 || A Monthly Peer Reviewed & Referred Journal |

|| Volume 11, Issue 9, September 2023 ||

| DOI: 10.15680/IJIRCCE.2023.1109026 |

# Deep Residual Networks for Image Recognition

**Prof. Nidhi Pateriya[1], Prof. Prerna Jain[2], Prof. Kalukuri Princy Niveditha[3], Vinayak Tiwari[4], Sheetal Vishwakarma[5]**

Department of Computer Science & Engineering, Baderia Global Institute of Engineering & Management, Jabalpur M.P, India [1, 2, 3,4,5]

**ABSTRACT:** Training deep neural networks is more challenging. We introduce a residual learning framework to simplify the training process for networks that are significantly deeper than those previously used. We specifically redesign the layers to learn residual functions based on the layer inputs, rather than learning functions without any reference. We present thorough empirical evidence indicating that residual networks are easier to optimize and can achieve higher accuracy with significantly increased depth. On the ImageNet dataset, we tested residual networks up to 152 layers deep, which are 8 times deeper than VGG networks but with lower complexity. An ensemble of these networks achieved a 3.57% error rate on the ImageNet test set, securing 1st place in the ILSVRC 2015 classification task. We also analyze networks with 100 and 1000 layers on CIFAR-10.

The depth of representations is crucial for various visual recognition tasks. By utilizing extremely deep representations, we achieved a 28% relative improvement on the COCO object detection dataset. Deep residual networks were the basis of our submissions to the ILSVRC and COCO 2015 competitions, where we secured 1st place in ImageNet detection, ImageNet localization, COCO detection, and COCO segmentation tasks.

## I. INTRODUCTION

Deep convolutional neural networks have driven significant advancements in image classification. These networks seamlessly integrate low, mid, and high-level features with classifiers in an end-to-end multilayer setup. The richness of feature levels is enhanced by increasing the number of stacked layers, or depth. Recent findings emphasize the critical role of network depth, with leading models on the challenging ImageNet dataset utilizing very deep architectures, ranging from sixteen to thirty layers. Numerous other complex visual recognition tasks have also seen substantial improvements due to the implementation of very deep models.

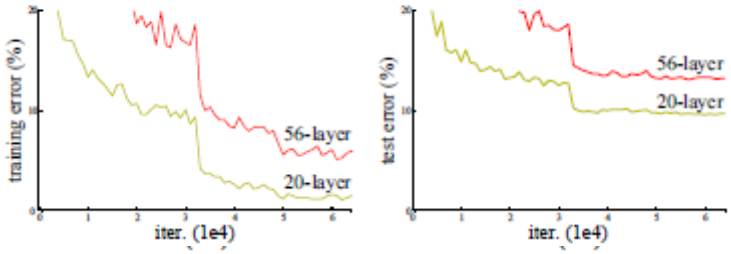



Figure 1. Training error (left) and test error (right) on CIFAR-10 with 20-layer and 56-layer "plain" networks. The deeper network has higher training error, and thus test error. Similar phenomena on ImageNet is presented in Fig. 4.

Given the importance of network depth, a key question emerges: Is improving networks simply a matter of adding more layers? The challenge in answering this lies in the well-known issue of vanishing or exploding gradients, which hinder early convergence. This problem has been largely mitigated by techniques such as normalized initialization and intermediate normalization layers. These methods have made it possible for networks with numerous layers to begin converging when using stochastic gradient descent (SGD) with back propagation.

Once deeper networks begin to converge, a new issue emerges: as network depth increases, accuracy initially saturates and then quickly deteriorates. Surprisingly, this degradation isn't due to over fitting. Instead, adding more layers to an

| e-ISSN: 2320-9801, p-ISSN: 2320-9798| www.ijircce.com | |Impact Factor: 8.379 || A Monthly Peer Reviewed & Referred Journal |

|| Volume 11, Issue 9, September 2023 ||

| DOI: 10.15680/IJIRCCE.2023.1109026 |

adequately deep model results in higher training errors. This phenomenon has been observed in previous studies and is confirmed by our experiments, as illustrated in Fig. 1.

The degradation in training accuracy indicates that not all systems are equally easy to optimize. Consider a shallower architecture and a deeper one created by adding more layers to it. There is a constructed solution for the deeper model where the additional layers act as identity mappings and the other layers are copied from the learned shallower model. This constructed solution suggests that the deeper model should not have higher training error than the shallower one. However, experiments show that current solvers are unable to find solutions as good as or better than this constructed solution.

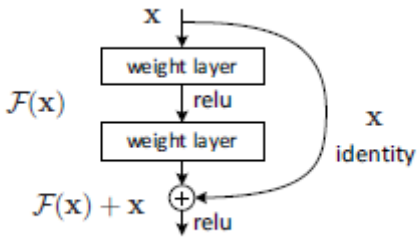



Figure 2. Residual learning: a building block.

In this paper, we tackle the degradation problem by introducing a deep residual learning framework. Instead of expecting each few stacked layers to directly learn the desired underlying mapping, we explicitly design them to learn a residual mapping. Formally, if we denote the desired underlying mapping as ( H(x) ), we allow the stacked nonlinear layers to fit a new mapping ( F(x) := H(x) - x ). Thus, the original mapping is redefined as ( F(x) + x ). We propose that optimizing this residual mapping is easier than optimizing the original unreferenced mapping. In an extreme case where an identity mapping is optimal, it is simpler to push the residual to zero than to have a stack of nonlinear layers fit the identity mapping.

The formulation ( F(x) + x ) can be implemented using feed forward neural networks with "shortcut connections" (Fig. 2). Shortcut connections, which skip one or more layers, perform identity mapping in our framework and their outputs are added to the outputs of the stacked layers (Fig. 2). These identity shortcut connections do not introduce extra parameters or computational complexity. The entire network remains trainable end-to-end using SGD with back propagation and can be easily implemented with standard libraries like Caffe without modifying the solvers.

We conducted comprehensive experiments on ImageNet to illustrate the degradation problem and assess our method. Our findings are: 1) Our extremely deep residual networks are easy to optimize, whereas equivalent "plain" networks (those that simply stack layers) show higher training error as depth increases; 2) Our deep residual networks benefit significantly from increased depth, achieving much better results than previous networks.

Similar patterns are observed with the CIFAR-10 dataset [20], indicating that the optimization challenges and the impacts of our approach are not limited to a specific dataset. We demonstrate that models with over 100 layers can be effectively trained on this dataset and investigate models with more than 1000 layers.

On the ImageNet classification dataset [35], we achieve outstanding results with very deep residual networks. Our 152-layer residual network is the deepest ever used on ImageNet, yet it maintains lower complexity compared to VGG networks [40]. Our ensemble achieves a top-5 error rate of 3.57% on the ImageNet test set and secured 1st place in the ILSVRC 2015 classification competition. These extremely deep models also demonstrate strong generalization across other recognition tasks, leading to victories in ImageNet detection, ImageNet localization, COCO detection, and COCO segmentation at the ILSVRC & COCO 2015 competitions. This compelling evidence suggests that the residual learning principle is broadly applicable and likely beneficial for both vision and non-vision problems.

## II. RELATED WORK

*Residual Representations:* In image recognition, VLAD [18] represents data by encoding residual vectors relative to a dictionary, while the Fisher Vector [30] can be viewed as a probabilistic variant [18] of VLAD. Both methods are effective shallow representations for image retrieval and classification [4, 47]. Additionally, encoding residual vectors [17] for vector quantization has been demonstrated to be more effective than encoding the original vectors.

| e-ISSN: 2320-9801, p-ISSN: 2320-9798| www.ijircce.com | |Impact Factor: 8.379 || A Monthly Peer Reviewed & Referred Journal |

|| Volume 11, Issue 9, September 2023 ||

| DOI: 10.15680/IJIRCCE.2023.1109026 |

In low-level vision and computer graphics, solving Partial Differential Equations (PDEs) often involves the Multigrid method [3], which addresses the problem by breaking it down into subproblems across multiple scales. Each subproblem handles the residual solution between a coarser and a finer scale. An alternative approach is hierarchical basis preconditioning [44, 45], which uses variables that represent residual vectors between scales. Studies [3, 44, 45] have shown that these solvers converge much more rapidly than traditional methods that do not account for the residual nature of the solutions. These techniques suggest that effective reformulation or preconditioning can significantly simplify the optimization process.

*Shortcut connections* have been explored extensively in practices and theories [2, 33, 48]. Early implementations of multi-layer perceptrons (MLPs) involved adding a linear layer that connected the network input directly to the output [33, 48]. To address issues like vanishing or exploding gradients, some approaches [43, 24] included direct connections from intermediate layers to auxiliary classifiers. Other methods [38, 37, 31, 46] used shortcut connections to center layer responses, gradients, and propagated errors. In [43], an “inception” layer integrates a shortcut branch with several deeper branches.

Around the same time as our work, "highway networks" [41, 42] introduced shortcut connections with gating functions [15]. Unlike our parameter-free identity shortcuts, these gates are data-dependent and have learnable parameters. When a gated shortcut is "closed" (approaching zero), the highway network layers function as non-residual. In contrast, our approach consistently learns residual functions, with identity shortcuts that remain active and always pass information through, incorporating additional residual functions to learn. Additionally, highway networks have not shown accuracy improvements with very deep networks (e.g., over 100 layers).

## III. DEEP RESIDUAL LEARNING

### 3.1 Residual Learning

Consider ( H(x) ) as a target function to be approximated by several stacked layers (though not necessarily the whole network), where ( x ) represents the input to the first layer. If we assume that multiple nonlinear layers can eventually approximate complex functions, this is equivalent to assuming they can asymptotically approximate residual functions, i.e., ( H(x) - x ) (given that the input and output dimensions are the same). Therefore, instead of expecting the stacked layers to approximate ( H(x) ), we directly let them approximate a residual function ( F(x) := H(x) - x ). The original function can then be expressed as ( F(x) + x ).

While both approaches should theoretically be able to approximate the desired functions asymptotically, learning might be easier with the residual approach. This reformulation addresses the degradation problem (Fig. 1, left), which suggests that deeper networks should not perform worse than shallower ones if the added layers are identity mappings. The degradation issue implies that approximating identity mappings with multiple nonlinear layers might be challenging. With the residual learning approach, if identity mappings are optimal, the network can adjust the weights of the nonlinear layers towards zero to approximate these identity mappings.

In practice, identity mappings may not be the optimal solution, but this reformulation helps to simplify the problem. If the optimal function is closer to an identity mapping than to a zero mapping, it should be easier for the network to find deviations from the identity mapping rather than learning a completely new function. Our experiments (Fig. 7) show that the learned residual functions typically have small responses, indicating that identity mappings offer a useful preconditioning.

### 3.2 Identify mapping by shortcuts

We apply residual learning to groups of stacked layers. As illustrated in Fig. 2, a building block is defined as:

$$y = F(x, \{W_i\}) + x$$

Here, ( x ) and ( y ) represent the input and output vectors of the layers, respectively. The function ( F(x, {W_i\}) ) denotes the residual mapping to be learned. For the example shown in Fig. 2, with two layers, ( F = W_2 sigma(W_1 x) ), where ( sigma ) denotes the ReLU activation function [29], and biases are omitted for simplicity. The residual function ( F + x ) is computed using a shortcut connection and element-wise addition. We apply the second nonlinearity after the addition, i.e., ( sigma(y) ), as shown in Fig. 2.

The shortcut connections in this formulation add no extra parameters or computational complexity, making them practical and important for fair comparisons between plain and residual networks. This allows us to compare networks

| e-ISSN: 2320-9801, p-ISSN: 2320-9798| www.ijircce.com | |Impact Factor: 8.379 || A Monthly Peer Reviewed & Referred Journal |

|| Volume 11, Issue 9, September 2023 ||

| DOI: 10.15680/IJIRCCE.2023.1109026 |

with the same number of parameters, depth, width, and computational cost, aside from the minor computational overhead of element-wise addition.

In Eqn. (1), the dimensions of ( x ) and ( F ) must be the same. If they differ, such as when changing input/output channels, a linear projection ( W_s ) can be applied via the shortcut connection to match the dimensions:

[ y = F(x, {W_i}) + W_s x ]

While ( W_s ) can be a square matrix, our experiments show that using an identity mapping is generally sufficient and more economical for addressing the degradation problem. Therefore, ( W_s ) is only used when necessary to match dimensions.

The residual function FFF can be flexible in its form. Experiments described in this paper use functions FFF with two or three layers (Fig. 5), although more layers are possible. However, using only a single layer for FFF yields a form similar to a linear layer:

y=W1x+xy = W_1 x + xy=W1x+x which has not shown any significant advantages. While our examples use fully-connected layers for simplicity, the same principles apply to convolutional layers, where F(x,{Wi})F(x, {W_i})F(x,{Wi}) can include multiple convolutional layers, and the element-wise addition occurs channel by channel across two feature .

### 3.3 NETWORK ARCHITECTURE

We have tested various plain and residual networks and observed consistent results. To illustrate, we describe two models used for ImageNet.

Plain Network: Our plain baseline models (Fig. 3, middle) are primarily based on the VGG network philosophy [40] (Fig. 3, left). These networks use mainly 3×3 convolutional filters and adhere to two design principles: (i) layers with the same output feature map size have the same number of filters, and (ii) when the feature map size is reduced by half, the number of filters is doubled to maintain consistent time complexity per layer. Downsampling is achieved through convolutional layers with a stride of 2. The network concludes with a global average pooling layer followed by a 1000-way fully-connected layer with softmax. The total number of weighted layers in this 34-layer network (Fig. 3, middle) is 34.

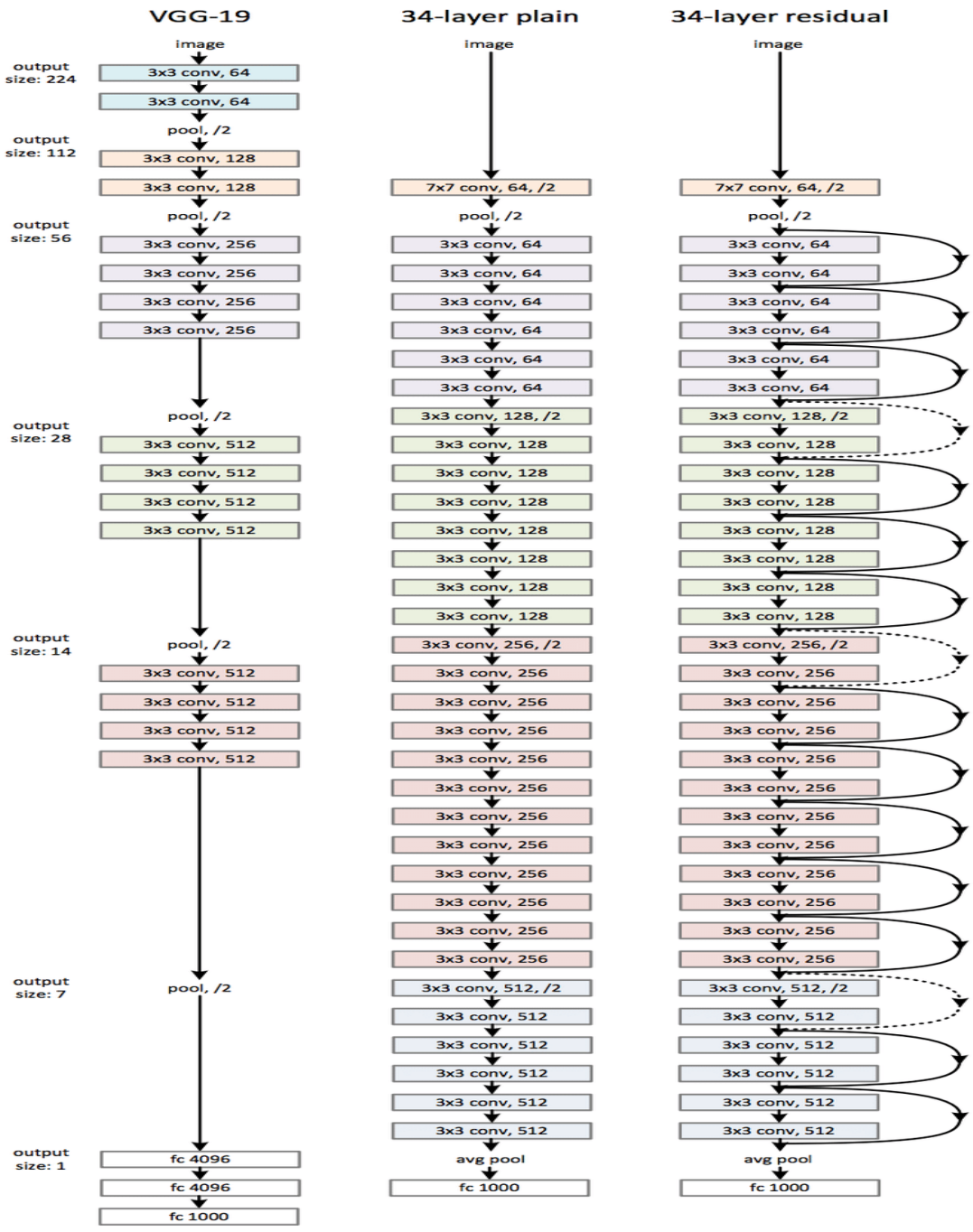
Notably, our model uses fewer filters and has lower complexity compared to VGG networks [40] (Fig. 3, left). Our 34-layer baseline network has 3.6 billion FLOPs (floating-point operations), which is only 18% of the FLOPs in VGG-19, which totals 19.6 billion FLOPs.

*Residual Network:* Building on the plain network described earlier, we introduce shortcut connections (Fig. 3, right) to create a residual version of the network. Identity shortcuts (Eqn. (1)) are used directly when the input and output dimensions are the same (solid line shortcuts in Fig. 3). When the dimensions differ (dotted line shortcuts in Fig. 3), we have two options: (A) Use identity mapping for the shortcut, adding zero-padding to accommodate the increased dimensions. This method does not introduce any additional parameters; (B) Use a projection shortcut (Eqn. (2)) to match dimensions via 1×1 convolutions. For both options, when shortcuts need to connect feature maps of different sizes, they employ a stride of 2.

| e-ISSN: 2320-9801, p-ISSN: 2320-9798| www.ijircce.com | |Impact Factor: 8.379 || A Monthly Peer Reviewed & Referred Journal |

|| Volume 11, Issue 9, September 2023 ||

| DOI: 10.15680/IJIRCCE.2023.1109026 |




3.4 IMPLEMENTATION

Our implementation for ImageNet follows the methods outlined in [21, 40]. The images are resized so that their shorter side is randomly sampled within the range [256, 480] for scale augmentation [40]. We then randomly crop a 224×224 region from the image or its horizontal flip, and subtract the per-pixel mean [21]. Standard color augmentation techniques from [21] are also applied.

We use batch normalization (BN) [16] immediately after each convolution and before activation, in line with [16]. Weight initialization follows the approach in [12], and all plain and residual networks are trained from scratch. We employ SGD with a mini-batch size of 256, starting with a learning rate of 0.1, which is reduced by a factor of 10 when the error plateaus. Training is carried out for up to 600,000 iterations, with a weight decay of 0.0001 and momentum of 0.9. Dropout [13] is not used, consistent with [16].

| e-ISSN: 2320-9801, p-ISSN: 2320-9798| www.ijircce.com | |Impact Factor: 8.379 || A Monthly Peer Reviewed & Referred Journal |

|| Volume 11, Issue 9, September 2023 ||

| DOI: 10.15680/IJIRCCE.2023.1109026 |

For testing and comparison, we use standard 10-crop testing [21]. To achieve the best results, we employ the fully-convolutional form as described in [40, 12] and average the scores across multiple scales, resizing images so that the shorter side is in {224, 256, 384, 480, 640}.

## REFERENCES

[1] R. Girshick. Fast R-CNN. In ICCV, 2015.
[2] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In CVPR, 2014.
[3] I. J. Goodfellow, D. Warde-Farley, M. Mirza, A. Courville, and Y. Bengio. Maxout networks. arXiv:1302.4389, 2013.
[4] K. He and J. Sun. Convolutional neural networks at constrained time cost. In CVPR, 2015.
[5] K. He, X. Zhang, S. Ren, and J. Sun. Spatial pyramid pooling in deep convolutional networks for visual recognition. In ECCV, 2014.
[6] K. He, X. Zhang, S. Ren, and J. Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In ICCV, 2015.
[7] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. R. Salakhutdinov. Improving neural networks by preventing coadaptation of feature detectors. arXiv:1207.0580, 2012.
[8] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In ICML, 2015.
[9] H. Jegou, M. Douze, and C. Schmid. Product quantization for nearest neighbor search. TPAMI, 33, 2011.
[10]H. Jegou, F. Perronnin, M. Douze, J. Sanchez, P. Perez, and C. Schmid. Aggregating local image descriptors into compact codes. TPAMI, 2012.
[11]Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell. Caffe: Convolutional architecture for fast feature embedding. arXiv:1408.5093, 2014.
[12]A. Krizhevsky, I. Sutskever, and G. Hinton. Imagenet classification with deep convolutional neural networks. In NIPS, 2012.
[13]C.-Y. Lee, S. Xie, P. Gallagher, Z. Zhang, and Z. Tu. Deeply supervisednets. arXiv:1409.5185, 2014.
[14]M. Lin, Q. Chen, and S. Yan. Network in network. arXiv:1312.4400, 2013.
[15]T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Doll´ar, and C. L. Zitnick. Microsoft COCO: Common objects in context. In ECCV. 2014.
[16]J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In CVPR, 2015.
[17]G. Mont´ufar, R. Pascanu, K. Cho, and Y. Bengio. On the number of linear regions of deep neural networks. In NIPS, 2014.
[18]T. Raiko, H. Valpola, and Y. LeCun. Deep learning made easier by linear transformations in perceptrons. In AISTATS, 2012.
[19]S. Ren, K. He, R. Girshick, and J. Sun. Faster R-CNN: Towards real-time object detection with region proposal networks. In NIPS, 2015.
[20]A. Romero, N. Ballas, S. E. Kahou, A. Chassang, C. Gatta, and Y. Bengio. Fitnets: Hints for thin deep nets. In ICLR, 2015.

INNO SPACE
SJIF Scientific Journal Impact Factor

Impact Factor: 8.379

doi cross ref

ISSN INTERNATIONAL STANDARD SERIAL NUMBER INDIA

निस्केयर NISCAIR

# INTERNATIONAL JOURNAL OF INNOVATIVE RESEARCH

## IN COMPUTER & COMMUNICATION ENGINEERING

9940 572 462 6381 907 438 ijircce@gmail.com

www.ijircce.com

Scan to save the contact details