Departmemt of Mathematics and Statistics

DEEP LEARNING PROJECT

Review: Deep Residual Learning for Image Recognition

Michael Carnival

Supervisor: **Dr. Shusen Pu**

October 5, 2024

Contents

CHAPTER 1

Introduction

In deep convolutional neural networks, a learning algorithm primarily used for image and video classification, there have been many breakthroughs for image classifications and recognition tasks. For instance, Alex Krizhevsky; presented a regularization method called ”dropout,” where neurons randomly deactivate a set of nodes in each iteration, which has proven to be very effective [1], Pierre Sermanet; incorporating a multiscale and sliding window within the ConvNet and learning to predict object boundaries using bounding boxes to increase detection confidence [2], and Matthew D Zeiler; using visualization technique that gives insight into the function of intermediate feature layers and also retraining SoftMax classifier [3]. There is evidence of how the network depth is crucial in enhancing the performance in many nontrivial visual recognition tasks [4, 5]; object detection, image segmentation, facial recognition, action, recognition, etc. However, with deeper networks arises the question from the author, ”is stacking more layers always a result of better learning?” Answering this question was notoriously difficult because of the vanishing/exploding gradients issue [6, 7]. A vanishing gradient occurs when the loss function’s gradient approaches zero, while the exploding gradient problem occurs when the function’s gradient becomes too large, resulting in large updates to weights during training. The issue has been largely addressed by normalized initialization [8] and Batch normalization [9], which allowed for tens of layers to start converging using stochastic gradient descent with backpropagation [10]. The normalized initialization helps maintain the variance of back-propagated gradients across layers and training iterations by preventing them from vanishing or exploding

as they move through the layers. Batch normalization (BN) is a technique that stabilizes the learning process of a neural network. The BN encourages higher learning rates and faster convergence in training. It also acts as a regularization parameter without the need for dropout regularization. The backpropagation algorithm calculates the gradient of the loss function with respect to each weight by the chain rule, which helps the network learn from its previous mistakes.

CHAPTER 2

Degradation Problem

When deeper networks converge, a degradation problem arises, causing accuracy to saturate. This saturation of accuracy is not caused by overfitting, and adding more layers to a deep model leads to higher training error [11]. This is indicated by the A. Another problem regarding deeper network is constrained time cost, fixing this can help reduce the cost computing, resource utilization and scalability for practical application [12]. The degradation problem and time complexity are improved by a deep residual learning framework.

2.1 Proposal

In this framework, the stacked layers are fitted to a residual mapping denoted by the desired mapping as $\mathcal{H}x$, the stacked nonlinear layer is fit to another mapping of $\mathcal{F}(x) := \mathcal{H}(x) - x$. The original mapping is recast into $\mathcal{F}(x) + x$. The author's hypothesis is that it would be easier to optimize the residual mapping to zero than to optimize the original mapping. The formulation of $\mathcal{F}(x) + x$ can be realized by feedforward neural networks with shortcut connection (aka skip connection). The shortcut connection works by bypassing one or more layers in between. The goal of a shortcut connection performs identity mapping without sacrificing higher computational complexity or by adding extra parameters to the network.

2.2 Significance of the Problem

Addressing the degradation problem while maintaining low complexity is crucial for applications related to recognition tasks. Before this paper was published, it was obvious to few whether increasing the depth of a deep network model would hinder its ability to perform recognition tasks. However, after this paper was in the public realm, the answer was shocking to most. This paper showed that it was possible to go deeper and achieve robust performance results. The news allowed deep learning models to become more complex and paved the way for new architectures and the development of more efficient models. Finally, better deep learning models entail advances in a wide range of real-world applications, from image recognition and natural language processing to the medical field and autonomous driving. Overall, addressing the degradation issue among deep learning models would allow the scalability of deeper models with practical applications in various domains.

CHAPTER 3

Related Work

The idea of the residual learning block came from image processing, where a vector of locally aggregated descriptors (VLAD) is a representation encoded by the residual vectors with respect to a dictionary [13]. The Fisher vector [14] is a probabilistic version of VLAD. Both vectors are powerful shallow representations for image retrieval and classification. For vector quantization, an idea that decomposes the space into a Cartesian product of low-dimensional subspaces and quantizes each subspace separately [15], has been shown to be more effective in encoding with residual vectors than in encoding original vectors.

The widely used multigrid method [16] reformulates the system as a subproblem at multiple scales, where each subproblem is responsible for the residual solution between a coarser and a finer scale. An alternative to the latter is hierarchical basis preconditioning, which depends on variables that represent residual vectors between two scales. These multigrid method solvers converge much faster than standard solvers that are unaware of the residual nature of the solution. These methods suggest that good reformulation or preconditioning can simplify optimization, leading to better performance.

The ideas that led to shortcut connections have been studied for a long time [17]. Early practice of implementing multi-layer perceptron (MLP) was an example of a shortcut connection, where a linear layer was connected to the input network and to the output of the network [18]. Contrary to the identity shortcut implemented in this paper, highway networks present shortcut connections with gating functions [19] that

depend on data and have parameters. In highway networks, a gate shortcut that is closed (approaching zero) represents a non-residual function, whereas in this paper, the formulation always learns the residual function and never closes.

CHAPTER 4

Methodology

4.1 Residual Learning

In residual learning, the $\mathcal{H}(x)$ as an underlying mapping to be fit by a few stacked layers with x denoting the inputs to the first layers. It is hypothesized that with multiple nonlinear layers, complex function can be approximated. Then, it can be hypothesized that the residual function, i.e., $\mathcal{H}(x) - x$ can be approximated (assuming the input and output are of the same size) as well. Essentially, rather than stacking layers to approximate the $\mathcal{H}(x)$, it would be equivalent to approximate the residual function $\mathcal{F}(x)$ such that the original function is modified to $\mathcal{F}(x) + x$. The reformulation of the functions was motivated by the left side of the Fig 1. and the related work section. The added layers would be constructed as identity mapping for skipping connections within the network (i.e., input to a layer is directly passed to an output of a subsequent layer preserving the original information). With the introduction of the residual learning reformulation as the identity mapping, it would make conventional solver easier to find the perturbations with reference to an identity mapping than it is to learn the function as anew. In real cases, the identity mappings are unlikely to be optimal. In Fig 7, the experiments show the response in layers is lower in ResNets than the plainNets.

4.2 Identity Mapping by Shortcuts

The residual learning is adopted for every few stacked layers shown in Fig 2. In this paper a building block is defined as

$$y = \mathcal{F}(x, W_i) + x. \tag{4.1}$$

Where x and y are the input and output vectors of the layers considered and $\mathcal{F}(x, W_i)$ represents the residual mapping learned. As an example in fig 2, with two layers, $\mathcal{F} = W_2\sigma(W_1x), \sigma$ denotes the ReLu and the biases are omitted. The $\mathcal{F} + x$ is performed by a shortcut connection and element wise addition. The benefit of the shortcut connections is that it neither adds extra parameters nor computation complexity. The x and $\mathcal{F}$ must have the same dimension to be equal to Eq.4.1, if it is not, a linear projection W_s is applied to x in order to match the dimensions:

$$y = \mathcal{F}(x, W_i) + W_sx \tag{4.2}$$

It is noted that the shortcut connection operations are applicable to convolutions layers where $\mathcal{F}(x, W_i)$ can represent multiple convolution layers.

4.3 Network Architectures

There were two networks tested: the plain network and the residual network, using the ImageNet dataset. The plain networks are mainly inspired by VGG nets, where the goal was to investigate the performance and accuracy of these convolutional networks by increasing depth [21]. Most of the convolutional layers had 3 x 3 filters and followed two rules: the layers have the same number of filters as the output feature map size, and if the feature map is halved, the number of filters is doubled to preserve the time complexity per layer. Downsampling by convolutional layers with a stride of 2 was applied to allow the network to reduce the size of the feature maps. The network ends with a global average pooling layer and a 1000-way fully connected layer using the Softmax activation function. The network consists of 34 weighted layers with 3.6 billion floating point operations per second (FLOPs) and is represented in the middle of Fig 3. It has fewer filters than the VGG nets in [21]. The residual network was built upon the plain network, where shortcut connections are inserted between two subsequent layers:

linear and ReLU nonlinearity layers, as shown at the top of Fig 3. When the dimensions of the convolutions increase, two options are considered: 1) the shortcut performing identity mapping with no extra zero entries, or 2) the projection shortcut in Eq. 4.2 to match the dimensions. Both options use a stride of 2 to go across the feature maps.

4.4 Implementation of residual network

Using the ImageNet dataset, it follows the practices of VGGNet and AlexNet. The input image size is adjusted to randomly sample from a range of [256, 480]. To obtain a fixed-size ConvNet input image, a 224 x 224 crop is randomly sampled and undergoes a random horizontal flip, with the mean activity over the training set subtracted from each pixel, and a random RGB color shift [1]. The paper adopts batch normalization right after each convolution and before activation, allowing higher learning rates and eliminating the need for Dropout, a training strategy for neural networks that randomly deactivates a set of nodes in each iteration [9]. The hyper-parameters include a weight decay of 0.0001, momentum of 0.9, no dropout, a mini-batch size of 256 using SGD, a starting learning rate of 0.1 that is divided by 10 when the error plateaus, and a total number of epochs of 80. For testing, 10-crop testing, fully convolutional form, and average scores at multiple scales were adopted for the best results.

CHAPTER 5

Experiment Results

5.1 Plain vs. ResNet

The results presented in the paper show that the 34-layer plain net has a higher validation error than the 18-layer plain net. The reason for this is unlikely to be caused by vanishing gradients; instead, it could be the solver since BN ensures the forward-propagated signal has non-zero variances. It is conjectured that plain nets have exponentially low convergence rates, which could lead to high training error. The residual network baseline architectures are the same as the plain nets, with the addition of a shortcut connection to each of the 3 x 3 filters shown in Fig. 3 (top). The result comparison between the 18-layer and the 34-layer nets shows that the 34-layer ResNet has lower training and validation errors with increasing iterations, as displayed in Fig. 4, and a 2.8% lower error in the 34 layers compared to the 18 layers, thus indicating that 34 layers are more generalizable to the validation data than the 18 layers. It also beats the 34-layer plain net in the top-1 error by a 3.5% reduction. Lastly, the 18-layer plain/residual nets are similar in accuracy, but residual nets do not have extra parameters and converge faster than the plain nets. These experiments help answer the degradation problem, where increasing the depth can increase accuracy.

5.2 Identity vs Projection shortcut

Shown in **Table 3**, three options are compared. For the first option, zero-padding shortcuts are used to increase the dimensions and are free of parameters. For the second option, projection shortcuts are used to increase the dimensions, with other shortcuts as identity. For the third option, all shortcuts are projections. The results in Table 3 show that all options perform better than the plain Net, with a small difference in performance between options, indicating that projection shortcuts are not crucial for addressing the degradation problem. However, identity shortcuts are important for bottleneck architectures. In this architecture, shown in Fig 5,

The residual function $\mathcal{F}$ uses a stack of 3 layers instead of 2, where the three layers are 1x1, 3x3, and 1x1 convolutions. Here, the 1x1 layers reduce and restore the dimensions, and the 3x3 layer has a smaller input/output dimension. Both designs in Figure 5 have the same time complexity. This means that the identity shortcut enables more efficient models, such as the bottleneck architecture. This is exemplified by the 50-layer, 101-layer, and 152-layer ResNet, where the complexity is still lower than the VGG-16/18 nets, with more accurate results than the 34-layer by a considerable margin (Tables 3 and 4).

5.3 Comparing State-of-the-Art Methods

The baseline 34-layer ResNets are very competitive among other methods. The 152-layer single model achieved a top-5 validation error of 4.49%, beating all previous ensemble results. Six models with different depths form the ensemble model, leading to a 3.57% top-5 error on the test set to win LLSVRC 2015.

5.4 CIFAR-10 Analysis

Additional studies were done on the CIFAR-10 dataset [22] to examine the behavior of extremely deep networks. The dataset contains 50k training images and 10k testing images in 10 classes. Following the VGG and AlexNet practice, the network inputs are 32 x 32 with the mean activity over the training set subtracted from each pixel. The network consists of 6n layers with 3 x 3 convolutions on feature map sizes of 32, 16,

and 8, respectively. There are 2n layers for each feature map size with filter lengths of 16, 32, and 64, respectively. Subsampling is done using convolution with a stride of 2, ending with global average pooling and a 10-way fully connected layer with softmax activation.

For this architecture, identity shortcuts were used so that the residual models have the same depth, width, and number of parameters as the plain net. The hyperparameters include a weight decay of 0.0001, momentum of 0.9, adoption of BN, a mini-batch size of 128, and a learning rate of 0.1 divided by 10 at 32k and 48k iterations, ending training at 64k iterations. The train/validation split was 45k/5k [23]. A 32 x 32 crop was randomly sampled from its horizontal flip, with the mean activity over the training set subtracted from each pixel and a random RGB color shift applied. For testing, a single view of the original 32 x 32 image was used, comparing $n = 3, 5, 7, 9$, which led to 20, 32, 44, and 56-layer networks. Figure 6 (left) shows that the plain net demonstrates the same high error with deeper layers, supporting the optimization difficulty with the plain net's architecture. In contrast, the ResNet (middle) demonstrates lower error with deeper layers. The right of Figure 6 shows that ResNet-110 has a lower error than ResNet-1202, indicating that extremely deep layers could lead to overfitting. When comparing the network to state-of-the-art methods such as FitNet and Highway, as shown in Table 6, ResNet method outperformed all of them with ResNet-110 6.43 error.

The ResNet analysis of layers reveals the strength of the residual functions, showing that ResNets generally have smaller responses than their plain counterparts, indicating that the residual functions are more able to push to zero than non-residual functions. While exploring above 1000 layers, (n = 200) leading to a 1202-layer network had no optimization difficulty and resulted in a 7.93% error. The open problem with the 1202-layer network is that it performs worse than the 110-layer network on the testing error but has the same training error.

5.5 Object Detection on PASCAL and MS COCO

The PASCAL VOC 2007 and 2012 action classification tasks and the Common Objects in Context (COCO) dataset using the ResNet framework with Faster R-CNN for detection resulted in 1st place in the ILSVRC and COCO 2015 competition. The winning

implementation had a remarkable 6.0% increase in COCO's standard metric mAP[0.5, 0.95]

5.6 Significant Findings

The findings in this paper demonstrate that residual network frames (skip-connection addition) are easier to optimize while maintaining higher accuracy with increasing depth, whereas plainnets (simply stacked layers) produce higher training errors with increasing depth. It is also noted that the effects of these residual nets are not limited to a particular dataset but have a broader scope in recognition tasks. This led the author and his team to win 1st place in ImageNet detection, ImageNet localization, COCO detection, and COCO segmentation in the ILSVRC & COCO 2015 competition. It is evident that the residual learning principle is generalizable to both vision and non-vision problems.

CHAPTER 6

Paper's Impact

In the field of deep learning, the state-of-the-art method VGG 16/19 at the time had 19.6 billion FLOPS. The ResNet baseline 34 has 3.6 billion FLOPS, which is only 18% of VGG 19. Fewer FLOPS mean less time for training deep networks. ResNet also provides excellent generalization ability to predict the test with high accuracy. The introduction of skip connections allows layers to converge on a solution better without adding more complexity. The experimental results proved that it is possible to have a deeper network without increasing error. The residual nets were an important breakthrough because, at the time of competition, it was the deepest network, and the result of the residual network reached the top 5 error for the ImageNet test set while maintaining lower complexity than the VGG nets [21]. This is important in the broader field by aiding medical professionals in making decisions on patient health, such as diagnosing lung tumors, skin diseases, breast diseases, and brain diseases ([27]). ResNet lays the benchmark for future research in deep learning, and recently in 2019, EfficientNet, a better-performing architecture, was discovered using ResNet as its inspiration and baseline [24]. The paper "Attention is Not All You Need: Pure Attention Loses Rank Doubly Exponentially with Depth" discusses how skip connections have been used to facilitate optimization in deep networks, especially in preventing transformer output from degenerating to rank one exponentially quickly with respect to network depth [26]. These residual connections (inspired by ResNet) in transformers (e.g., NLP models like BERT and GPT-3) improve the flow of gradients and enhance performance [28]. Lastly, recognizing and localizing endangered species is of great importance to the

environment. In a competition hosted on Kaggle by NOAA tackling the issue of fewer than 500 North Atlantic right whales, an individual, Felix Lau, was able to obtain second place with a 67-layer ResNet bottleneck architecture [29]. The above examples demonstrate the implications of what the ResNet architecture can accomplish in various fields.

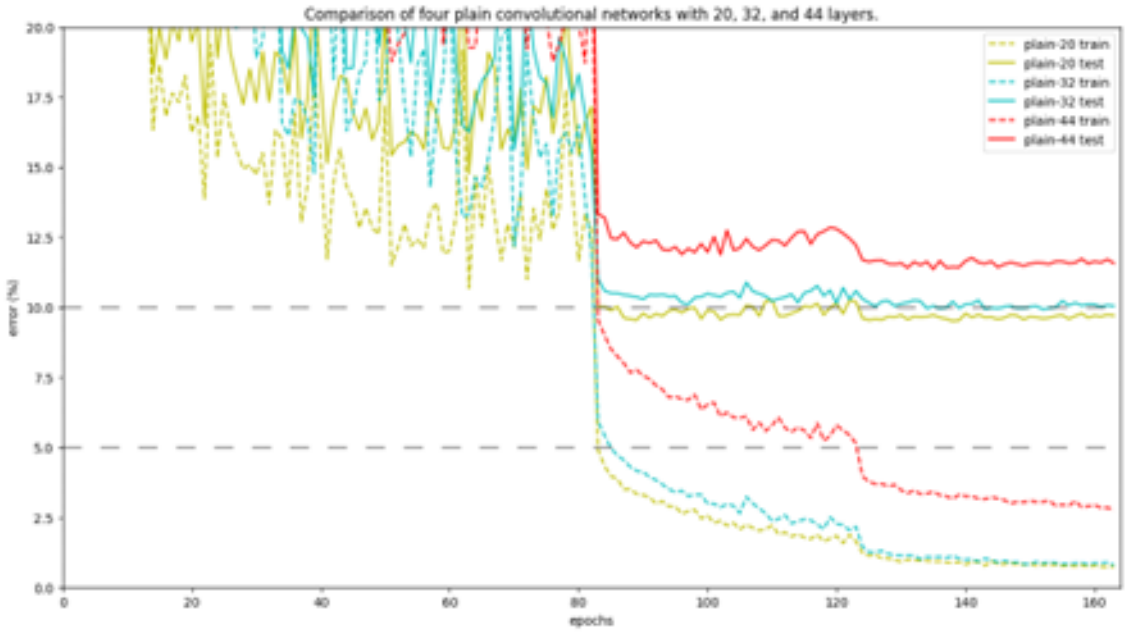
CHAPTER 7

Code Implementation

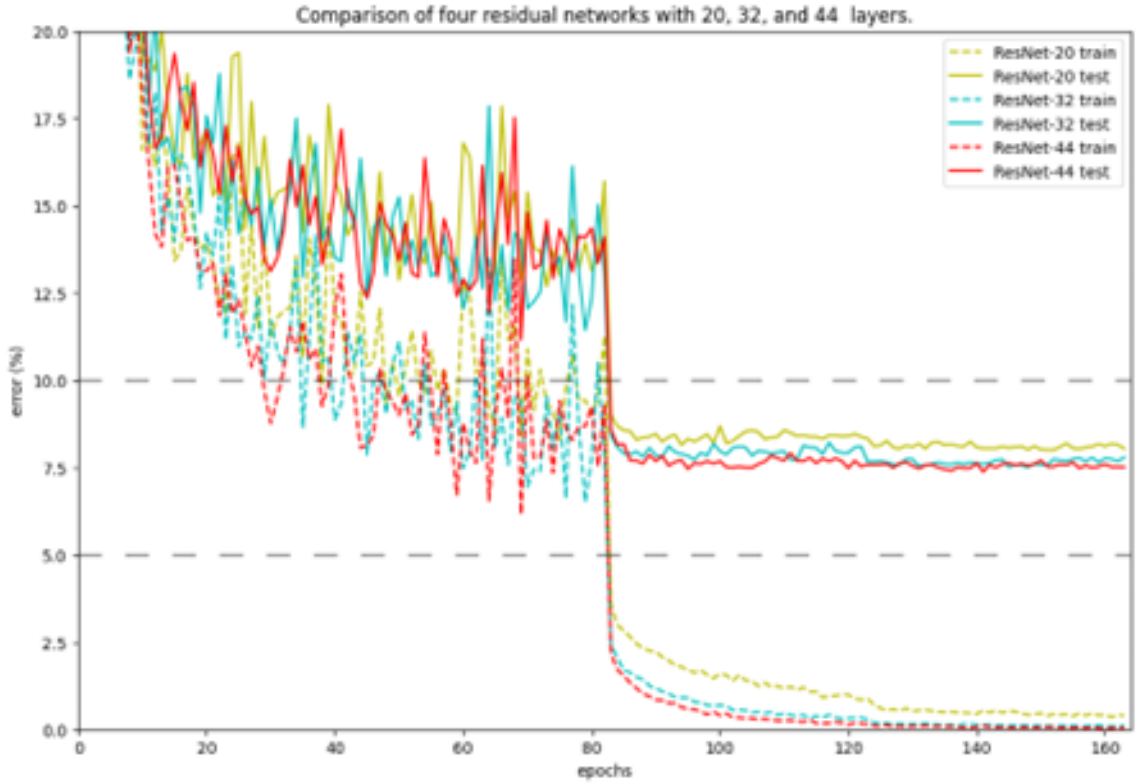
This code only covers the CIFAR-10 dataset. Running the ImageNet dataset without proper hardware would have taken me a few days of constant running. The PlainNet20 model, using local hardware (i.e., CPU), took more than 10 hours to complete due to the incapability of my GPU, AMD Radeon (TX) RX 460 Graphics, to compute on CUDA. With the code provided by the author Kaiming He (resnet.py) and A-Martyn's GitHub repository (main.ipynb, how_residual_shortcuts_speed_up_learning.ipynb, train.py, utils.py, and data_loader.py,), I was able to generate results close to those discussed in the paper[25]. The code from A-Martyn came with a few errors. The first error in the code notebook was in the file main.ipynb under the section for data augmentation training and testing set, where data_iter.next() caused it to stop. I had to change it to data_iter.__next__(). The next error occurred in the train.py file at the evaluate function, where the y_true and y_pred variables were assigned dtype = np.int. Due to outdated syntax, I changed dtype = np.int to dtype = int. After these errors were corrected, the code was able to run and generate results. Below is evidence of training working for the PlainNet20.

```
164     350     0.036
train_err: 0.00738 test_err: 0.0969
Finished Training
```

Luckily, the repository provided the finished models and results. With the results, the plot of train/test of the plainnet 20, 32, and 44 are represented. The test set are the solid lines and the train sets are the dashed lines.



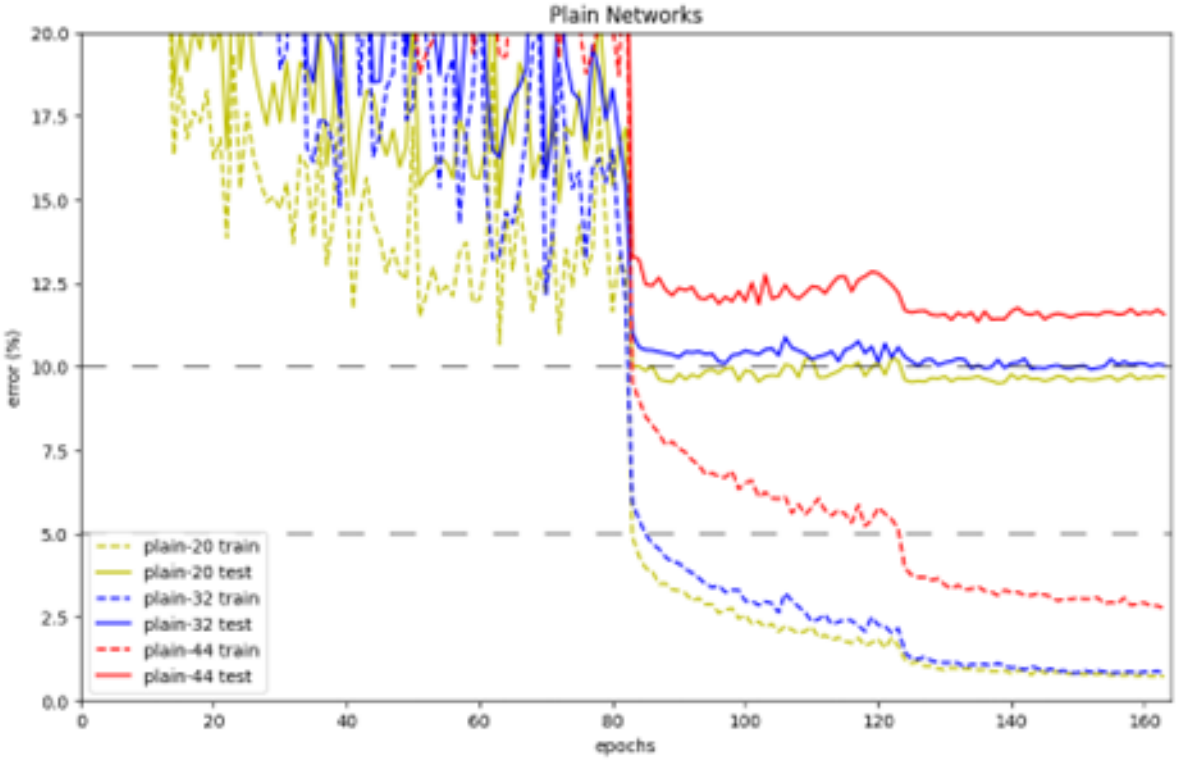
Clearly, from the graph, the red series with 44 layers has a higher error rate than the yellow series with 20 layers in both training and testing, supporting the previously held notion that deeper layers saturate accuracy. The proposed solution, ResNet, shows a different conclusion regarding the depth of the layers, as shown below.



The plot the resembles the results of what was written in the paper, where the red series are deeper than the rest of the series with lower error %. The side-by-side comparison shown below using the paper's cifar10 result and reproducing the result clearly exhibit

similar characteristics.

Code implementation side by side results



Code implementation ResNet results

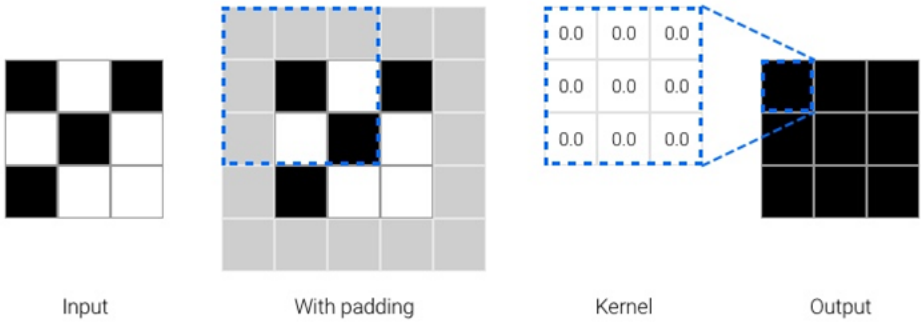
model	test_err
ResNet20	0.0800
ResNet32	0.0751
ResNet44	0.0738

CHAPTER 8

Further Investigations

To further explain why shortcut connections work, the notebook provided by A-Martyn demonstrates the workings. As an example, provided below with a kernel size of 3, stride 1, and padding 1.

With skip connection

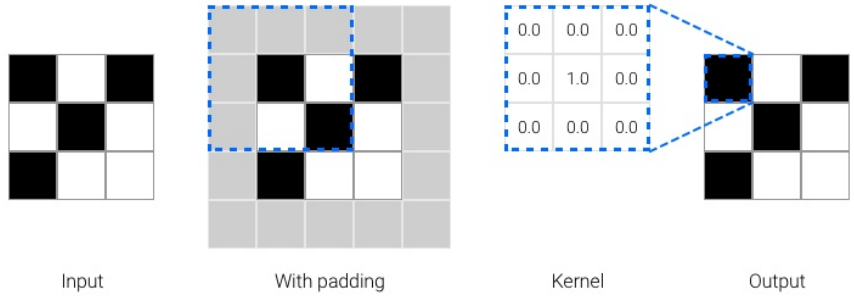


The layer's objective is to output a matrix of zeros irrespective of input because the input is added to the layer's output via Equation 2. The blue line multiplies every input with zero at every possible position, thus the output must be zero.

```
Input:
 tensor([[[[0.4320, 0.3021, 0.0559],
           [0.4008, 0.9953, 0.7212],
           [0.2603, 0.2033, 0.5992]]]])
Output:
 tensor([[[[0., 0., 0.],
           [0., 0., 0.],
           [0., 0., 0.]]]], grad_fn=<ConvolutionBackward0>)
```

Without skip connection and a weight of 1 at the center position, shown below

No skip connection

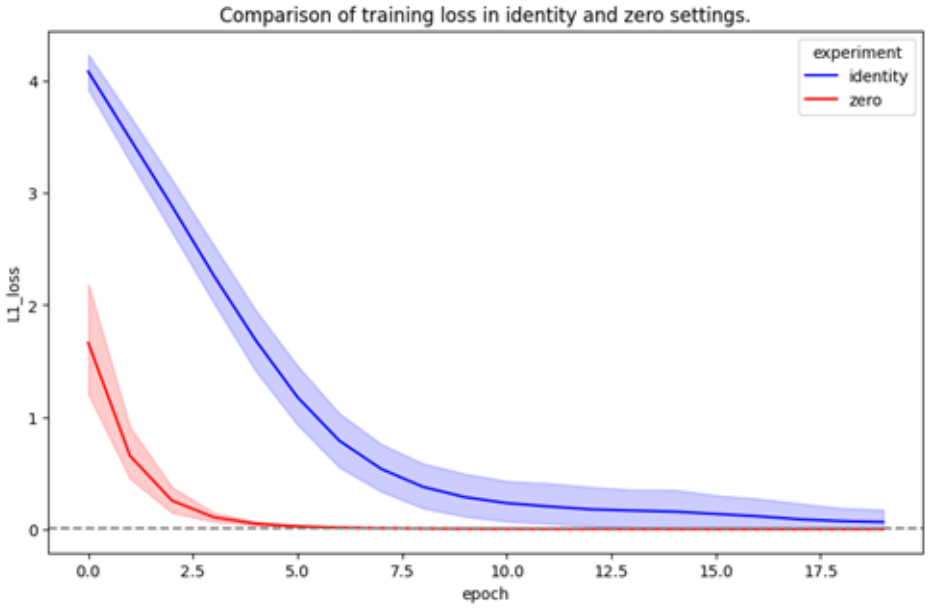


Without skip connection and a weight of 1 at the center position, shown below

No skip connection

```
Input:
 tensor([[[[0.4752, 0.4173, 0.2474],
          [0.6374, 0.3077, 0.1215],
          [0.8707, 0.8964, 0.6278]]]])
Output:
 tensor([[[[0.4752, 0.4173, 0.2474],
          [0.6374, 0.3077, 0.1215],
          [0.8707, 0.8964, 0.6278]]]], grad_fn=<ConvolutionBackward0>)
```

To test He's hypothesis, a minimal model using skip, and no-skip connection, a single convolution layer of 3x3 kernel, stride 1, padding of 1, ReLU activation, and 9 initialized learnable parameters was experimented. The training loop uses an L1 loss function to measure the absolute difference between the target and output of the model. The rest of the process utilizes a backpropagation training loop using the Adam optimizer. The plot below is produced



We can see that setting the output layers to zero reduces the loss to below 0.01 (red) in 3.5 epochs, whereas the identity setting (blue) takes 10 epochs. The conclusion for this experiment supports He's hypothesis, in which the model learns much faster with the zero setting than it does with the identity function.

CHAPTER 9

Conclusions

In this paper, we explored the intricacies of how deep residual networks address the degradation problem. Our findings reveal that skip-connections enable the development of deeper and more complex models without increasing complexity. The significance of this approach has been demonstrated in various applications, from identifying whales to creating more efficient transformers in BERT, GPT, and NLP. Our code implementation replicated the results presented in the paper, comparing the performance of plain networks and ResNet architectures. The investigation into how layers learn with and without skip-connections supported the author's hypothesis that it is easier to learn to map the zero identity function than the original identity function. Overall, ResNet architecture has had a tremendous impact on the field of deep learning and continues to be utilized for various recognition tasks.

APPENDIX A

Appendix Figure

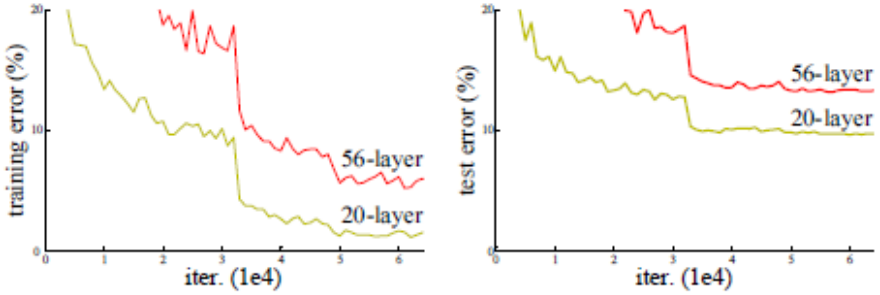


Figure 1. Training error (left) and test error (right) on CIFAR-10 with 20-layer and 56-layer "plain" networks. The deeper network has higher training error, and thus test error. Similar phenomena on ImageNet is presented in Fig. 4.

Figure 1

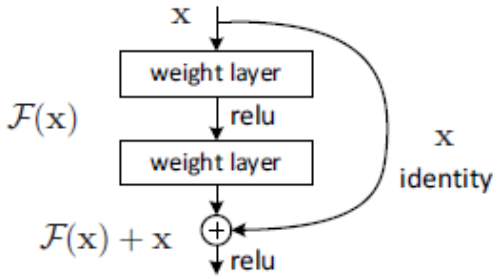


Figure 2. Residual learning: a building block.

Figure 2

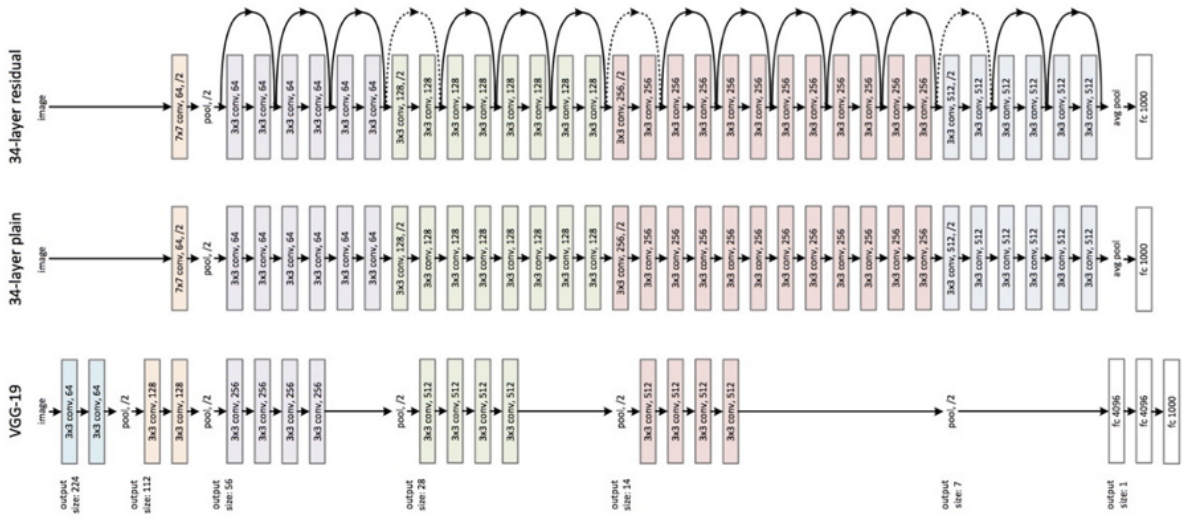


Figure 3

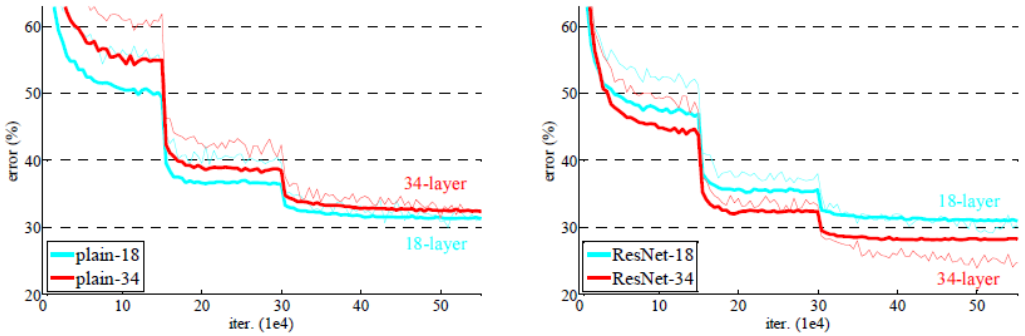


Figure 4. Training on **ImageNet**. Thin curves denote training error, and bold curves denote validation error of the center crops. Left: plain networks of 18 and 34 layers. Right: ResNets of 18 and 34 layers. In this plot, the residual networks have no extra parameter compared to their plain counterparts.

Figure 4

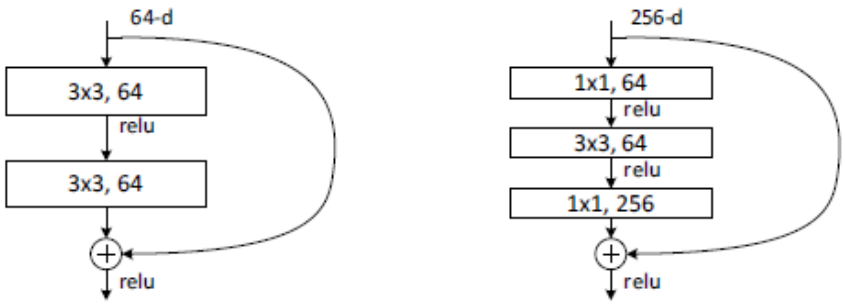


Figure 5. A deeper residual function $\mathcal{F}$ for ImageNet. Left: a building block (on 56×56 feature maps) as in Fig. 3 for ResNet-34. Right: a "bottleneck" building block for ResNet-50/101/152.

Figure 5

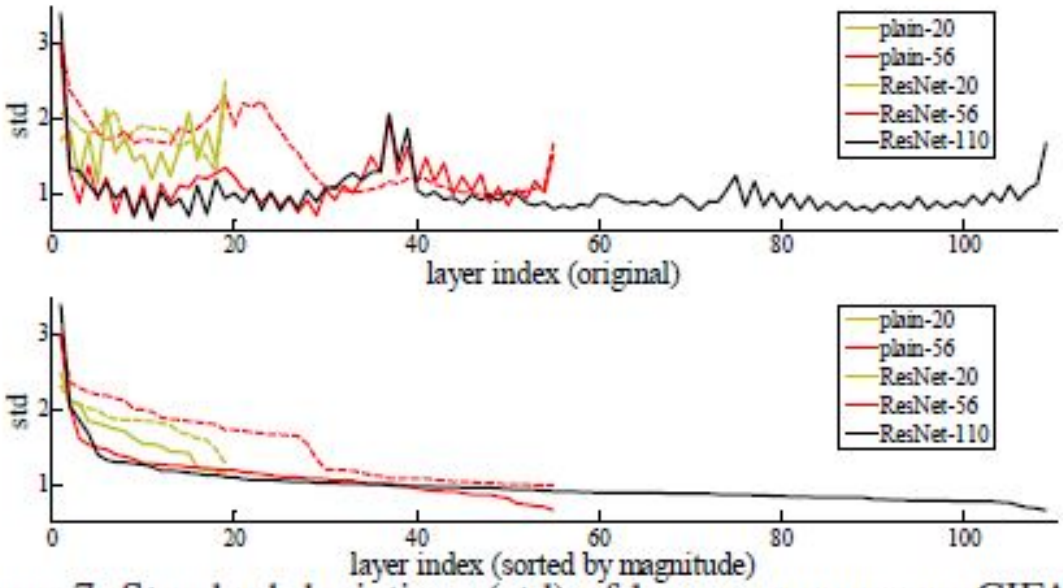


Figure 7. Standard deviations (std) of layer responses on CIFAR-10. The responses are the outputs of each 3×3 layer, after BN and before nonlinearity. **Top**: the layers are shown in their original order. **Bottom**: the responses are ranked in descending order.

Figure 7

APPENDIX B

Appendix Table

model	top-1 err.	top-5 err.
VGG-16 [40]	28.07	9.33
GoogLeNet [43]	-	9.15
PReLU-net [12]	24.27	7.38
plain-34	28.54	10.02
ResNet-34 A	25.03	7.76
ResNet-34 B	24.52	7.46
ResNet-34 C	24.19	7.40
ResNet-50	22.85	6.71
ResNet-101	21.75	6.05
ResNet-152	**21.43**	**5.71**

Table 3. Error rates (%, **10-crop** testing) on ImageNet validation. VGG-16 is based on our test. ResNet-50/101/152 are of option B that only uses projections for increasing dimensions.

method	top-1 err.	top-5 err.
VGG [40] (ILSVRC'14)	-	8.43†
GoogLeNet [43] (ILSVRC'14)	-	7.89
VGG [40] (v5)	24.4	7.1
PReLU-net [12]	21.59	5.71
BN-inception [16]	21.99	5.81
ResNet-34 B	21.84	5.71
ResNet-34 C	21.53	5.60
ResNet-50	20.74	5.25
ResNet-101	19.87	4.60
ResNet-152	**19.38**	**4.49**

Table 4. Error rates (%) of **single-model** results on the ImageNet validation set (except † reported on the test set).

Bibliography

[1] A. Krizhevsky, I. Sutskever, and G. Hinton. Imagenet classification with deep convolutional neural networks. *In NIPS*, 2012.

[2] P. Sermanet, D. Eigen, X. Zhang, M. Mathieu, R. Fergus, and Y. Le-Cun. Overfeat: Integrated recognition, localization and detection using convolutional networks. *In ICLR*, 2014

[3] M. D. Zeiler and R. Fergus. Visualizing and understanding convolutional neural networks. *In ECCV*, 2014.

[4] R. Girshick. Fast R-CNN. *In ICCV*, 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] X. Glorot and Y. Bengio. Understanding the difficulty of training deep feedforward neural networks. *In AISTATS*, 2010.

[7] Y. Bengio, P. Simard, and P. Frasconi. Learning long-term dependencies with gradient descent is difficult. *IEEE Transactions on Neural Networks*, 5(2):157â166, 1994.

[8] K. He, X. Zhang, S. Ren, and J. Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. *In ICCV*, 2015.

[9] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *In ICML*, 2015.

[10] 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*, 1989.

[11] R. K. Srivastava, K. Greff, and J. Schmidhuber. Highway networks. *arXiv:1505.00387*, 2015.

[12] K. He and J. Sun. Convolutional neural networks at constrained time cost. *In CVPR*, 2015.

[13] H. Jegou, F. Perronnin, M. Douze, J. Sanchez, P. Perez, and C. Schmid. Aggregating local image descriptors into compact codes. *TPAMI*, 2012.

[14] F. Perronnin and C. Dance. Fisher kernels on visual vocabularies for image categorization. *In CVPR*, 2007.

[15] H. Jegou, M. Douze, and C. Schmid. Product quantization for nearest neighbor search. *TPAMI*, 33, 2011.

[16] W. L. Briggs, S. F. McCormick, et al. A Multigrid Tutorial. *Siam*, 2000.

[17] C. M. Bishop. Neural networks for pattern recognition. *Oxford university press*, 1995.

[18] B. D. Ripley. Pattern recognition and neural networks. *Cambridge university press*, 1996.

[19] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735â1780, 1997.

[20] Y. Bengio, P. Simard, and P. Frasconi. Learning long-term dependencies with gradient descent is difficult. *IEEE Transactions on Neural Networks*, 5(2):157â166, 1994.

[21] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *In ICLR*, 2015.

[22] A. Krizhevsky. Learning multiple layers of features from tiny images. *Tech Report*, 2009.

[23] C.-Y. Lee, S. Xie, P. Gallagher, Z. Zhang, and Z. Tu. Deeply-supervised nets. *arXiv:1409.5185*, 2014.

[24] M. Tan. Efficientnet: Rethinking model scaling for convolutional neural networks. *arXiv preprint arXiv:1905.11946*, 2019.

[25] a-martyn. Deep Residual Learning for Image Recognition: CIFAR-10, PyTorch Implementation. *GitHub*, 2019. https://github.com/a-martyn/resnet

[26] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. Attention is all you need. *In Advances in Neural Information Processing Systems*, 2017.

[27] W. Xu, Y. L. Fu, and D. Zhu. ResNet and its application to medical image processing: Research progress and challenges. *Computer Methods and Programs in Biomedicine*, 240, 107660, 2023.

[28] G. Xu, et al. Development of Skip Connection in Deep Neural Networks for Computer Vision and Medical Image Analysis: A Survey. *arXiv preprint arXiv:2405.01725*, 2024.

[29] F. Lau. Recognizing and localizing endangered right whales with extremely deep neural networks. *Github*, 2015. https://felixlaumon.github.io/2015/01/08/kaggle-right-whale.html