Highway Networks

E4040.2016Fall.DLGS.report

Giovanni Gambarotta gg2607, Mitsuhisa Hirose mh3464, Marek Sabata ms5112
Columbia University

Abstract

In this project, we explore the idea behind Highway Networks and various gating mechanisms used for training very deep neural networks. The motivation behind using deep networks is that there is plenty of evidence, both theoretical and empirical, that depth of neural network is one of the key factors in its success. Our goals are to replicate the results of the original paper and understand the key insignts the authors used. We also compare the results using gating mechanisms presented in other papers.
The biggest technical challenge we encountered is that highway networks appear to be very sensitive to hyperparameter settings, hence finding a good set of hyperparameters with limited computational capacity was very challenging.
The main result of our project basically confirm that Highway Networks architecture might be very well utilized to train very deep networks.

1 Introduction

Depth of neural network is a crucial factor of its success, as is empirically and theoretically justified, and deep networks usually perform significantly better than shallow networks (see [3], [4], [8] and [10] for references). However, training deeper networks introduces lot of difficulties (see [5]) compared to training shallow networks. Highway Networks are a novel architecture that should ease the optimization of very deep neural networks.
The main goal of our project is to implement the ideas in the original paper [2] and try to replicate the results the authors got. We also implemented different network architecture - so called ResNet - mentioned in paper [4], and at least partially compared the results between these two architectures. As the project is computationally very intensive, we tried to replicate the results only on MNIST and CIFAR-10 datasets, leaving out CIFAR-100. Namely, we have tried to replicate Figure 1 and Figure 2 on page 4 and 5 respectively of the Highway Networks paper [2] using the MNIST data set, Table 1 page 4 using the CIFAR-10 data set, and Table 6 in [4] also using the CIFAR-10 dataset.
While trying to replicate the results for highway networks, we needed to search heavily for hyperparameters that would give good results. This was by far the biggest difficulty and probably the main challenge of this paper. We have done extensive hyperparmeter search, training the highway network over 300 times for the 10 and 20 layer deep network with different hyperparameter settings. Unfortunately, at first even the best results were not really near to what the authors obtained. We have also tried to use the same hyperparameters as the authors used - which can be found in a database containing their code and results [11] - but even with those hyperparameters our results differed.
Hence we decided to take the hyperparameters that gave us most similar results and reproduce the results to the best possible extent. Having hard time finding good hyperparameters actually led us to interesting conclusion, which we will describe in greater detail in the rest of this project report.
Another challenge was training time of the algorithms. The training times for 50 and 100 layers highway networks, convolutional highway networks and almost all networks dealing with CIFAR-10, took very long to run. As there was no real way of speeding things up, we at first proceeded with smaller data set and then replicate the results on CIFAR-10 and MNIST only, leaving out CIFAR-100.

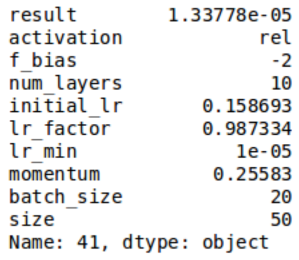
2 Summary of the Original Paper

The original paper - Highway Networks [2] - introduces new architecture of deep neural networks that allows for training very deep neural nets. The architecture is based on adding a special gating unit in each layer of the network, which allows unimpeded information flow across many layers, so called information higwhays.
The authors claim that their architecture might be used to train deep networks of virtually arbitrary depth by using the gating mechanism, which itself is inspired by Long Short Term Memory recurrent neural networks [6]. Their claims are supported by experimenting with deep plain and highway networks and comparing the results. It shows that highway networks managed to outperform plain networks in terms of minimizing cross entropy over 400 epochs for depths of 10, 20, 50 and 100. The authors also compare Highway Networks to Fitnet results, which are presented in [7].
In summary, highway networks utilize the gating mechanism to pass part of the information almost unchanged through many layers, which allows not just easier training but also routes the information in the trained network. Detailed explanation of the ideas is given in the following subsection.

2.1 Methodology of the Original Paper

In this section, we will talk in greater detail about the approach authors of the paper took to obtain their results. This

coincides with our approach too, as our main goal is to replicate the results of the paper.
The authors claim that very deep networks are difficult to optimize even if we use the variance-preserving initialization scheme introduced in [4]. To demonstrate that highway networks do not have this difficulty, they decided to run series of experiments on the MNIST data set and measure crossentropy on the training set. They trained plain and highway networks with the same architecture with depth $L = 10, 20, 50$ and 100. The first layer is always regular fully connected layer and the following $L-1$ layers are highway layers fed to a softmax classifier. The number of hidden units was set to be 50 for the highway network and in order to roughly match the number of parameters in the plain network, the number of hidden neurons in the plain network was set to 71.
The orinal paper also explores how transform gate biases changes in each layer during training. It's an interesting visualisation showing us roughly where the information is simply passed through and when the hidden layer activation plays larger role.
In order to train the networks, we need to specify the hyperparameters. In the setting of this paper, these are: initial learning rate, momentum, learning rate decay, activation function of the H function in hidden layer (which was constrained to either *ReLU* or *tanh*) and the value of initial transformation bias $\mathbf{b}_T$. All other weights were initialized according to the paper [4].
The authors implemented their optimization algorithms in Caffe. They claim to have done 40 runs to find a good hyperparameters for the networks. However, we are a bit sceptical about this part, as we've made over 300 runs and none of it got even close to their results. When we tried out the exact parameters they've used, we were far from the results they claim to achieve. Even intuitively 40 runs seem like a small number as there are 5 hyperparameters, meaning that even if we try just 2 values for each of them, we will need $2^5 = 32$ runs. Also, when we found out online which hyperparameters the authors used, it looks quite improbable that any hyperparameter search approach would yield their hyperparameters after 40 just runs. The following table shows hyperparameters used by the authors of the paper.



```
result         1.33778e-05
activation             rel
f_bias                  -2
num_layers              10
initial_lr        0.158693
lr_factor         0.987334
lr_min               1e-05
momentum           0.25583
batch_size              20
size                    50
Name: 41, dtype: object
```

Figure 1. Hyperparameters used by the authors of the Highway Networks paper.

Nevertheless, after doing our own hyperparameter search, we have decided to use similar parameters as the authors used.

2.2 Network Architecture

We'll briefly describe the architecture of the highway network from a mathematical standpoint. Detailed computational graph can be found in section 4.
Suppose we have a network with L layers and in each layer l we apply nonlinear transform function $H = H(\mathbf{W_H}, l)$. The output can then be written as

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

In case of highway networks, we define so called transform gate $T(\mathbf{x}, \mathbf{W_T})$ and carry gate $C(\mathbf{x}, \mathbf{W_C})$. The highway layer is then constructed as follows:

$$\mathbf{y} = H(\mathbf{x}, \mathbf{W_H}) \cdot T(\mathbf{x}, \mathbf{W_T}) + \mathbf{x} \cdot C(\mathbf{x}, \mathbf{W_C})$$

For simplicity the authors set $T = 1 - C$. In this manner, we can see that if T gets close to one, we are sending the original vector almost unchanged and if T gets close to zero, the information flow from $\mathbf{x}$ gets diminished. Specifically we have

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

and as for the gradient descent, we can easily see that the Jacobian of $\mathbf{y}$ with respect to $\mathbf{x}$ is

$$\frac{\partial \mathbf{y}}{\partial \mathbf{x}} = \begin{cases} \mathbf{I}, & \text{if } T(\mathbf{x}, \mathbf{W_T}) = \mathbf{0} \\ H'(\mathbf{x}, \mathbf{W_H}) & \text{if } T(\mathbf{x}, \mathbf{W_T}) = \mathbf{1} \end{cases}$$

It is now not hard to see the main idea behind highway networks, and how is the information flow going to change with changing the gating unit T. Note that dimensionality of $\mathbf{x}$ and $\mathbf{y}$ needs to be the same.
When it comes to the training of the network, the authors state that they've used simple stochastic gradient descent with momentum and decaying learning rate.

2.3 Key Results of the Original Paper

The key result of the paper is demonstrating that using the gating function of highway networks, we are able to train very deep networks without any optimization problems. The gating mechanism is able to pass information through many layers, which allows not only for easier training, but also for information routing to the trained network. The paper showed that negative bias initialization in the T gating function was sufficient for learning very deep networks with various zero mean distributions of $\mathbf{W}_H$ and different activation functions H. Moreover, the bias often got even more negative during the training, meaning we use more information from the original vector.

The paper also claims that while plain networks' performance significantly degrades as their depth increases, highway networks do not suffer from increasing depth at all. The authors also pointed out that the highway networks always converge significantly faster than the plain ones.

3 Methodology

In this section we will comment on how we approached replication of the paper's results. As the basis of our methodology is the same as in the paper [2], and we described their methodology and network in detail in the previous section, we will comment mostly how our methodology differes from theirs. We will first present the objectives of our project and then move on the the specifics of our implementation.
As our objectives can be split into four main tasks, we will first give a brief introduction of each one of them.

3.1 Objectives, Technical Challenges, Problem Formulation and Design

Our objectives can be split into four main parts. These are:
1) Replicate the plots in Figure 1 using the MNIST dataset in paper [2],
2) Replicate the heatmap in Figure 2 using the MNIST dataset in paper [2],
3) Replicate results of Table 1 using the CIFAR-10 dataset in [2] by implementing 11 and 19 layer convolutional Highway Networks,
and finally
4) Replicate Table 6 in [4] using the CIFAR-10 dataset by implementing 20,32 and 44 layer deep Residual Network and compare the results to the results obtained in paper [2].
We'll now describe the tasks in greater detail.
1) and **2)**: In order to reproduce the plots, we need to measure the mean crossentropy error on the training set for the first 400 epochs of the training algorithm of plain and highway network using the best hyperparameter settings. This is done for network of depths $L = 10, 20, 50$ and 100. To get the best possible hyperparameter settings, we have done extensive hyperparameter search (roughly 3 days of computation) for the highway networks, but our results were still more than order of magnitude worse than they obtained in the paper. As stated before, the hyperparameters to determine were the learning rate, learning rate decay, highway layer activation function, momentum and gate bias. We restricted the number of epochs to be just 50 for the hyperparameter search, as suggested by the TA. We were mainly focused on obtaining similar performance for the highway networks and not so much for the plain networks, as we could always use the plain networks benchmark they used in the paper. This fully showed when we were recreating Figure 1 for the first task, as our plain networks for depth 50 and 100 performed extremely poorly with the hyperparameters we specified.
At first the optimization algorithm we used was RMSprop, because it is supposed to be one of the most powerful optimizers and we expected to get better results. This was, however, not the case and our results for basically any set of hyperparameters was off by roughly one magnitude. Hence we decided to switch to Momentum with exponential decay as the authors claim they did in the paper. We also used smaller data set this time, because retraining everything on the full data set would be too time consuming and the TA's suggested we might reduce the data set.
We have also searched for publicly available code [11] and were able to find out what hyperparameters authors of the paper used (we showed them in Figure 1) and then searched for hyperparameters in similar range. In the end we decided to use the best set of hyperparameters from all of our searches to proceed with the results replication and run the models for full 400 epochs.
Most of the challenges were already mentioned in the previous paragraphs, but to summarize them, it was definitely the training time, hyperparameter search and not enough guidance on how exactly the authors implemented their ideas. For example, we didn't know the exact weight initialization.
As we've already mentioned, the previous two problems used the MNIST data set. The following two tasks are concerned with the CIFAR-10 dataset.
3) As mentioned above, we have also tried to reproduce Table 1 in [2], specifically the models called Highway 1-4. In the paper, they claim that Highway 1 and Highway 2 are based on the architecture of Fitnet 1 and Fitnet 4 developed by Romero et al. [7], and Highway 3 and 4 are said to be thinner and deeper networks, although no further explanation is provided. Since there is no guidance on how to replicate Highway 3 and 4, we decided to implement Highway 1 and 2 and see how much testing error we can achieve with these two models. We then compare the testing error we achieved with the one achieved in the original paper [2] and also the errors achieved in paper [4].
As it was not exactly clear from the paper how to integrate Highway Networks with Fitnets, it was still a challenge for us to figure out the exact architecture we should use. We refered to the authors' Caffe code posted on Github [11] for Highway 2. According to the code, Highway 2 with 19 layers is composed of 3x3 convolutions with 32, 80, and 128 channels followed by a 1x1 convolution with 100 channels. It also looks like they've utilized dropout layers between different channels and average global pooling at the end of the network. Unfortunately, we were unable to find any code regarding Highway 1, so we could only estimate it given the architectures of Fitnet 1 and Highway 2. In our final settings, Highway 1 and 2 have 222K and 2.25M parameters, respectively. That's fewer than the paper specified, so it's possible our model will have worse performance than the one achieved in the paper. Detailed tables regarding number of parameters can be found in Appendix.
Regarding the optimization algorithm, the authors used SGD with Nesterov momentum and multi-step decay. We tried to proceed with the exact same algorithm, but noticed that Nesterov momentum is significantly slower than the standard momentum due to theano.clone function needed to implement Nesterov momentum. Given our time constraints, we decided to proceed with standard momentum combined with multi-step decay.

4) The last set of models we implemented were residual networks, which we used to try to replicate Table 6 in paper [4]. We followed the architectures outlined in the paper which has a stack of 6*n* layers with 3x3 convolutions, which yielded exactly the same number of parameters as they claim the network should have in the paper. Again, detailed tables regarding number of parameters can be found in the Appendix.
The only difference we made is that we introduced pooling and dropout layers instead of convolutions with stride 2 for downsampling and weight decay with batch normalization for reguralization. Since our goal is to compare the performance of highway and residual networks, we believe our approach of having similar architectures to Highway 1 and 2 is very reasonable.
For the same reason, our optimization algorithm is exactly the same as the one we used for Highway Networks, that is momentum with multistep decay. It is also worth noting that using pooling layers, there is no mismatch in image sizes for computing the

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

part (which is how the network works), although we still needed to add extra zero entries whenever the number of channels increased.
Also, for part 3) and 4) data augmentation was applied both for highway and residual networks in the original papers. Since they did not provide much detail, we decided to use a combination of standard transformation techniques: translation, random flipping, and rotation. In order to maximize the generalization capability, we augmented the input data at the start of each epoch so that we never see exactly the same training set twice.
In the following section, we will present graphical details regarding the networks.

4 Implementation

In this section, we will present the computational graph for one layer of highway network and the architectures we implemented for convolutional and residual networks for CIFAR-10. Most of the details of actual implementation for each of the subproblem were provided in the previous section, so we'll use this section to demonstrate the implementation graphically.

4.1 Deep Learning Network

As we've described how Highway Networks work from the mathematical standpoint in the second part, and presented our methodology in the previous part, we will just show detailed computational graph of one hidden layer for the Highway Network here.
The following figure is relevant to the first and second subproblems we were working on. It shows the computational graph of one layer in a highway network.

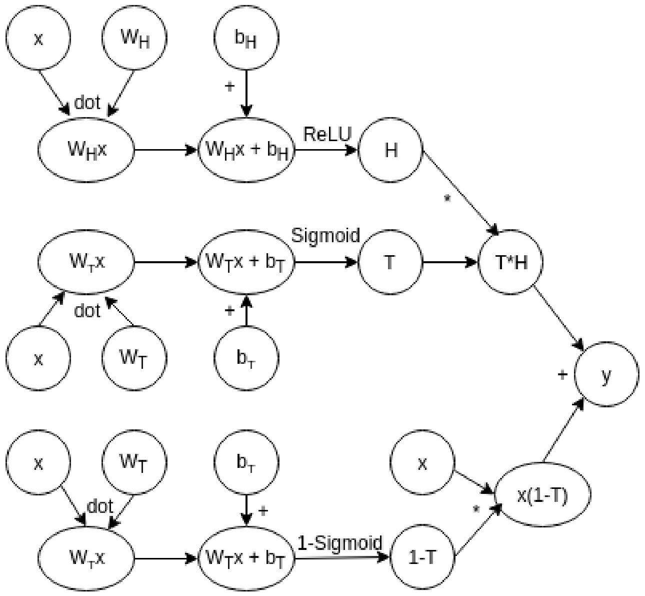


Figure 2. Computational graph of highway layer.

For the third and fourth subproblem, we mentioned that we made some modifications to the Residual network presented in [4] and the Highway 1 and 2 networks. The detailed architecture is presented in the following flowchart.

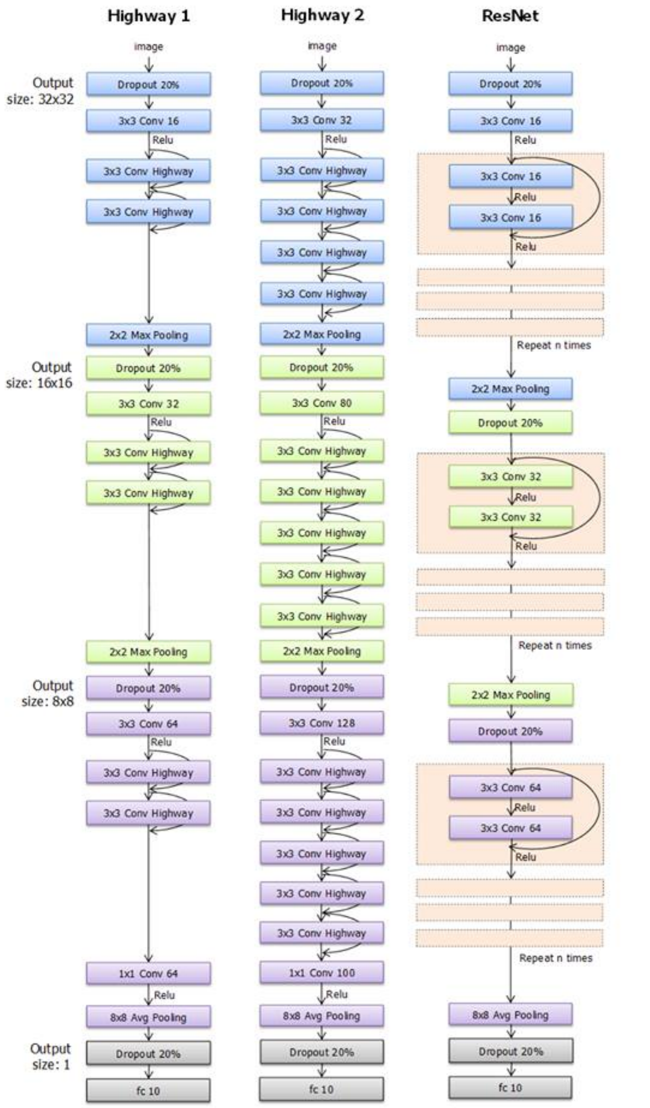


Figure 3. The curved arrows represent highway routes for highway networks or shortcuts for ResNets.
Left: Highway 1 with 11 layers. **Middle:** Highway 2 with 19 layers. **Right:** ResNet with $6n+2$ layers where $n = \{3,5,7\}$.

The architecture of highway networks really doesn't differ much from plain networks. As we've covered plain networks heavily during the course, we don't feel the need to provide pseudocode of how they work. On the other hand, implementation of the above networks is relatively complicated and we don't see any good way of briefly summarizing it in a pseudocode. We believe that the code we attached to this project is highly readable and can be used as a reference.
All the training algorithms we used were covered in the lecture, so again, we feel like it would be redundant to cover it here again.

4.2 Software and Hardware

As the algorithm implementation details were provided in the previous sections, we will just comment on the languages and computational capacities we've used. All the codes in our project were produced in Python 2 with the deep learning training and optimization carried out in Theano. The parts of our code meant for presentation were executed in Jupyter Notebook, tables created in Pandas and plots with Matplotlib.
The codes were executed on our personal laptops, one disponing with with Nvidia GeForce GTX 960M GPU graphics card, and the other team members used Amazon's web services' Tesla K80 GPU's.
The code should be accessible in *src* folder and well documented for anyone's read.

5 Results

In this section, we will first comment on the results we've managed to achieve for all of our subproblems and subsequently, we'll compare them to the results achieved in papers [2] and [4]. We also discuss how and why our results differ, what could be the main cause of the difference and how could we possibly make the results better.

5.1 Project Results and Comparisons

We'll describe our results for all four subproblems we've attempted to solve.
1) We will start with the first subproblem, that is with replication of Figure 1 in [2]. The goal was to obtain very similar values of mean cross entropy on the training set for 10, 20, 50 and 100 layers deep highway networks as are presented in the paper.

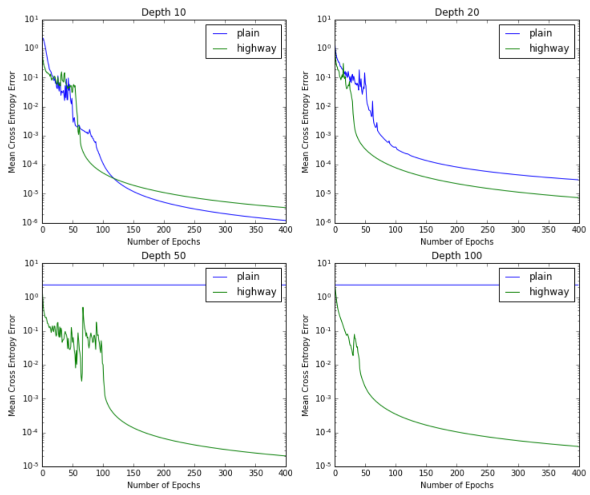


Figure 4. Comparison of optimization of plain networks and highway networks of various depths.

We can see from the above figure, that we managed to reproduce the results for Highway Networks quite successfully.

For comparison, we show the results of the paper in a figure at the end of this paragraph. We achieve roughly the same magnitude of the average crossentropy as is achieved in the paper, though it looks like our networks' convergence is slower. It is also the case that for 10 layer networks, the plain network outpefromed the highway network, exactly as is the case in the paper.

What is now more than apparent is that our hyperparameters for the deep plain networks are performing very poorly, as the 50 and 100 layer plain networks were unable to converge anywhere in 400 epochs. However, we believe this is a great demonstration of the power of highway nets in practice - plain networks didn't manage to converge at all while Highway networks were still converging to lower average crossentropy throughout all 400 epochs.

In order to achieve better results, we would need to resolve all the issues mentioned in the third section. That is having better hyperparameters, knowing exactly the weight initialization, and possibly knowing the exact implementation of the momentum optimization algorithm the authors used. Then our networks would perform on par with the networks in the paper and we believe it would track the results very closely. Obviously, if we had more time, we would do a proper hyperparameter search for the plain network, so that it works at least partially for the deep 50 and 100 layer networks.

Below is a figure displaying the original results we aimed to replicate.

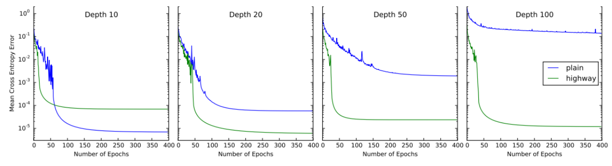


Figure 1. Comparison of optimization of plain networks and highway networks of various depths. All networks were optimized using SGD with momentum. The curves shown are for the best hyperparameter settings obtained for each configuration using a random search. Plain networks become much harder to optimize with increasing depth, while highway networks with up to 100 layers can still be optimized well.

Figure 5. Results of the original paper.

2) The second problem asked us to replicate the heat map. We managed to do so for the MNIST data set. Unfortunately, as hyperparameter search and training for the CIFAR-100 dataset was extremely time demanding, we didn't have time to produce any good results for it.

In the figure below, we present the results we obtained by tracking value of the transformation bias in the 50 layer highway network. For the heatmap presented in the paper, please see figure at the end of this paragraph.

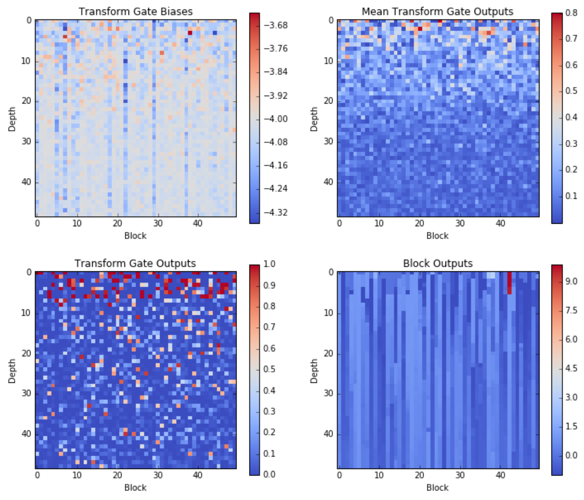


Figure 6. Visialization of certain internals of the blocks in the best 50 hidden layer highway network on MNIST.

We belive that we've managed to reproduce the patterns quite well. On absolute scale, our results differ as we've used different transform gate biases and different activation function, but the patterns match. Namely, you can notice that the transform gate outputs and their means behave almost exactly the same as in the paper. The transform gate biases have quite similar pattern for the first roughly 10 layers and later on they appear to be more consistent in our case, whether in the paper they change significantly. The last part of the plot - block outputs - also replicates the patterns very well. Overall the patterns we managed to obtain replicate the patterns of the original paper in a solid manner.

In order to achieve even better replication, the first step would be to find out why different activation function performed better for us than the one authors used and then replicate the results with the same activation function. It would also be good to know the exact initialization of the weights (not just the transformation gate biases) to get closer replication of the heatmap.

Below, you can see the transformation gate biases from the original paper.

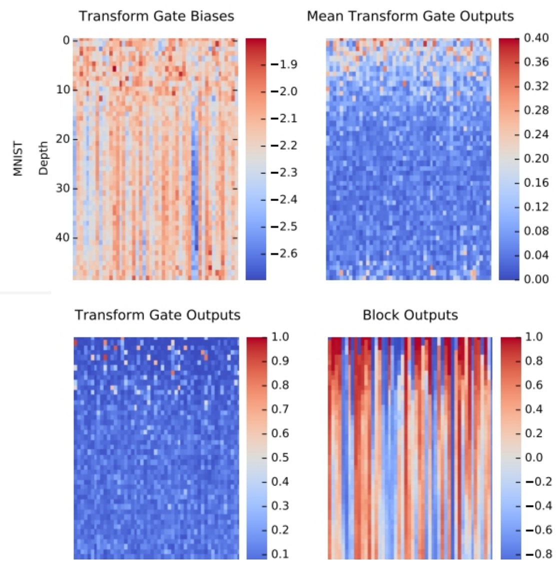
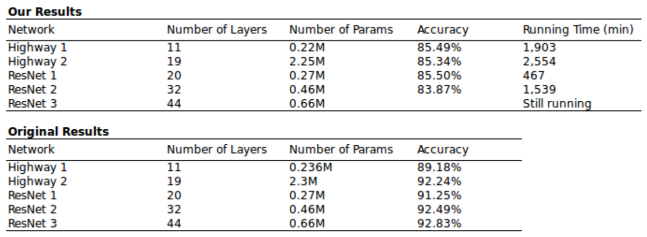


Figure 7. Results of the original paper.

3) and **4)**. Let's move on to replication of Table 1 in [2] and Table 6 in [4]. As we managed to only replicate few of the results, we merged them into one table. The summary of our results and the results obtained in both of the papers is in the following table.



Our Results

Network	Number of Layers	Number of Params	Accuracy	Running Time (min)
Highway 1	11	0.22M	85.49%	1,903
Highway 2	19	2.25M	85.34%	2,554
ResNet 1	20	0.27M	85.50%	467
ResNet 2	32	0.46M	83.87%	1,539
ResNet 3	44	0.66M		Still running

Original Results

Network	Number of Layers	Number of Params	Accuracy
Highway 1	11	0.236M	89.18%
Highway 2	19	2.3M	92.24%
ResNet 1	20	0.27M	91.25%
ResNet 2	32	0.46M	92.49%
ResNet 3	44	0.66M	92.83%

Figure 8. Generalization result from Highway 1 and 2, ResNet 1 and 2 compared with the results of the original papers.

Note that Resnet1 was trained on more powerful GPU, that's why it took 'only' 8 hours and not more than a day as the other models.
We can see that we've managed to achieve roughly 5% worse accuracy in all cases (Higwhay 1 and 2, ResNet 1 and 2) than the papers achieved. However, as this was computationally very expensive, we couldn't do proper hyperparameter search, including generalization and data augmentation. Moreover, there were some parts of the overall architecture that we weren't able to determine from what the authors provide, so we were forced to make certain changes (detailed explanation in part 3 of this project). In order to improve our results, it would be a good start to do a proper hyperparameter search and then to reach out to the authors of the paper to ask about the pertinent details we were unable to understand from the paper or the publicly provided pieces of code.

Details of all our computations might be found in log files we submitted together with the report.

5.2 Comparison of Results

Overall we have managed to obtain similar results as are presented in the paper. Even though our results differ, the main trends and underlying messages of our results are the same as in the paper for all tasks. Namely, we also demonstrated that training very deep highway networks can be done without difficulties and that highway networks usually converge faster than plain networks. For the smaller models that were possible to train on our own GPU on in a reasonable time on Amazon server, we obtained quite good accuracy that was just a few percent below the papers' state of art accuracy. We consider this a success as our computational and time capabilities were limited.
Even though we gave specific details why our resulta might differ while commenting on each subproblem, we will describe the differences in general in the next subsection.

5.3 Why the results differ?

We believe the root cause of the differences between our results and the results in the paper could be the following:
1) Different initialization: As none of the papers fully revealed how exactly they initialized weights, we belive this could be one source of errors.
2) Slightly different network set up: We have already said that some parts of the network in the papers weren't described fully, so we needed to make small changes. This probably caused some differences.
3) Different implementation: What might also be happening is that as we are using Theano and the papers were implemented in Caffe, the actual implementations may be different, which might introduce another source of error.
4) Hyperparameters: We've described this problem quite extensively before, so we include it here just to make the exposition complete. We belive that using different hyperparamaters was one of the main sources of errors in our results.
5) Data augumentaion: In the third and fourth subproblem, the authors do not state exactly what data augmentation they've used. We were forced to fill in the blanks here, and this might have introduced additional errors.

Then there are couple of things we've intentionally did differently, as it yielded better results for us, or was simply too hard to implement in Theano.
6) Different bias initialization and activation function: For the second subproblem, we have used different transformation gate bias initialization and activation function, as we were getting better results with it. This, of course, shifted the results to different interval and possibly skewed them.
7) For the third and fourth subproblem, authors of the paper state that they used weight decay. this might be crucial for large network such as Highway 2. Investigating the log data, we found that the cost function of Highway 2 became approximately 100 times smaller than the other models, but

it did not achieve a higher testing accuracy. It is clear that we should have imposed more regularizations for Highway 2.
8) Nesterov: We have not used Nesterov in the momentum, as we've already discussed that it caused much higher running time of the algorithm. This might be another reason why our results differ.
9) Experimenting with normalization: At last, we didn't really have much time to experiment e.g. with data normalization, weight normalization or layer normalization. Using these would yield different, potentially better results.
As you can see, there are many factors that could cause different behavior of our network and therefore result in different behavior of our network. Most probably in each instance we've obtained worse results because of combination of few of the factors mentioned above. Looking at the above list should remind us of just how many factors there are that play a significant role in setting up and training neural networks and it should be clear that every proposed idea of how to train neural networks will yield many different implementations.

5.4 Discussion of Insights Gained

In this subsection we'll summarize the insights we gained while working on this project. As we all agreed on the insights, we will present them as a team. We believe that the biggest insight of Highway Networks is that they might be beneficial for training very deep networks in supervised setting. It also works well even when we are interested in generalization performance on the test set.
One insights we've discovered while searching for the hyperparameters, which we all believe is quite unfortunate, is that it appears that performance of highway networks is highly dependent on the hyperparameters, weight initialization, and even the optimization algorithm. This was demonstrated by our extensive search for the hyperparameters, using different optimizers, and weights, and still not getting as good results as are presented in the paper. Moreover, even when we used the exact hyperparameters as the authors of the paper used, the same optimizier and the only thing that differed was weight initialization (as we couldn't know their exact initialization scheme), our results weren't as good as the ones in the paper.
We believe that this might be a major drawback of highway networks - even though they make it possible to train very deep networks, the hyperparameter search might be a huge bottleneck for rapid implementation to obtain good results on practical problems.

6 Conclusion

In this project, we attempted to reproduce results of the Highway Networks paper [2] and at least some results of the Deep Residual Learning for Image Recognition [4] paper. There were four main tasks: 1) replicate figures comparing optimization of deep highway networks to deep plain networks, 2) replicate heatmap demonstrating behavior of transformation gate biases in 50 layer deep highway network, 3) implement highway convolutional networks and compare their accuracy with the one achieved in paper [2], and 4) implement ResNet convolutional network and compare its accuracy in paper [4].
With some modifications, we have successfully managed to replicate most of the results, though not to the full extent of the papers.
We believe that further research on understanding Highway Networks, particularly how to make them more robust to hyperparameter settings, could be very beneficial to the deep learning community and might make Highway Networks widespread and very popular method of tackling training of deep networks.

6.1 Team members contribution to the project

All member of our team worked more or less on each part of the project, so we can't really differentiate who did most work on what. We worked extremely well together, implementing most codes together, splitting all experiments equally, running times roughly equally, and parts of the write up equally. Hence we can say with clear consience that we all contributed 33.33% to the project.

7 Acknowledgments

We want to thank Professor Zoran Kostic for providing a great introduction to neural networks and deep learning. We would also like to express our gratitude to all the TA's and CA's of ECBM4040 for their constant support and guidance on our assignments and project, and the insights they provided. We wouldn't be able to learn what we have without you and your support in the course.

8 References

List of references:
[1] Bitbucket link: `https://bitbucket.org/e_4040_ta/e4040_project_dlgs`
[2] R.K. Srivastava, K. Greff, J. Schmidhuber, "Highway Networks", 3 November 2015, ICML 2015 Deep Learning workshop, URL: `https://arxiv.org/abs/1505.00387`
[3] R.K. Srivastava, K. Greff, J. Schmidhuber, "Training Very Deep Networks", 22 July 2015, Advances in Neural Information Processing Systems 2015, URL: `https://arxiv.org/abs/1507.06228`
[4] K. He, X. Zhang, S. Ren, J. Sun, "Deep Residual Learning for Image Recognition", 10 December 2015, Tech report
[5] X. Glorot, Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks", May 2010, Aistats, vol. 9, pages 249-256, URL: `http://www.jmlr.org/proceedings/papers/v9/glorot10a/glorot10a.pdf?hc_location=ufi`
[6] S. Hochreiter, J. Schmidhuber, "Long short term memory", technical report FKI-207-95, Technische Universitat

Munchen, August 1995

[7] A. Romero, N. Ballas, S.E. Kahou, A. Chassang, C. Gatta, Y. Bengio, ”FitNets: Hints for thin deep nets”, December 2014, URL: `http://arxiv.org/abs/1312/6120`

[8] K. Simonyan, A. Zisserman, ”Very deep convolutional networks for large-scale image recognition”, September 2014, URL: `http://arxiv.org/abs/1409.1556`

[9] Y. Bengio, P. Simard, P. Frasconi, ”Learning long-term dependencies with gradient descent is difficult”, IEEE Transactions on Neural Networks, 5(2): 157-166, 1994

[10] A. Krizhevsky, I. Sutskever, G. Hinton, ”Imagenet classification with deep convolutional neural networks”, NIPS 2012, URL: `https://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf`

[11] Github of authors of paper [2]: `https://github.com/flukeskywalker/highway-networks/`

9 Appendix

List of figures:

ResNet
n=3 (20 Layers)

Img Size	Type	Size	# Params Coefficient	Bias
32x32	Convolution	3x3x16	432	16
	Convolution	3x3x16	2,304	16
	Convolution	3x3x16	2,304	16
	Convolution	3x3x16	2,304	16
	Convolution	3x3x16	2,304	16
	Convolution	3x3x16	2,304	16
	Convolution	3x3x16	2,304	16
16x16	Convolution	3x3x32	4,608	32
	Convolution	3x3x32	9,216	32
	Convolution	3x3x32	9,216	32
	Convolution	3x3x32	9,216	32
	Convolution	3x3x32	9,216	32
	Convolution	3x3x32	9,216	32
8x8	Convolution	3x3x64	18,432	64
	Convolution	3x3x64	36,864	64
	Convolution	3x3x64	36,864	64
	Convolution	3x3x64	36,864	64
	Convolution	3x3x64	36,864	64
	Convolution	3x3x64	36,864	64
1x1	FC	10	640	10
			Total:	269,034

ResNet
n=5 (32 Layers)

Img Size	Type	Size	# Params Coefficient	Bias
32x32	Convolution	3x3x16	432	16
	Convolution	3x3x16	2,304	16
	Convolution	3x3x16	2,304	16
	Convolution	3x3x16	2,304	16
	Convolution	3x3x16	2,304	16
	Convolution	3x3x16	2,304	16
	Convolution	3x3x16	2,304	16
	Convolution	3x3x16	2,304	16
	Convolution	3x3x16	2,304	16
	Convolution	3x3x16	2,304	16
	Convolution	3x3x16	2,304	16
16x16	Convolution	3x3x32	4,608	32
	Convolution	3x3x32	9,216	32
	Convolution	3x3x32	9,216	32
	Convolution	3x3x32	9,216	32
	Convolution	3x3x32	9,216	32
	Convolution	3x3x32	9,216	32
	Convolution	3x3x32	9,216	32
	Convolution	3x3x32	9,216	32
	Convolution	3x3x32	9,216	32
	Convolution	3x3x32	9,216	32
8x8	Convolution	3x3x64	18,432	64
	Convolution	3x3x64	36,864	64
	Convolution	3x3x64	36,864	64
	Convolution	3x3x64	36,864	64
	Convolution	3x3x64	36,864	64
	Convolution	3x3x64	36,864	64
	Convolution	3x3x64	36,864	64
	Convolution	3x3x64	36,864	64
	Convolution	3x3x64	36,864	64
	Convolution	3x3x64	36,864	64
1x1	FC	10	640	10
			Total:	463,018

Highway 1
(11 Layers)

			# Params for H(x)		# Params for T(x)	
Img Size	Type	Size	Coefficient	Bias	Coefficient	Bias
32x32	Convolution	3x3x16	432	16		
	Conv Highway	3x3x16	2,304	16	2,304	16
	Conv Highway	3x3x16	2,304	16	2,304	16
16x16	Convolution	3x3x32	4,608	32		
	Conv Highway	3x3x32	9,216	32	9,216	32
	Conv Highway	3x3x32	9,216	32	9,216	32
8x8	Convolution	3x3x64	18,432	64		
	Conv Highway	3x3x64	36,864	64	36,864	64
	Conv Highway	3x3x64	36,864	64	36,864	64
	Convolution	1x1x64	4,096	64		
1x1	FC	10	640	10		
					Total:	222,378

Highway 2
(19 Layers)

			# Params for H(x)		# Params for T(x)	
Img Size	Type	Size	Coefficient	Bias	Coefficient	Bias
32x32	Convolution	3x3x32	864	32		
	Conv Highway	3x3x32	9,216	32	9,216	32
	Conv Highway	3x3x32	9,216	32	9,216	32
	Conv Highway	3x3x32	9,216	32	9,216	32
	Conv Highway	3x3x32	9,216	32	9,216	32
16x16	Convolution	3x3x80	23,040	80		
	Conv Highway	3x3x80	57,600	80	57,600	80
	Conv Highway	3x3x80	57,600	80	57,600	80
	Conv Highway	3x3x80	57,600	80	57,600	80
	Conv Highway	3x3x80	57,600	80	57,600	80
	Conv Highway	3x3x80	57,600	80	57,600	80
8x8	Convolution	3x3x128	92,160	128		
	Conv Highway	3x3x128	147,456	128	147,456	128
	Conv Highway	3x3x128	147,456	128	147,456	128
	Conv Highway	3x3x128	147,456	128	147,456	128
	Conv Highway	3x3x128	147,456	128	147,456	128
	Conv Highway	3x3x128	147,456	128	147,456	128
	Convolution	1x1x100	12,800	100		
1x1	FC	10	1,000	10		
					Total:	2,256,838