Highway Networks

ECBM4040.2021Spring.HIGH.report

Kaan Yarali ky2446, Zoran Kostic zk2172

Columbia University

***Abstract*— Both shallow and deep neural networks are capable of approximating any functions; there are several theoretical and empirical evidence that the depth of neural networks is quintessential for their success. However, the training process becomes more difficult as the number of layers increases. The authors of the original paper introduce a new architecture to ease the gradient-based training and solve the notorious vanishing gradient problem. In the proposed architecture, there are unimpeded connections between hidden layer inputs and outputs called information highways. This architecture was inspired by the gating mechanism of Long Short-Term Memory (LSTM) recurrent neural networks, which allow information to flow through several layers without attenuation. The proposed architecture was the first working deep feedforward neural networks with over 100 layers. In this project, the goal was accomplished since I got very similar results to those reported by the authors. In the original paper, there is a webpage link for the project where they provided the hyperparameter settings of the networks and the architectures of the convolutional Highway networks. However, the weblink has been broken, so I did a random search for the hyperparameter settings and obtained very close results.**

I. Introduction

Deep neural networks have become very popular in recent years because of their phenomenal success in supervised machine learning problems. In traditional machine learning models, a handcrafted feature extractor gathers relevant information from the input, and a classifier categorizes the resulting feature representations. Since the features are handcrafted, the models' success is bottlenecked by the lack of success of the feature extractor in terms of representation of the input. The significant advantage of deep learning over traditional machine learning models is its capacity to do representation learning, which is the process of automatically learning representations of input data for detection or classification. Composing simple yet non-linear modules that each transforms the representation at one level produces multiple levels of representation (starting with the input) [2]. More complex and powerful features are extracted as the depth of the network increases. For example, in VGG16 and VGG19 [3], smaller filters such as 3x3 CONV layers were preferred instead of 7x7 CONV layers in Alexnet [4]. Stacked three layers of 3x3 CONV have the same receptive field size with a single 7x7 CONV layer, but the features learned by stacking CONV layers contain non-linearities that make their feature more expressive.

In theory, deep networks can represent certain function classes exponentially more efficiently than shallow ones. The use of deep networks can offer both computational and statistical efficiency for complex tasks [5]. Yet, there are challenges in training deep networks because the optimization becomes more difficult as the number of layers increases. This problem leads to research of efficient variance preserving initialization schemes, Glorot Uniform and He Normal [6, 7] and techniques of training networks in multiple stages [8] or with temporary companion loss functions attached to some of the layers [9].

In the aforementioned paper, the authors presented a novel architecture that allows training deep networks as deep as 900 layers. This was accomplished by the gating mechanism for information flow through consecutive layers, inspired by the Long Short-Term Memory (LSTM) recurrent neural networks [10]. Because of this gating mechanism, information can flow through several layers without attenuation. They call such paths as information highways and such networks as highway networks [11]. In the experiments, both highway and plain networks were optimized using Stochastic Gradient Descent (SGD) with momentum. However, optimization of plain networks is highly dependent on the weight initializations, type of activation function used, and the depth of the network. Plain networks cannot be trained without use of variance preserving weight initializations [6,7]. On the other hand, highway networks are independent of those factors.

II. Notation

Vectors and matrices are denoted by boldface letters, while transformation functions are denoted by italicized capital letters. Vectors of zeros and ones are denoted by $\mathbf{0}$ and $\mathbf{1}$, respectively, and an identity matrix is denoted by $\mathbf{I}$. The function $\sigma(x)$ is known as $\sigma(x) = \frac{1}{1+e^{-x}}, x \in R$ [11].

A. Highway Networks

In plain feedforward neural networks, a linear transform of input x_l is followed by the non-linear activation function H to produce output y_L in the lth layer [11]. For convenience, the input of network is denoted as x_1 and the output of network is denoted y_L since it consists of L ($l \epsilon \{1,2,..L\}$) layers. This linear transform operation is parametrized by the W_{Hl} Overlooking the indexes,

$$y = H(x, W_H) \tag{1}$$

For Highway networks, two non-linear transforms $T(x, W_T)$ and $C(x, W_C)$ are additionally defined for the gating mechanism. The block output y is formulated as

$$y = H(x, W_H).T(x, W_T) + x.C(x, W_C). \tag{2}$$

T is referred to as the transform gate, and C is referred to as the carry gate, since they convey how much of the output is generated by transforming and carrying the input, respectively [11]. In my project and also in this paper, C is set as C=1-T for simplicity. Therefore, the output y is equal to

$$y = H(x, W_H).T(x, W_T) + x.(1 - T(x, W_T)). \tag{3}$$

The dimensionality of x, y, $H(x, W_H)$ and $T(x, W_T)$ have to be the same for Equation 3. $T(x)$ is defined as $T(x) = \sigma({W_T}^T x + b_T)$, where W_T is the weight matrix and b_T is bias of the transform gate [11]. Since the output of the sigmoid activation function is between 0 and 1, it decides how much of the output is produced by transforming the input. For endpoints of $T(x)$:

$$y = \begin{cases} x, & if\ T(x, W_T) = 0 \\ H(x, W_H), & if\ T(x, W_T) = 1 \end{cases} \tag{4}$$

The initialization of the bias of the transform gate is different than the other parameters of the network. It can be initialized with a negative value such as between -1 and -10 to make network initially biased towards carry behavior which is inspired by the initialization scheme of the LSTM recurrent network [10].

The Jacobian of the layer transform,

$$\frac{dy}{dx} = \begin{cases} I, & if\ T(x, W_T) = 0 \\ H'(x, W_H), & if\ T(x, W_T) = 1 \end{cases} \tag{5}$$

The vanishing gradient problem is solved because of the skip connections added to the network. Even if the output of the layer is not produced by the transformation of the input, the gradient of next layer multiplied by the identity matrix is backpropagated through backwards layer.

The number of parameters of the network can be substantial if the input x has a large size and the highway layer is attached to the network before reducing the dimensionality of the input. Since the dimensionality of x, y, $H(x, W_H)$ and $T(x, W_T)$ have to be the same, one can reduce the dimensionality of the input x by sub-sampling or zero padding and pass the x' to the highway layer as a new input. In my project and the aforementioned project, a plain layer without highway is used in order to lower the dimensionality and is followed by stacked highway layers. For convolutional highway networks, "same" convolution is applied for this purpose.

In the experiments, it was shown that negative bias initializations of transform gate biases were sufficient for training deep neural networks whereas plain networks should be optimized very carefully. Although variance preserving weight initializations were applied for plain networks, most of the plain networks could not be trained except for the network using tanh activation function. The reasons for this problem have not been elucidated in the aforementioned paper, but I did some analysis to explore the potential reasons given in the IV. Section of the current report.

III. Results of the Paper

They trained very deep plain networks and highway networks on the MNIST digit classification dataset to compare the cross-entropy error on the training set and do not compare the generalization performance on the test set. A fully connected layer is followed by 9,19,49, or 99 fully connected or highway layers and a single softmax output layer. The number of neurons is set to 71 and 50 in plain network and Highway network respectively, in order to keep the number of parameters roughly same for both networks. The hyperparameters were set after a random search of 40 runs for both plain and highway networks. However, I could not obtain these hyperparameter settings since the weblink has been broken.

SGD with momentum and exponential weight decay were used in both networks. The initial learning rate, momentum, learning rate decay rate, activation function for H either ReLU or tanh and the transform gate biases (between -1 and -10) were optimized. Plain networks cannot be trained without variance preserving weight initializations [6,7] so Glorot Uniform [6] were used.

Figure 1. depicts the convergence plots for the best performing networks at each depth. Although highway network performs admirably at all depths, plain network tends to run into training degradation problems after depth 10. The 100-layer highway network's final performance is around 1 order of magnitude higher than the 10-layer network, and it's on par with the 10-layer simple network [11]. Even though it has not been quoted that tanh activation function was used for plain network, I inferred that this activation function has been used through my analysis, as given in part IV.

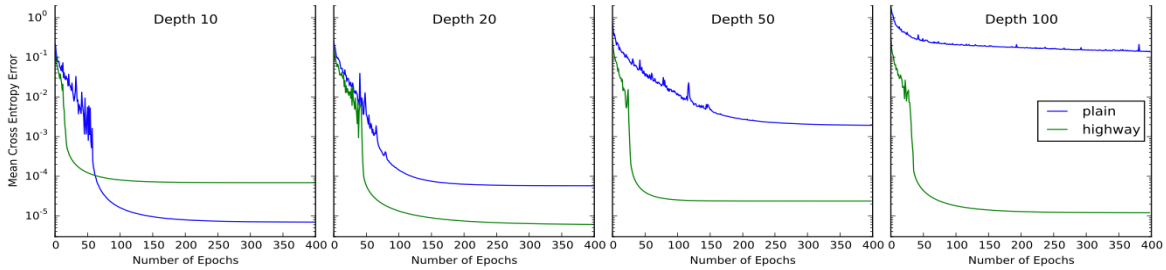

Figure 1. The optimization of plain networks and Highway networks of different depths is compared. The curves depicted are for the best hyperparameter settings obtained from a random quest for each configuration. However, those hyperparameter settings have not been reported. Optimization of plain networks becomes more difficult as the number of layer increases and the mean cross entropy error increased gradually.

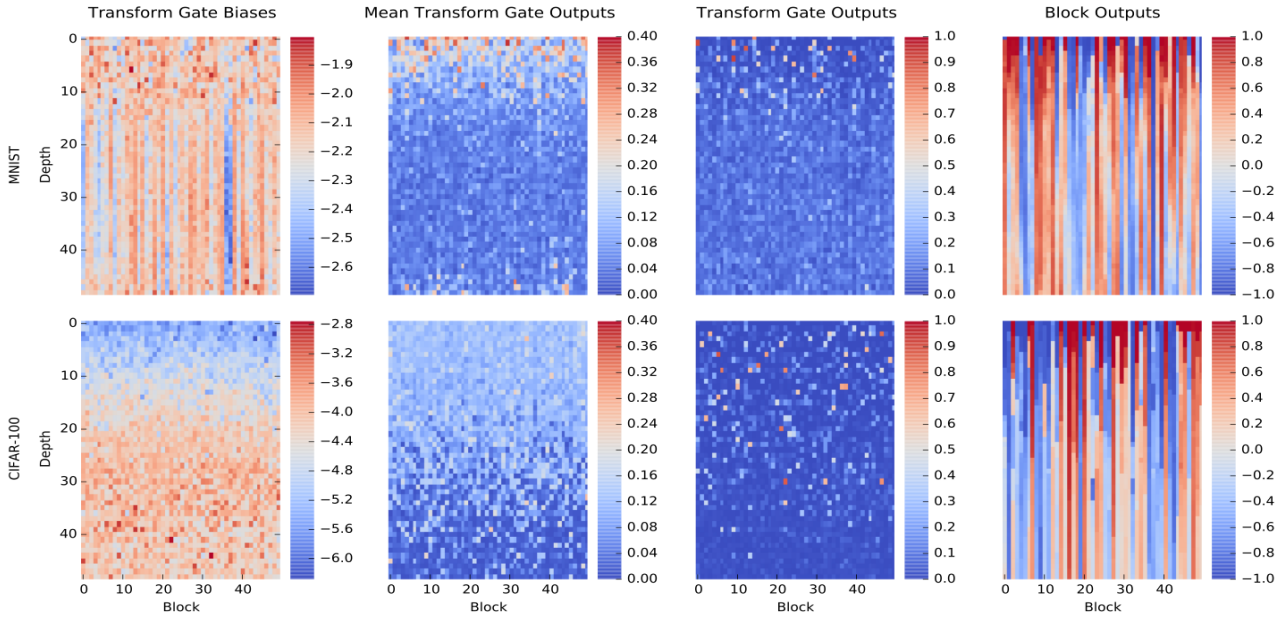

Figure 2. Visualization of the transform gate biases, mean transform gate outputs, transform gate output and block outputs of highway networks trained on MNIST and CIFAR 100 respectively. In both networks, the first layer is a dense layer (without skip connection) to lower the dimensionality to 50, and 49 highway layers are stacked after the first dense layer. Therefore, the networks are 50 layers excluding the final dense layer and each layer consists of 50 neurons. In this heatmap, the neuron values were plotted for each layer. Transform gate biases were initialized as -2 and -4 respectively. The first columns depict the transform gate biases after the training. The second column shows the mean transform gate outputs of 10000 random samples. Transform gate output and block output of a random sample drawn from training set are exhibited in the 3rd and 4th column.

They also visualized 50 transform gate biases for each of the 50 hidden layer Highway networks trained on MNIST and on CIFAR 100. In Figure 2, the first three columns display the bias, mean behavior over 10K random samples, and activity for a single random sample for each transform gate. The last column displays the block outputs for the same single sample.

In Figure 2, they showed some of the innerworkings of 50-layer Highway networks trained on CIFAR100 (bottom row) and MNIST (top row). The first three columns indicate the bias, mean behavior over 10K random samples, and activity for a single random sample for each transform gate. The last column displays the block outputs for the same single sample. The transform gate bias was set to -2 for network trained on MNIST and -4 for network trained on CIFAR 100. Of interest, in the heatmap, most of the transform gate biases became more negative during training. For CIFAR 100, transform gate biases increased in successive layers. The authors explained this issue as the network became more selective in early layers that made

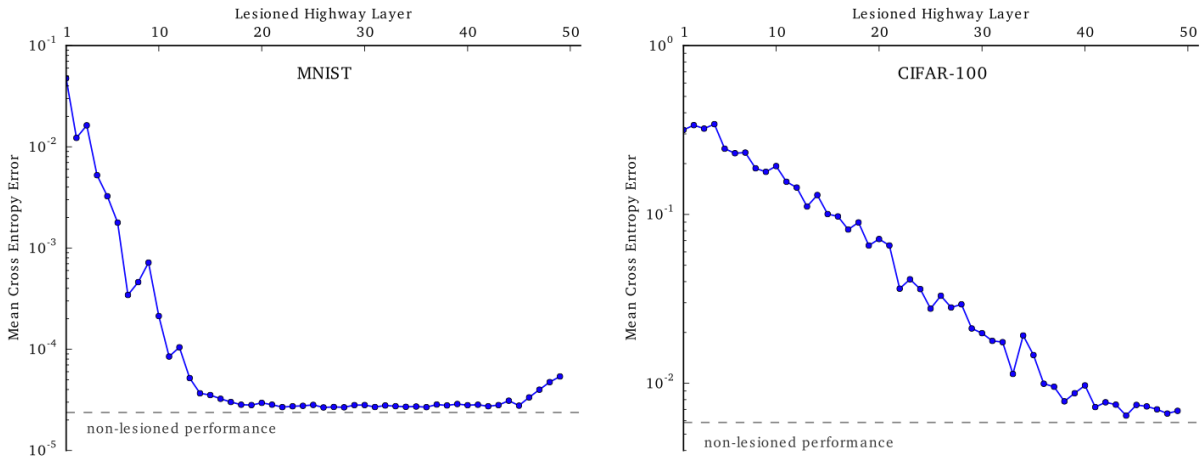

Figure 4. The mean cross entropy error of the lesioned Highway networks trained on MNIST and CIFAR 100 as a function of removed layer (x-axis) is depicted. A dotted line at the bottom represents non-lesioned results.

the most of the transform gate biases decrease. However, in MNIST, transform gate biases of mid layers decreased and they became "information highways" effectively as they only pass the input to successive layers without transforming it. The "information highways" can be seen very clearly from the last column of Figure 2 as most of the outputs stay constant over many layers, which make stripes in the plot.

There is a long version of the aforementioned paper where the authors make further analysis about the layer importance of the Highway networks [12]. I also added this part to my project. Therefore, I would like to show those results as well, in order to compare my results and their results in the IV. section. The authors initialized the transform gate biases to negative values to bias the network towards copying inputs and passing them to successive layers without any transformation. However, this issue raises a question whether the final network is equivalent to a network with much fewer layers. They iteratively remove a single highway layer or set to transform gate output to 0 at each step to observe the effect of the removed layer by comparing the mean cross entropy loss with the original network. They refer to this operation as *lesioning* the network. Figure 4. depicts the resulting performance of the lesioned network due to the lesioned layer. For MNIST (left), if some of the early layers are discarded from the network, the error increases dramatically, but layers 15 through 45 seem to have little impact on the final performance. Most of the mid-layers behave as a skip connection and do not contribute to the output since the MNIST is an easy dataset which does not require complex representations compared to CIFAR100. However, for CIFAR100, training performance degrades notably if any of the first 40 layers are removed. Therefore, Highway network can learn to use many of its layers for complicated problems, such as CIFAR100, whereas it uses most of the layers as skip connection for the simple problems, such as MNIST.

IV. My Results & Comparison with the Original Paper

MNIST and CIFAR 100 datasets were used for this project. The input images are normalized by dividing by 255 to rescale pixel values 0-255 to 0-1. The learning process is accelerated by normalizing the input since higher learning rates can be used and the ravines on the loss contours turn into spherical loss contours.

I was amazed by the performance of the plain network since it can be trained without highway layers, although the mean cross-entropy error is relatively high compared to the Highway network. I trained multiple deep plain networks using different activation functions to observe their training performance even though the authors of the aforementioned paper did not make such an analysis. Although variance preserving weight initializations, Glorot Uniform and He Normal, [6,7] are applied, training a deep plain network is a very complex problem. For example, the training of the plain network using ReLU activation function and Glorot Uniform [6] has failed because of the dead ReLU problem. The mean and the variance of activations gradually decrease layer by layer and become 0 in final layers, so the network cannot be trained as gradient propagating backwards is 0. Figure 5 depicts the mean activation value of each layer in the 50-layer plain network using the ReLU activation function. Also, Figure 6 depicts the variance of activation value of each layer in the 50-layer plain network using the ReLU activation function. However, if He Normal initialization [7] is applied, the network can be trained by carefully setting the hyperparameters. One potential problem for this network is, exploding gradient problem since the gradients of the activation function (ReLU) are unbounded, and the weights of the network can potentially diverge. Gradient clipping or exponentially decaying learning rate methodologies can be applied to overcome this problem.

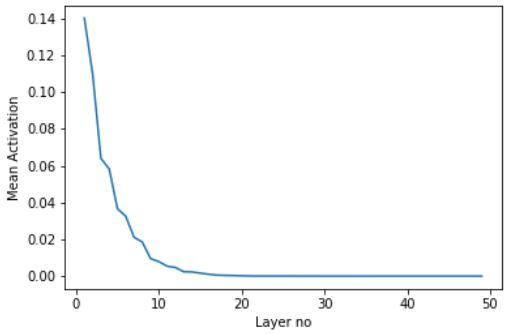

Figure 5. The mean activation of each layer is shown. The mean activation converges to 0 as the number of layer increases.

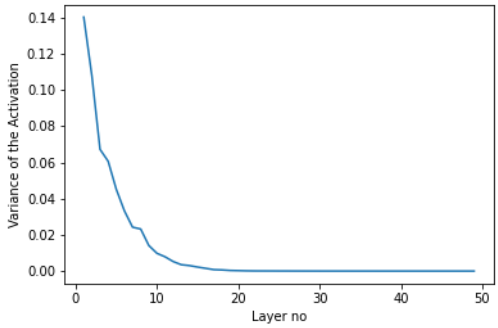

Figure 6. The variance of the activation of each layer is shown. The mean activation converges to 0 as the number of layer increases.

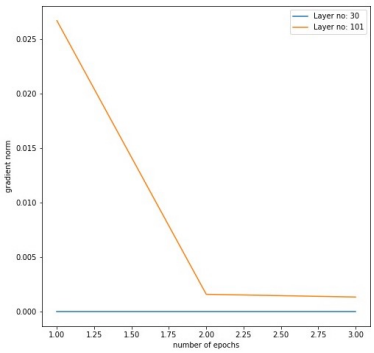

Figure 7. The L2 Norm of the gradient vector of layers 101 and 30 are shown. Layer 30 has a 0 gradient because of the vanishing gradient problem.

I also trained two deep plain networks (50 layers) using the sigmoid activation function with both Glorot Uniform and He Normal weight initializations [6,7], and these networks could not be trained either. However, this time a different problem arises. The network could not be trained because of the vanishing gradient problem. Although the variance of the activation of layers did not become exactly zero, the gradient becomes 0 for the early layers. In Figure 7, L2 norm of the gradient is shown accordingly to each layer. Therefore, I inferred that the authors used the tanh activation function for plain networks in Figure 1 because it does not suffer from vanishing gradient problem and variance of each layer remains in the same range.

After deciding tanh activation function for plain networks, I trained very deep plain networks and Highway networks on the MNIST digit classification dataset to compare the cross-entropy error on the training set. The first layer is a fully connected layer and is followed by 9, 19, 49 or 99 fully connected or highway layers and a single softmax output layer [11]. The number of neurons is set to 71 and 50 in the plain network and Highway network, respectively, to keep the number of parameters roughly the same for both networks. The authors did not share the hyperparameter settings, so I did a cross validation for those settings. SGD with momentum and exponentially decaying learning rate were used for both of the networks. For highway networks, the lowest mean cross-entropy loss was achieved using the transform gate bias, momentum, learning rate and the decay rate as -4, 0.9, 0.1 and 0.96 (decay step = 100,000), respectively. Training plain networks is not as straightforward as training highway networks. Each of the hyperparameters need to be optimized differently for each configuration. For 10-layer and 20-layer plain networks, I used a learning rate of 0.001. However, I had to decrease the learning rate to 0.00001 for 50-layer and 100-layer plain networks to train the network properly. Momentum and the decay rate are kept at a constant value of 0.9 and 0.96 (decay step = 100,000) respectively, for all configurations in the plain networks. I believe the authors used an exponentially decaying learning rate to overcome the exploding gradient problem, although the vanishing gradient problem disappears because of the skip connections added to the network.

In Figure 8, the training curves for the best performing networks for each depth are exhibited. As the results reported in the original paper, highway networks are resistant to suffering from training degradation problem as the number of layers increase. On the other hand, plain networks show good performance at 10 and 20 layers, because the mean cross entropy loss is between 10^{-3} and 10^{-4}. 100-layer highway network performed more than two orders of magnitude better compared 100-layer plain network. As inferred from Figure 8, highway networks converge dramatically faster than plain ones. The results in the original paper as presented in Figure 1 are very close to my results presented in Figure 8. There might be small differences, such as slight oscillations in the loss graphs of the plain networks due to the unknown hyperparameter settings of the original paper.

I have visualized 50 transform gate biases for each of the 50 hidden layer Highway networks trained on MNIST and on CIFAR 100. In Figure 9, the first three columns display the bias, mean behavior over 10K random samples, and activity for a single random sample for each transform gate. Block outputs for

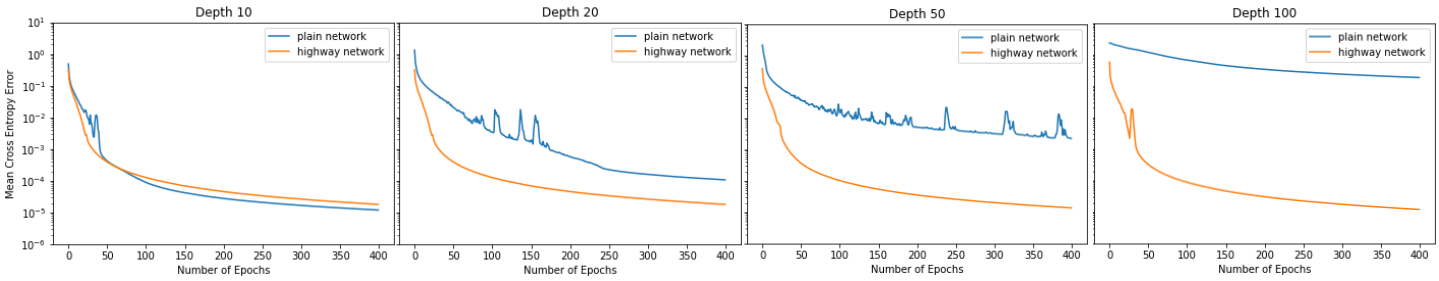

Figure 8. The optimization of plain networks and Highway networks of different depths is compared. The mean cross entropy error increases as the number of layers increase in plain networks while the Highway network does not suffer from this problem. At depth 100, Highway network's mean cross entropy error is at the same scale as the error of depth 10 of plain network.

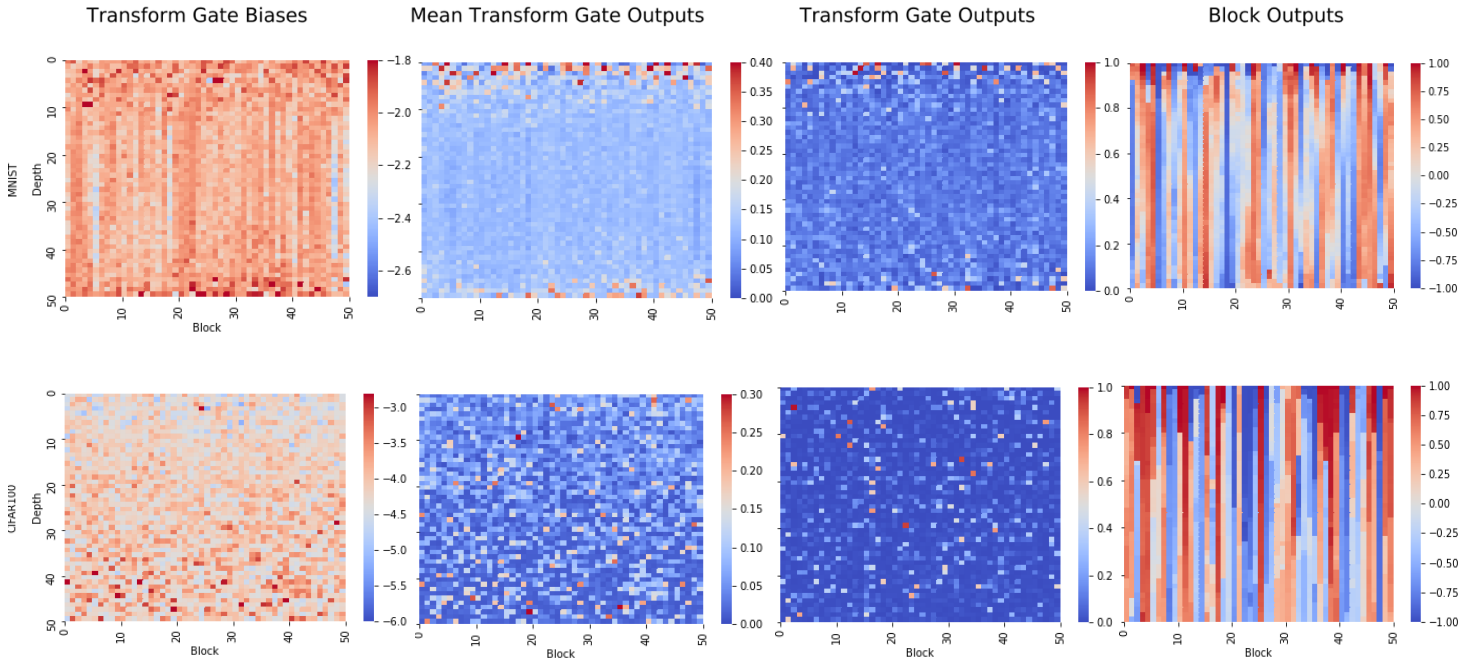

Figure 9. Visualization of the transform gate biases, mean transform gate outputs, transform gate output and block outputs of highway networks trained on MNIST and CIFAR 100 respectively. The architectures are same with the ones reported in Figure 2. In this heatmap, the neuron values were plotted for each layer. Transform gate biases were initialized as -2 and -4 respectively. The first columns depict the transform gate biases after the training. The second column shows the mean transform gate outputs of 10000 random samples. Transform gate output and block output of a random sample drawn from training set are exhibited in the 3rd and 4th column.

the same single sample is presented on the last column. For both of these networks, as their hyperparameter settings except the transform gate bias are not disclosed, which led me to optimize them, and resulted in slight differences. For the 50-layer highway network, I set the transform gate bias, momentum, learning rate and the decay rate as -2, 0.9, 0.1 and 0.96 (decay step = 100,000) respectively. For the 50-layer plain network, I set the transform gate bias, momentum, learning rate and the decay rate as -4, 0.9, 0.0001 and 0.96 (decay step = 100,000), respectively. I trained the plain networks for 1000 epochs, whereas I trained the Highway network for 400 epochs since CIFAR 100 is a more complex dataset.

In Figure 9, The first three columns indicate the bias, mean behavior over 10K random samples, and activity for a single random sample for each transform gate. The last column displays the block outputs for the same single sample. The transform gate bias was set to -2 for network trained on MNIST

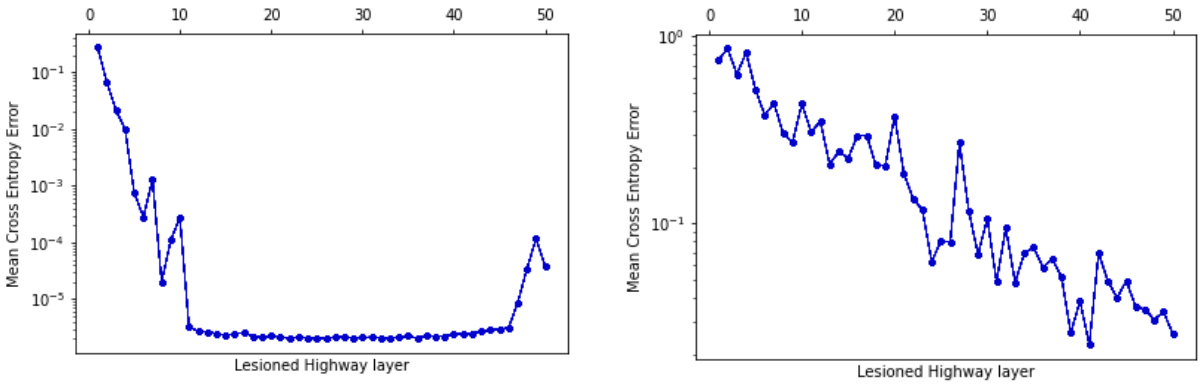

Figure 10. The mean cross entropy error of the lesioned Highway networks trained on MNIST (left) and CIFAR 100 (right) as a function of removed layer (x-axis) is depicted.

(top row) and -4 for network trained on CIFAR 100 (bottom row) as reported by authors. However, there are slight differences between my heatmap plots in Figure 9 and their heatmap plots in Figure 2 due to the other hyperparameter settings and the number of training epochs. Most of the transform gate biases decreased in early layers and increased in successive layers in the network trained on CIFAR 100. One slight difference between the original heatmap and mine is that in the original paper the transform gate biases became more negative compared to my network's transform gate biases resulting in lighter blue blocks in earlier layers. However, concordant with the original paper, the transform gate biases at final layers increased in my analysis due to the reason as explained by the authors. The "information highways" in the network trained on MNIST dataset can be seen explicitly in the first and last columns of Figure 9, since the outputs were not transformed and remained unchanged, resulting in stripes in the plots. Again, the transform gate biases decreased less compared to the heatmap in the original paper resulting in a lighter blue color. I assume these slight differences originate from the use of different hyperparameter settings. However, the plots are very similar to the original one, and one can extract the main context from the plots.

I was curious and very interested in testing the highway networks by removing highway layers or making the transform gate outputs to 0 to examine the mean cross-entropy performance compared to the non-lesioned network. The highway layers at each depth were removed in both networks trained on MNIST and CIFAR 100. If the networks were equivalent to networks with far fewer layers, the loss could not decrease at no depth. However, the impact of layers can be noted easily from Figure 10 which refutes this hypothesis. For MNIST (left), if earlier layers are removed from the network, the error increases significantly, but layers around 15 through around 45 have small impact on the final performance. Most of the mid-layers behave as a skip connection and do not transform the input since the MNIST is an easy dataset which does not require complex representations compared to CIFAR100. However, for CIFAR100, training performance degrades undoubtedly if any of the first 40 layers are removed. Therefore, one should be cautious while lesioning the network since most of the highway layers have high influence on the overall network performance. As it can be seen, my results are very close to Figure 4, and the final network is not equivalent to a network with much fewer layers.

V. Conclusion

The idea of implementing a gating mechanism inspired by LSTM recurrent neural networks [10] enables training deep feedforward neural networks. The information passes through skip connections without attenuation even if 0 gradient is propagating backwards. Training plain networks suffer from degradation problem as the number of layers increases independent from the choice of non-linear activation function and the optimizer that were used. However, variance preserving weight initializations such as Glorot Uniform and He Normal [6,7] partially solve this problem, but the optimization process must be completed very carefully. The plain network using sigmoid activation function suffers from vanishing gradient problem where the gradient diminishes dramatically as it is propagated backward through the network. On the other hand, the plain network using ReLU activation function suffers dead ReLU problem where the mean and the variance of the activations become exactly 0 as the number of layer increases and the ReLU units can no longer be useful during training. However, if He Normal [7] weight initialization is applied, the variance does not converge to exactly 0 so one can train network up to 50 layers without highway layers. Highway networks can be trained without suffering any problems similar to those of the plain networks. However, if ReLU activation function is used, one should use gradient clipping or exponentially decaying learning rate to maintain the norm of the gradient at a certain level to prevent exploding gradient problem. Highway networks converge significantly faster compared to plain networks because of their unique architecture. Adding skip connections to the network and

initializing the transform gate biases to negative values to bias the network towards carry behavior could bring a question to mind: Is the highway network equivalent to a network with much fewer layers?
However, it was found that the highway networks use many of its layers as skip connections and removing one does not hurt the overall performance significantly for simple problems. Nevertheless, for complex problems, removing a highway layer hurts the overall performance noticeably.

My results and the results reported in the original paper [11] are very close but there are slight differences because of the different hyperparameter settings and number of training epochs. The website that the authors gave the details of the training procedure of highway networks cannot be reached at the moment, so I did a cross validation to find the best settings for the networks. I also implemented the convolutional highway network for my project. However, I did not add the results to this report because the authors did not disclose the architecture of convolutional highway networks. Since my results were very close to those reported in the original paper, I conclude that I fulfilled my goal.

VI. References

[1] https://github.com/ecbme4040/e4040-2021spring-project-HIGH-ky2446

[2] LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *nature*, *521*(7553), 436-444.

[3] Simonyan, K., & Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*.

[4] Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, *25*, 1097-1105.

[5] Bengio, Yoshua, Courville, Aaron, and Vincent, Pas- cal. Representation learning: A review and new perspectives. *Pattern Analysis and Machine In- telligence, IEEE Transactions on*, 35(8):1798–1828, 2013

[6] Glorot, Xavier and Bengio, Yoshua. Understanding the difficulty of training deep feedforward neural networks.

[7] He, Kaiming, Zhang, Xiangyu, Ren, Shaoqing, and Sun, Jian. Delving deep into rectifiers: Surpassing human-level performance on ImageNet

[8] Romero, A., Ballas, N., Kahou, S. E., Chassang, A., Gatta, C., & Bengio, Y. (2014). Fitnets: Hints for thin deep nets. *arXiv preprint arXiv:1412.6550*.

[9] Lee, C. Y., Xie, S., Gallagher, P., Zhang, Z., & Tu, Z. (2015, February). Deeply-supervised nets. In *Artificial intelligence and statistics* (pp. 562-570). PMLR.

[10] Gers, F. A., Schmidhuber, J., & Cummins, F. (1999). Learning to forget: Continual prediction with LSTM.

[11] Srivastava, R. K., Greff, K., & Schmidhuber, J. (2015). Highway networks. *arXiv preprint arXiv:1505.00387*.

[12] Srivastava, R. K., Greff, K., & Schmidhuber, J. (2015). Training very deep networks. *arXiv preprint arXiv:1507.06228*.