Residual Neural Networks for Image Recognition

Dhruv Gelda, Joshua Schiller, Anish Shenoy, Pramod Srinivasan

(Dated: December 14, 2016)

In this report we demonstrate the efficacy of existing residual neural network architectures and attempt to improve their accuracy. We successfully reproduce the results of the original authors and explore two other improvements: a residual of residual network with shortcuts bypassing component blocks and a weight initialization scheme that successively trains blocks in the residual neural network. While neither attempt resulted in an increase in accuracy, we gained a deeper understanding about the functioning of residual networks. We also present our own hypotheses as to why these attempts did not perform as we had expected.

I. INTRODUCTION

In recent years, deep convolutional neural networks[1,2] have led to breakthroughs in supervised machine learning problems, particularly for image recognition. Recent observations[3,4] suggest that one of the primary reason for its success is the presence of large number of stacked layers (depth) in the network. The significance of 'depth' of a network is best illustrated by the fact that the highest accuracy architectures results[3–6] on the 1000-class ImageNet dataset[7] have exploited models with depths ranging from sixteen to thirty. This observation begs a natural question: **Is it possible to improve the performance of the model by stacking a greater number of layers ?**

Answering this question is challenged by the problematic phenomena of exploding or vanishing gradients[8,9], which make convergence for deeper networks difficult. However, this problem has been recently addressed by normalized initialization[5,8,10,11] and intermediate normalization layers[6]. Nonetheless, even when the deeper networks start converging, a degradation problem exists. Such behavior is demonstrated by a sample calculation in Fig. 1. While the performance deteriorates only slightly as we go from 32-layer to 56-layer network, the 110-layer network results in a greatly increased error rate. Further, this problem cannot simply be explained by overfitting as addition of more layers to the network leads to a higher training error[12,13]. The degradation of training loss indicates that all models are not similar in regards to the ease with which they can be optimized. This phenomena can be further emphasized with a thought experiment. Consider a shallower architecture and its deeper counterpart constructed by adding more layers onto it. Now, it is always possible to construct a deeper network by copying the initial layers from shallower architecture and stacking additional layers of identity mapping. Such an architecture for the deeper model should not produce a training error higher than its shallower counterpart. However, the experimental observations suggest that the current solvers are unable to find solutions to the deeper networks that are comparable or better than the shallower archi-

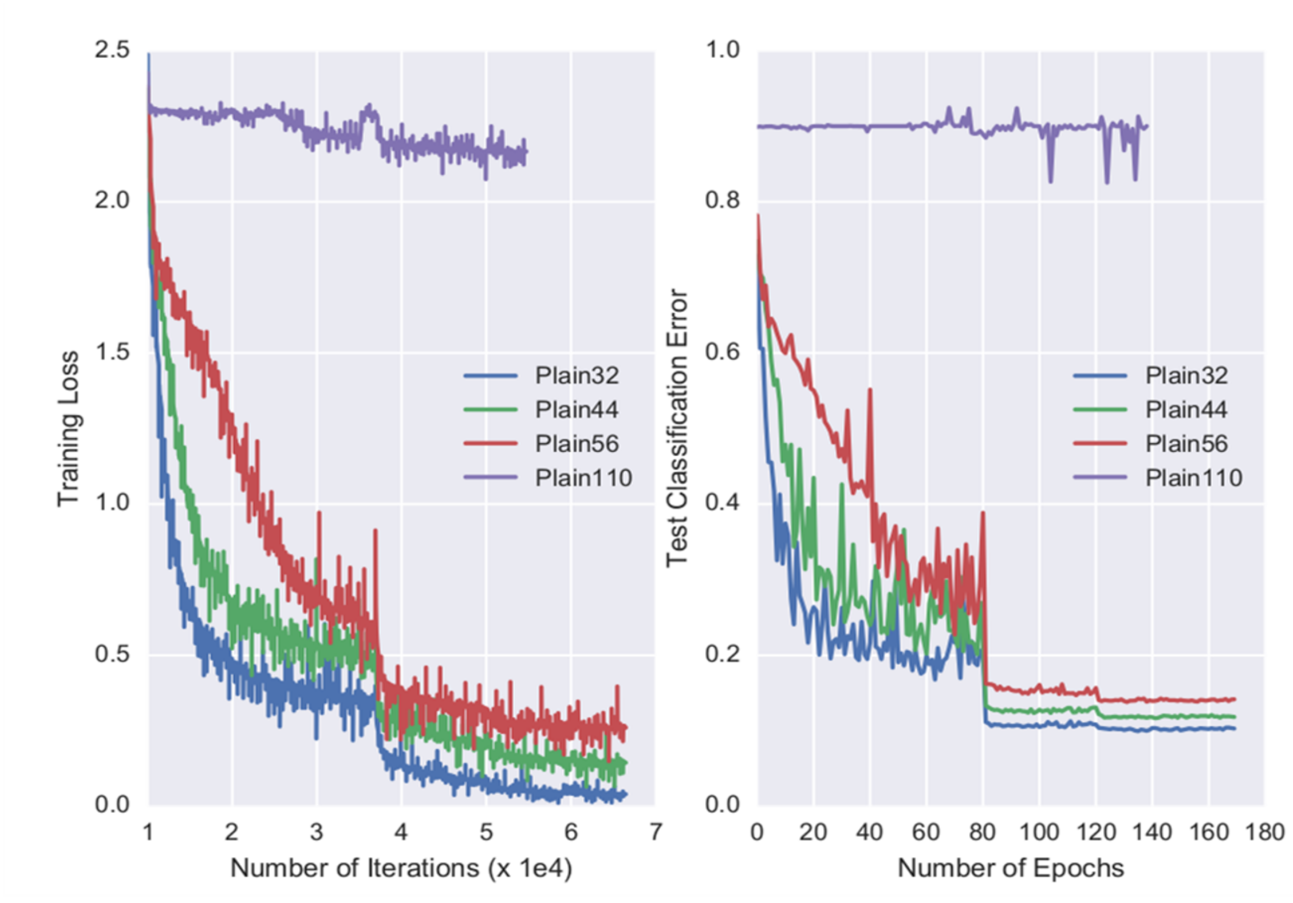


FIG. 1: Training loss (left) and test classification error (right) on CIFAR-10 for 'plain' networks with 32, 44, 56 and 110 number of layers.

tectures. Therefore deep convolutional neural networks (CNNs), despite having the potential of better classification performance, have been found harder to train in practice. Residual networks (ResNets)[5] have been proposed as a solution to tackle these challenges.

A. Model Description

The difference between the building block architecture of plain and residual networks is shown using a schematic in Fig. 2. The conventional plain architecture stack the layers to fit the desired underlying mapping denoted by $H(x)$. On the other hand, instead of mapping an input x directly to an output, the layers of ResNets attempt to fit a residual mapping denoted by $F(x)$ such that the original mapping is recovered by H(x) = F(x) + x. The formulation of $F(x) + x$ has been realized by using "shortcut connections"[14–16] that perform identity mapping. The primary reason behind such a proposition is that the residual mapping $F(x)$ is found to be easier to optimize compared to the original mapping $H(x)$. For example, in an extreme case, if identity mappings were optimal, it would be easier to push the residual to zero than pushing the stack of nonlinear layers to identity mapping.

Apart from ResNets, "Highway" networks[13] have

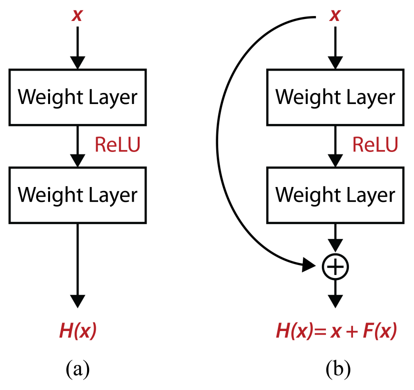
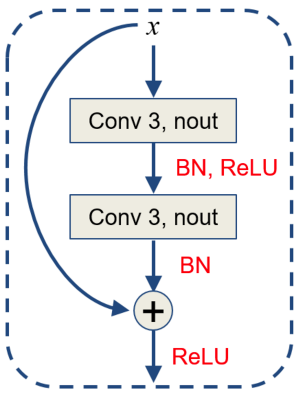


FIG. 2: Bulding block architecture for (a) Plain networks and (b) Residual networks

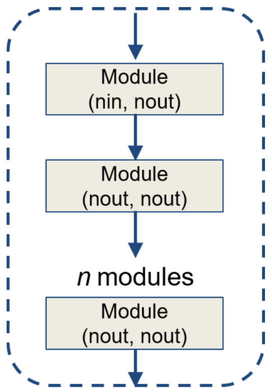
also been proposed to address the degradation problem. This architecture consist of shortcut connections modulated by gating functions . This approach has been inspired by the Long Short Term Memory recurrent neural networks, and uses a learned gating mechanism for regulating information flow. Due to this gating mechanism, a neural network can have paths (information highways) along which information can flow across several layers without attenuation. These networks suffer from the drawback that when gated shortcuts are closed, the layers in the highway networks represent non-residual functions. Moreover, highway networks have not demonstrated accuracy gains with increased network depth.

II. IMPLEMENTING THE RESIDUAL NETWORK

First, the per pixel mean is subtracted for every image in the training and test datasets. During the training phase, each training image was first padded with 4 pixels on each side, and then a random crop of 32x32 pixels is sampled from the padded image. Then, each image is horizontally flipped with a probability of 0.5. The test image was used without any modifications.



(a) Module



(b) Block

FIG. 3: Functional units of Residual Network

The general structure of the ResNet architecture is illustrated in Table I. The basic building unit of a residual network consists of two 3x3 convolutional layers with a shortcut connection (Fig. 2(b)), which we call a residual module. Additionally, each convolutional layer in this module is followed by a batch normalization layer[4]. A fixed number, n, of residual modules arranged sequentially make up a residual “block”(Fig. 3(b)). All the modules in a block have the same number of output channels. In our implementation of ResNet, all the modules have 16

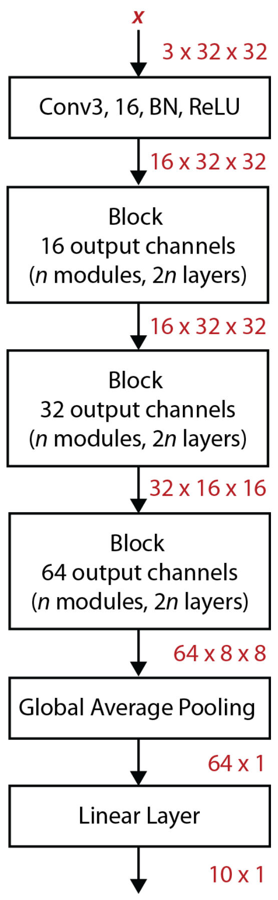


FIG. 4: Diagram of the residual network architecture.

group name	output size	block type $= B(3,3)$
conv1	$16 \times 32 \times 32$	$[3{\times}3,\ 16]$
conv2	$16 \times 32 \times 32$	$\begin{bmatrix} 3{\times}3,\ 16 \\ 3{\times}3,\ 16 \end{bmatrix} \times n$
conv3	$32 \times 16 \times 16$	$\begin{bmatrix} 3{\times}3,\ 32 \\ 3{\times}3,\ 32 \end{bmatrix} \times n$
conv4	$64 \times 8 \times 8$	$\begin{bmatrix} 3{\times}3,\ 64 \\ 3{\times}3,\ 64 \end{bmatrix} \times n$
average pooling	64×1	
fully connected	10×1	

TABLE I: Structure of Residual Network

output channels in the first block (conv2), 32 output channels in the second block (conv3), and 64 output channels in the third block (conv4) (Fig. 4). Since the size of the image halves and the number of channels double moving from one block to the other, the first module in each block transforms the input to make it conform to the required size of the image and number of channels. Practically, this is done by reducing the image size using a stride 2 convolution and stacking additional number of channels with zero entries. These details are also summarized in Table I. These convolution layers are finally followed by a global average pooling layer and a fully connected layer. The architecture always consists of 3 blocks while the number of modules in a block are given by the variable n as mentioned above. Therefore, the total number of layers in the described ResNet architecture are given by $6n{+}2$ layers.

This architecture was implemented using Chainer[17]. We found an existing Chainer implementation of a residual network (`https://github.com/mitmul/chainer-ResNet`). We were not able to run this implementation directly on Blue Waters, and we had to make substantial modifications to the code to replicate the method and architecture described in the paper[5]. For example, the github implementation did not calculate test accuracy correctly, and also used a different method of data augmentation. We implemented the data augmentation strategy as described in the paper, and also made other changes such as He initialization of weights[18], and saving weights from each block for further experiments that we describe in the next section. The data augmentation process greatly reduces overfitting and resulted in a $\sim 4\%$ increase in test accuracy.

We were able to successfully reproduce the test

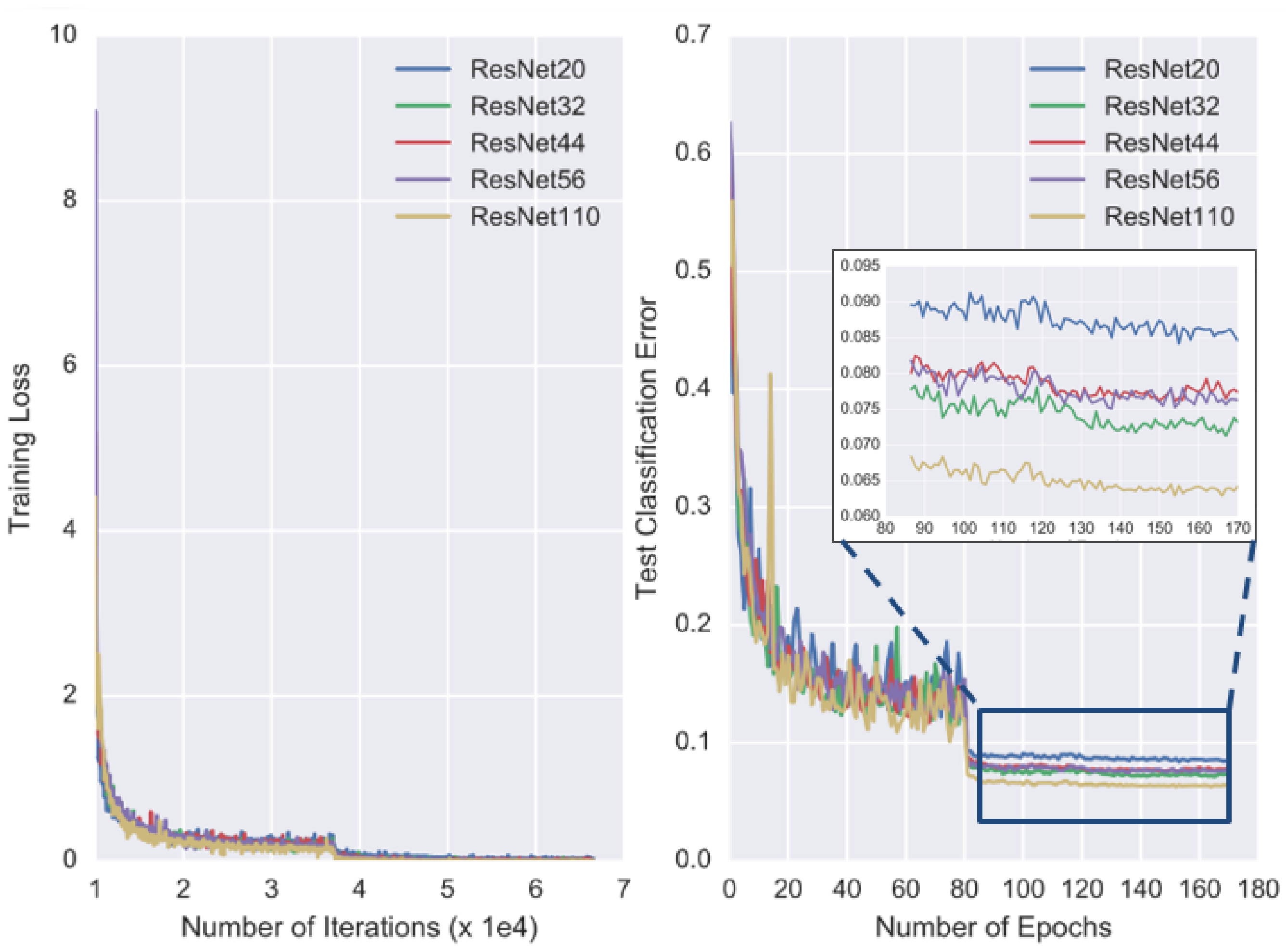


FIG. 5: Training loss (left) and test classification error (right) on CIFAR-10 for residual networks with 32, 44, 56 and 110 number of layers.

Layers	Paper	Our results
20	91.25	91.53
32	92.49	92.66
44	92.83	92.20
56	93.03	92.30
110	93.57	93.59

FIG. 6: Table of final test accuracies for the residual network comparing our implementation with the paper

accuracies on the CIFAR-10 dataset from the paper, as shown in Figs. 5-6.

III. EXPLORING BEYOND RESIDUAL NETWORKS

After successfully replicating the original residual neural network proposed by He et al.[5], we explored two different ideas which go beyond the existing residual network design design.

A. Residual of Residual Neural Networks (ResResNet)

Our first scheme was to create a network that consisted of residual blocks comprised of residual

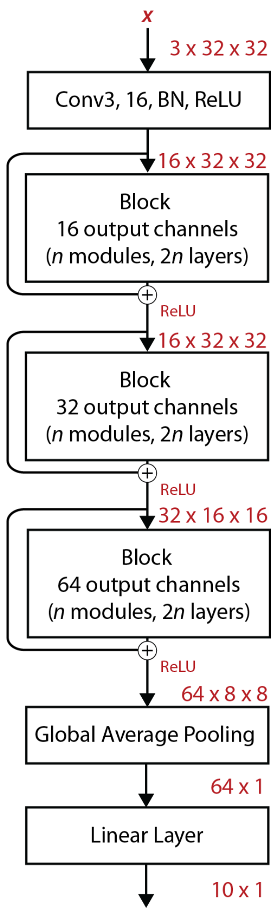


FIG. 7: Diagram of the residual of residual neural network architecture.

layers. The hypothesis was that adding these extra shortcuts were a natural extension of the existing ResNet architecture and the same benefits from the residual layers should translate to residual blocks. At the very least, we felt that the added shortcuts should improve the accuracy for the same reason as the plain ResNet: the intermediate blocks should be able to compensate for any deviation from the ideal output introduced by the addition of the input to the block (i.e. if the desired output were $H(x)$, the block would simply converge to $F(x) = H(x) - x$).

We designed our residual of residual architecture to be comparable to the existing ResNet architecture by keeping the number of blocks and component modules the same for each network depth. Modification of the networks consisted of adding shortcuts from the input to the output of each block and then rectifying the sum (ReLU). Consequently, each architecture only had three shortcuts added regardless of the depth of the individual model. A diagram of the ResResNet is illustrated in Figure 7. We kept the same data augmentation procedure for the ResResNet model, number of epochs, batch size and learning rate decay strategy that was used for the plain and ResNet architecture experiments.

From the results in Figure 9, it can be seen that our hypothesis did not prove to be correct. As can be seen Figure 9(b), None of the architectures we attempted performed as well as the normal ResNet architecture. Figure 9(a) illustrates that all of the models had very low losses at the end of their run, indicating that models fit the training data very well. Nonetheless, an interesting pattern emerged, whereby the extremely shallow (20 layer) and extremely deep (110 layer) networks had higher accuracies than the intermediately sized networks. We believe this occurs for a different reason for each of the shallow and deep networks. In the intermediate depth networks, we believe the decreased accuracy is the result of over-fitting the models to the data, which is supported by the extremely low training loss for the corresponding models at the end of the runs. We hypothesize that the limited number of variable parameters in shallow model reduced the impact of this over-fitting. In essence, we believe that whatever over-fitting that occurred as a result of the added shortcuts was partially com-

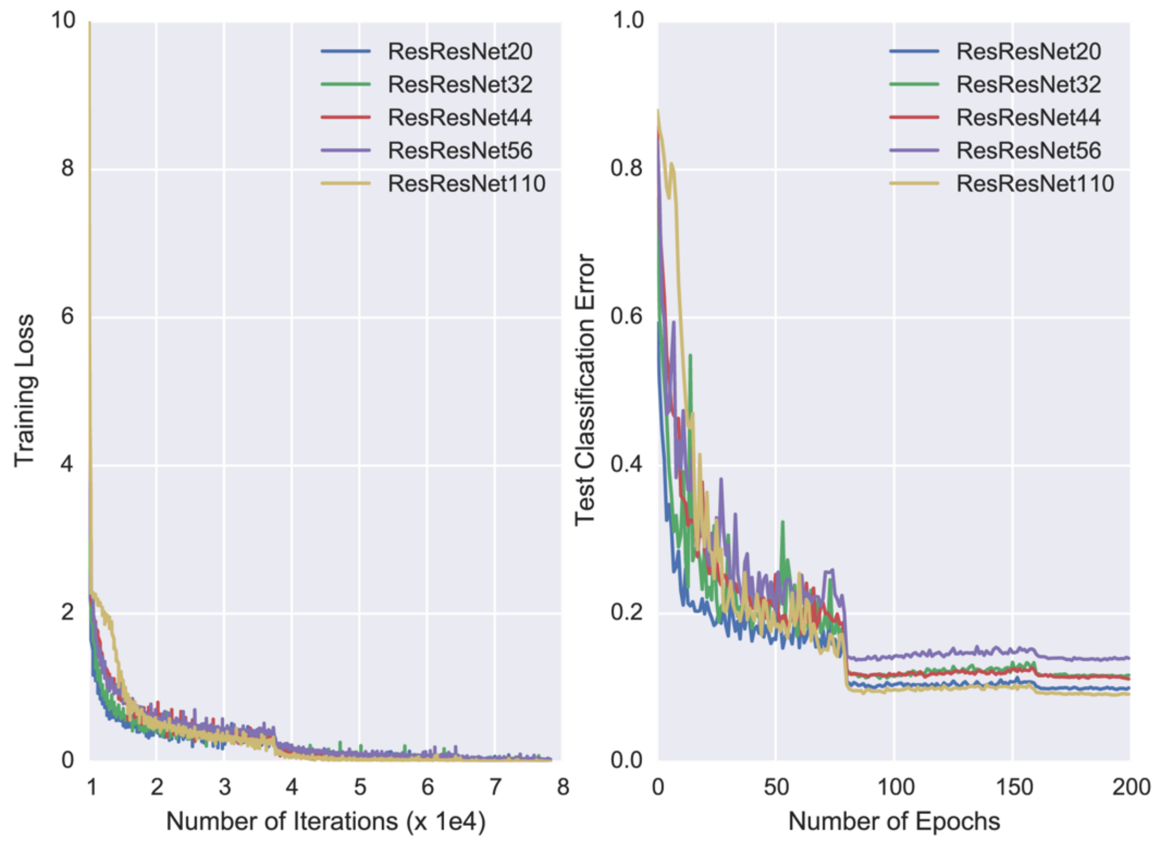


FIG. 8: Loss and accuracy of the residual of residual neural network for different size architectures

Layers	Plain	ResNet	ResResNet
20	90.61	91.53	90.05
32	88.22	92.66	88.32
44	87.49	92.20	88.82
56	85.81	92.30	86.03
110	9.98	93.59	90.89

FIG. 9: Table of final accuracies for the residual of residual network compared to the plain and residual network architectures.

pensated by the limited flexibility of the underlying shallow architecture. However, as the architecture increases in depth, the added complexity from the additional shortcuts magnified the complexity of the model and inhibited regularization. However, the negative effects became diminished for the very deep network. With only three added shortcuts for a 110-layer model, this may be because the added shortcuts had less of an effect on such an otherwise complicated model. Nonetheless, it is clear that the strategy of creating residual blocks of residual layers does not seem to be as successful as the tradition ResNet scheme.

B. Initialization of Weights by Partial Training

The intuition behind the mechanics of residual neural networks is that it is easier to model a small perturbation to an input x with the function $F(x) = H(x) - x$ and add the perturbation to input than to model the function $H(x)$ itself. With this in mind, we hypothesized that successive layers might perform better if the previous layers were pre-trained as then deeper layers would only need to model the perturbation. To test this hypothesis we successively trained each of the three blocks in a 32-layer residual neural network, using the same architecture as described earlier.

In the top-down case, an initial model was made with only the top block of the architecture, with its output dimensions coaxed to be compatible with the output of the network with average pooling and the appropriate number of output channels. After training the model on just this top block, the next block was trained with its output made to be compatible with the output layer. Finally, the third block was added and the model was trained again. For the bottom-up case, the same strategy was used, but the blocks were added starting from the bottom of the architecture upwards. For both cases, the first two additions of blocks were trained for 50 epochs with a learning rate of 0.1, which reduced to 0.01 after 40 epochs. After the final block was added, the model was run for 100 epochs, with the learning rate decaying from 0.1 to 0.01 at 40 epochs and then to 0.001 at 80 epochs. As with the previous models, a batch size of 128 was used.

After training the ResNets with this initialization strategy, the bottom-up architecture resulted in an accuracy of 92.15, comparable to the existing architecture, while the top-down had a reduced accuracy of 89.82. We hypothesize the learning rate increase and random distribution of weights after adding a block may have counteracted any benefit from the initialization technique. The increased learning rate may have pushed the pre-initialized blocks away from their converged weights, but more importantly the weights in the new block had weights that may have interfered with the old blocks. In the future, it may be worthwhile to have successive blocks initialized with weights scaled by a factor $\lambda \ll 1$. In this way, the layers would more closely model a perturbation to the output of the previously trained layers.

Another interesting phenomenon is that the top-down approach performed more poorly than the bottom-up one. This is puzzling as one would expect the architectures to have an accuracy that is, at a minimum, comparable to the original ResNet approach. We hypothesize that this is due to the added blocks in the top-down approach not sufficiently converging their weights. This may indicate that the weights in deeper layers are much less plastic than the top layers that nearer to the input. Consequently, the added top blocks can compensate for any deficiencies in the lower layers, but the reverse is not as easy. Even though a better accuracy was not obtained for these architectures, the result is still a potentially intriguing glimpse of the inner-workings of deep neural networks.

IV. CONCLUSION

We were successful in creating a Chainer implementation[17] for ResNets. We showed that ResNets have the ability to overcome the degradation problem and make the optimization of deeper

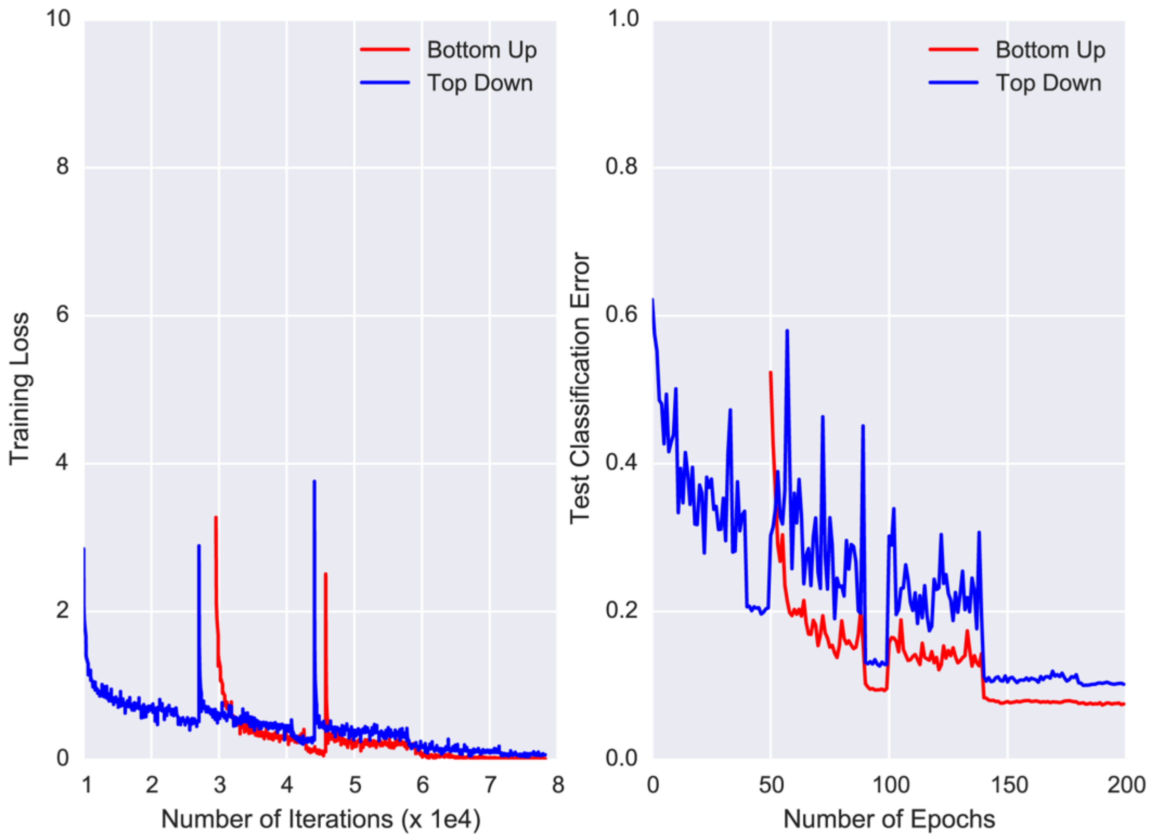


FIG. 10: Loss and accuracy of the 32-layer, top-down and bottom-up weight initialization schemes.

networks possible. We showed the convergence of ResNet architectures with number of layers ranging from 32 to 110. We showed that it was possible to achieve a gain in performance with increasing depth. Although, we have evaluated our ResNet architectures on CIFAR-10 dataset, the inference regarding performance gain with increasing depth holds true in general. Our test classification performances on CIFAR-10 data set are very similar to the existing literature[5].

Furthermore, we also explored two modifications to the original residual network. Firstly, we implemented additional shortcut connections at the module level, calling this new architecture a "residual of residual network". Our hypothesis was that the benefits of shortcut connections at the module level should also translate to block level, and therefore, the new architecture should perform at least as well as the original ResNet. However, in our experiments we observed that the new architecture failed to outperform or even match the original ResNet architectures, irrespective of the depth of the network. Even though the residual of residual networks had very low training losses, they had lower test accuracies compared to the residual networks. We believe that the poor performance is caused by a loss of regularization resulting from the added complexity introduced by the new shortcut connections. Moreover, shortcut connections at the module level are found to have a lower effect for deep networks.

Next, we tried two different approaches for initializing weights during training. In the first approach, we proceeded in top-down fashion, by initially training a network with only the first block being present, then added a second block and retrained, and finally the third block. In the second approach, we performed a similar procedure but in the reverse direction (bottom-up). Here, we first trained a network with just the third block being present, then added the second block and retrained and so on. The top-down approach resulted in a lower accuracy compared to the original ResNet, whereas the bottom-up strategy resulted in a comparable accuracy. Importantly, we observe that when training using the top-down approach, the lower accuracy is caused because the weights trained in the upper blocks do not adapt when lower blocks are added. On the other hand, during bottom-up training, the weights in the lower blocks are more pliable and adapt as the higher blocks are added, which allows this method to perform comparably to the original architecture.

In conclusion, this project allowed us to gain a practical understanding of residual networks. We also gained insights into the inner workings of the residual blocks by implementing additional residual connections and by using two different approaches for initializing weights.

[1] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[2] Yann LeCun, Bernhard Boser, John S Denker, Donnie Henderson, Richard E Howard, Wayne Hubbard, and Lawrence D Jackel. Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1(4):541–551, 1989.

[3] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[4] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–9, 2015.

[5] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *arXiv preprint arXiv:1512.03385*, 2015.

[6] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

[7] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 115(3):211–252, 2015.

[8] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Aistats*, volume 9, pages 249–256, 2010.

[9] Yoshua Bengio, Patrice Simard, and Paolo Frasconi. Learning long-term dependencies with gradient descent is difficult. *IEEE transactions on neural networks*, 5(2):157–166, 1994.

[10] Andrew M Saxe, James L McClelland, and Surya Ganguli. Exact solutions to the nonlinear dynamics of learning in deep linear neural networks. *arXiv preprint arXiv:1312.6120*, 2013.

[11] Patrice Y Simard, Yann A LeCun, John S Denker, and Bernard Victorri. Transformation invariance in pattern recognitiontangent distance and tangent propagation. In *Neural networks: tricks of the trade*, pages 239–274. Springer, 1998.

[12] Kaiming He and Jian Sun. Convolutional neural networks at constrained time cost. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5353–5360, 2015.

[13] Rupesh Kumar Srivastava, Klaus Greff, and Jürgen Schmidhuber. Highway networks. *arXiv preprint arXiv:1505.00387*, 2015.

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

[15] William N Venables and Brian D Ripley. Modern applied statistics with s-plus. *Springer: New York*, 1999.

[16] Brian D Ripley. *Pattern recognition and neural networks*. Cambridge university press, 2007.

[17] Seiya Tokui, Kenta Oono, Shohei Hido, and Justin Clayton. Chainer: a next-generation open source framework for deep learning. In *Proceedings of Workshop on Machine Learning Systems (LearningSys) in The Twenty-ninth Annual Conference on Neural Information Processing Systems (NIPS)*, 2015.

[18] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1026–1034, 2015.