University of Peradeniya
Department of Computer Engineering

CO542 Neural Networks and Fuzzy Systems

Neural networks Reading Group 2021

03. Training very deep networks

Group 02:

E/16/022	Chamath Amarasinghe
E/16/025	Diwanga Amasith
E/16/222	Wishva Madushanka
E/16/232	Shamra Marzook

September 29, 2021

Contents

List of Figures

List of Tables

Paper title	Training Very Deep Networks
Paper year	2015
Paper authors	Rupesh Kumar Srivastava, Klaus Greff, Jurgen Schmidhuber

Table 1: Paper information

1 Introduction

It is extremely difficult to train a really deep neural network. It is discovered that optimizing a highly deep neural network is difficult. This is most likely due to a gradient disappearing issue. Many efforts have been made to address this issue. To solve the challenges of training very deep networks, a novel architecture design has been introduced.

2 Questions and Answers

2.1 What are the difficulties of very deep neural networks?

Deep neural network can be simply identified as a feedforward network with many hidden layers. Though deep neural networks are super-efficient for many tasks, there are some limitations regarding them. Commonly, the requirement to have a very large amount of data becomes a major difficulty when performing better than other techniques. On the other hand, it is extremely expensive to train due to complex data models used in deep neural networks. At the same time, there is no standard theory to guide in selecting the right deep learning tools as it requires knowledge of topology, training method and other parameters.

Answered by: Amarasinghe D.L.C. (E/16/022)

2.2 Explain Highway Networks vs Residual Networks with their pros and cons.

Residual networks can be considered as a special case of highway networks. To put it simply, both methods utilize the idea that if activation y is equal to input x, then dydx = 1 or if cx is an additive term of y then |dydx| $\gtrapprox$ |c|. This addresses the vanishing gradients problem in deep networks where back-propagated gradient may go toward 0 when local gradients are less than 1.

The following equation in “Highway networks” defines the non-linear transformation at each layer as:

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

Every layer output is a weighted combination of input x with carry weight C and some nonlinear transformation $H(x, W_H)$ with transform weight T.

Some major differences in highway networks vs residual networks are,

- Number of parameters:
  Residual networks don't have the extra parameters compared with highway networks.
- Architecture:
  Residual networks try to create the ideal mapping by using X (design matrix) directly while highway networks use a 'gate' mechanism.
- Performance:
  Residual networks basically swept all the competitions while the performance of highway networks need more experiments.

Residual networks are basically the best for classification as of now.

Answered by: Amarasinghe D.L.C. (E/16/022)

2.3 Can we use memory cells to avoid vanishing gradient problem in LSTM?

Yes, memory cells can be used to avoid vanishing gradient problem in LSTM. LSTM was invented specifically to address this issue. Memory cells store previous instances. It is supposed to do that with the Constant Error Carousel (CEC), which on the diagram below corresponds to the loop around the cell.

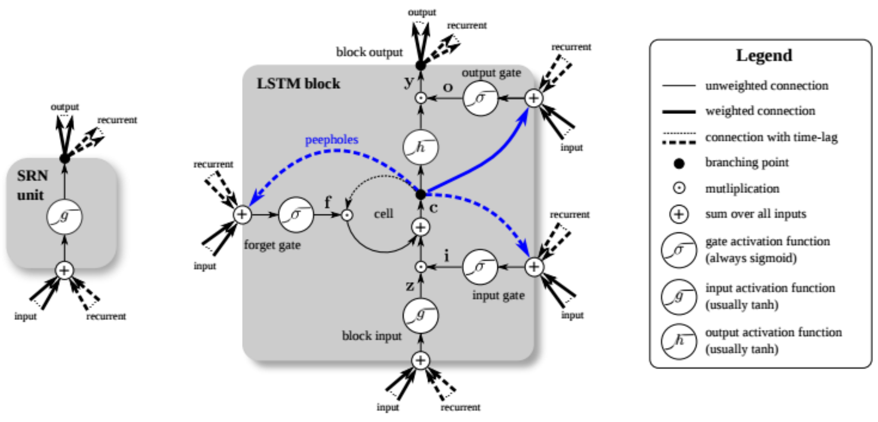

Figure 1. Detailed schematic of the Simple Recurrent Network (SRN) unit (left) and a Long Short-Term Memory block (right) as used in the hidden layers of a recurrent neural network.

Figure 1: Detailed schematic of Simple Recurrent Network (SRN) unit and a Long Short-Term Memory block

A schematic of the vanilla LSTM block can be seen in Figure 1. It features three gates (input, forget, output), block input, a single cell (the Constant Error Carousel), an output activation function, and peephole connections. The output of the block is recurrently connected back to the block input and all of the gates.

Answered by: Amarasinghe D.L.C. (E/16/022)

2.4 Why the accuracy of MNIST dataset having no vast change with the highway network?

MNIST is a set of small images of handwritten digits. There are 70,000 images and each image has 784 features. Training set is of 60,000 examples and the test set is of 10,000 examples. Highway networks are designed for training of very deep networks with increasing depth.

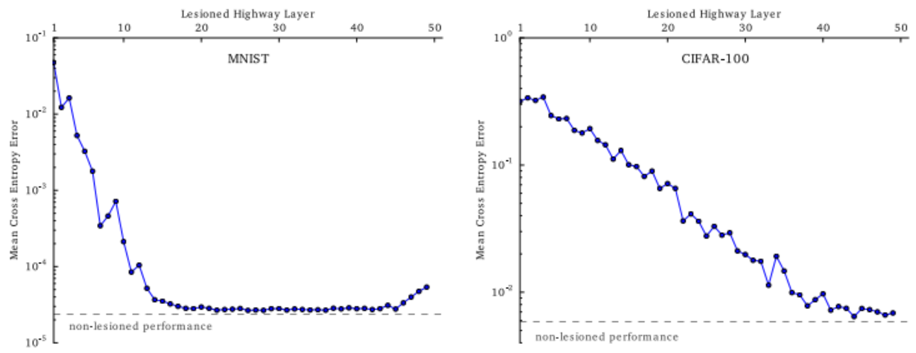

Figure 2: Lesioned training set performance (y-axis) of the best 50-layer highway networks on MNIST (left) and CIFAR-100 (right), as a function of the lesioned layer (x-axis)

As in the above figure, for MNIST, it can be seen that the error rises significantly if any one of the early layers is removed, but layers $15 - 45$ seem to have close to no effect on the final performance. About 60% of the layers don't learn to contribute to the final result, likely because MNIST is a simple dataset that doesn't require much depth.

For the CIFAR-100 dataset with performance degrading noticeably when removing any of the first ≈ 40 layers. This suggests that for complex problems a highway network can learn to utilize all of its layers, while for simpler problems like MNIST, it will keep many of the unneeded layers idle. Therefore, there won't be any significant change in accuracy in MNIST dataset when highway networks are used.

Answered by: Marzook F.S. (E/16/232)

2.5 What is the difference between Batch Gradient Descent and Stochastic Gradient Descent?

Gradient Descent is one of the most important concepts used in the training of neural networks for supervised learning. Gradient Descent is an iterative optimization algorithm for finding the minimum of a function. The algorithm takes steps proportional to the negative gradient of the function at the current point.

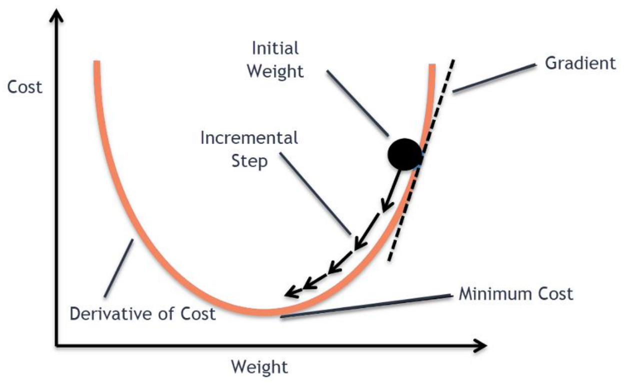

Figure 3: A plot explaining Gradient Descent algorithm

In deep learning, neural networks are trained by defining a loss function and optimizing the parameters of the network to obtain the minimum of the function. The optimization is done using the gradient descent algorithm. It is to pass the training set through the hidden layers of the neural network and then update the parameters of the layers by computing the gradients using the training samples from the training dataset. This procedure can be done in the following ways.

1. Stochastic Gradient Descent
2. Batch Gradient Descent

In Stochastic Gradient Descent method, one training sample is passed through the neural network at a time and the parameters of each layer are updated with the computed gradient. So, at a time a single training sample is passed through the network and its corresponding loss is computed. The parameters of all the layers of the network are updated after every training sample.

The concept of Batch Gradient Descent is similar to stochastic gradient descent. The difference is that instead of updating the parameters of the network after computing the loss of every training sample in the training set, the parameters are updated once, that is after all the training examples have been passed through the network.

For example, if the training set contains n number of samples, then the parameters are updated n times that is one time after every individual example is passed through the network in SGD, while in Batch GD, the parameters of the neural network are updated once.

The SGD and Batch GD have many advantages and disadvantages. Some of them are listed below.

- SGD reaches the convergence much faster and convergence in Batch GD is slow. For larger datasets, SGD can converge faster as it causes updates to the parameters more frequently. But Batch GD is not suitable for huge training samples.
- Batch GD is computationally efficient as all computer resources are not being used to process a single sample rather are being used for all training samples but since frequent updates are done in SGD, it is computationally expensive due to using all resources for processing one training sample at a time.
- In SGD, due to frequent updates, the steps taken towards the minima of the loss function have oscillations which can help getting out of local minimums of the loss function (in case the computed position turns out to be the local minimum). In Batch GD too sometimes a stable error gradient can lead to a local minima and unlike stochastic gradient descent no noisy steps are there to help get out of the local minima.
- It produces a more stable gradient descent convergence and stable error gradient than stochastic gradient descent, thus giving an optimal solution given sufficient time to converge. But, in SGD, good solutions are obtained but they are not optimal.

Answered by: Marzook F.S. (E/16/232)

2.6 Can you explain about the N-bit parity problem?

In the study of neural networks, parity problem is a major concern. It has a long history in this field. It is used as a basis for illustrating the limitations of computational power of perceptrons. The parity mapping problem has since been recognized as one of the most popular benchmarks in evaluating neural network training algorithms. The N-bit parity function is a mapping defined on 2N distinct binary vectors that indicates whether the sum of the N components of a binary vector is odd or even.

$$\psi(\beta) = \begin{cases} 1 & \text{if } \sum_{k=1}^{N} \beta_k \text{ is odd,} \\ 0 & \text{if } \sum_{k=1}^{N} \beta_k \text{ is even.} \end{cases}$$

.

The parity mapping is considered difficult for neural network learning since changes in a single bit result in changes in the output.

Answered by: Amasith K.T.D (E/16/025)

2.7 What does convergence means and why do highway networks converge faster?

In this context, artificial neural networks convergence can be described as a progression towards a network state where the network has learned to properly respond to a set of training patterns within some margin of error.

Essentially meaning, a model converges when its loss actually moves towards a minima (local or global) with a decreasing trend is strictly speaking rarely exists practically, but is spoken in a manner telling us how close the model is to the ideal scenario for convexity, or in this case convergence.

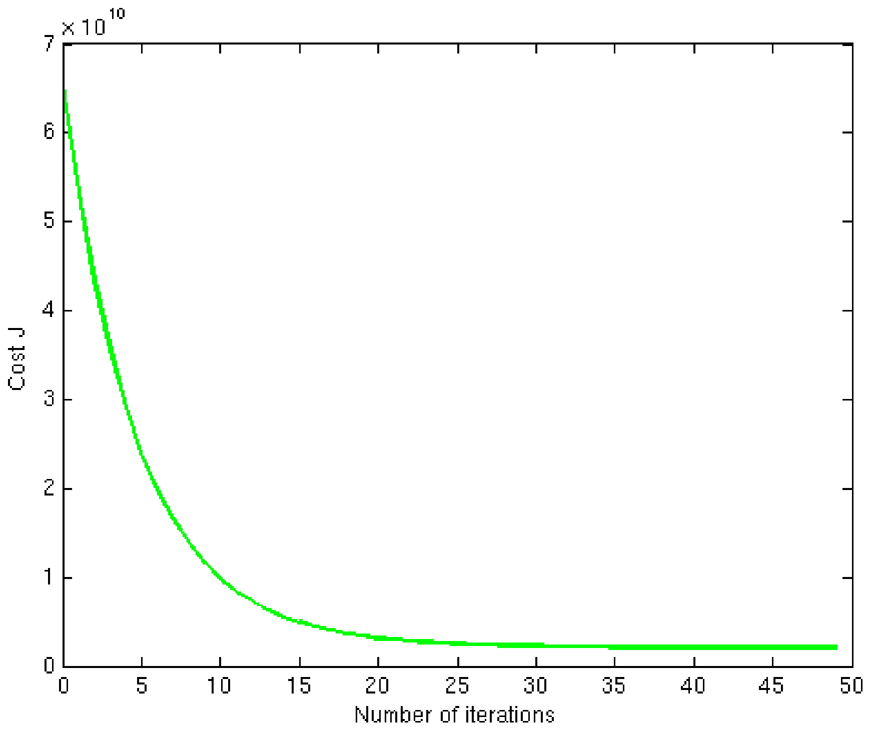

Figure 4: Convergence of a training model with number of iterations

In general, training model reaches convergence when it achieves a state during training in which loss settles to within an error range around the final value. We can simply say a model converges when additional training will not improve the model.

Answered by: Amasith K.T.D (E/16/025)

2.8 Why do highway networks converge faster?

Fast convergence is possible due to the architecture of the highway network. Let's see how it differ from normal network. Let’s start with plain network which consists of L layers where the L-th layer is,

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

where x is input, W_H is the weight, H is the transform function followed by an activation function and y is the output and for i-th unit,

$$y_i = H_i(\mathbf{x})$$

Normally, we compute the y_i and pass it to the next layer.

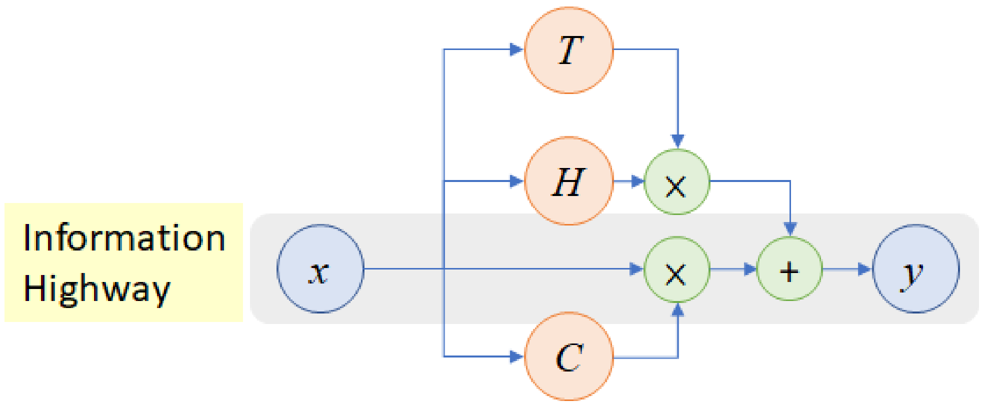

Figure 5: Highway circuit

- In highway network, two non-linear transforms T and C are introduced.

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

- ***T* is the Transform Gate** and ***C* is the Carry Gate**.
- In particular, ***C* = *1* – *T***,

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

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

When T=0, we pass the input as output directly which creates an information highway. That’s why it is called Highway Network. In other words, lesioning. It is meant to manually set all the transform gates of a layer to 0 forcing it to simply copy its inputs. This architecture causes faster convergence and also, we can see less number of layers needed in deep highway networks. This mostly can be seen efficiently in large datasets.

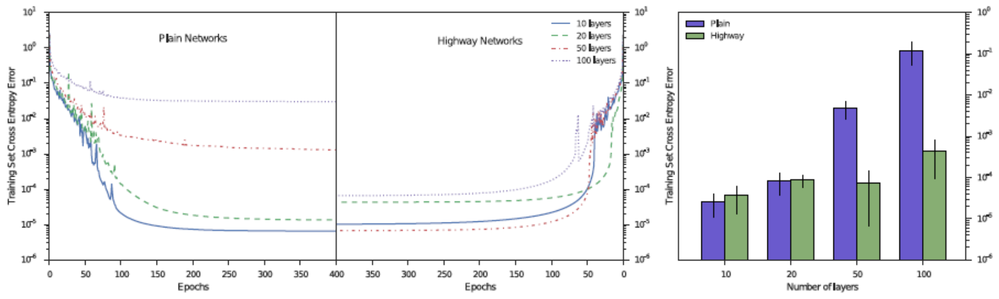

Figure 6: Comparison of optimization of plain networks and highway networks of various depths.

Left of Figure 1 shows the training curves for the best hyperparameter settings obtained for each network depth while right of Figure 1 shows mean performance of top 10 (out of 100) hyperparameter settings.

Here you can see that when we have a large data set, by using highway networks we can reduce training time as well as it is giving more accuracy.

Answered by: Amasith K.T.D (E/16/025)

2.9 What are the differences between Ciphar-10 and Ciphar-100 datasets?

- CIFAR-10
    - Consists of 60000 of 32x32 colour images in 10 classes, with 6000 images per class
    - 50000 training images and 10000 test images

- CIFAR-100
    - 100 classes containing 600 images each
    - 500 training images and 100 testing images per class
    - 100 classes are grouped into 20 super classes

Answered by: Madushanka S.D.W. (E/16/222)

2.10 What is the learning rate in neural networks, how learning rate affects the training process of the network, what is the purpose of learning rate, what happens if the learning rate is high and low?

Deep learning neural networks are trained using the stochastic gradient descent algorithm. Stochastic gradient descent is an optimization algorithm that estimates the error gradient for the current state of the model using examples from the training dataset, then updates the weights of the model using the back-propagation of errors algorithm, referred to as simply back propagation.

The amount that the weights are updated during training is referred to as the step size or the "learning rate." The learning rate is a hyperparameter that controls how much to change the model in response to the estimated error each time the model weights are updated. The challenge of training deep learning neural networks involves carefully selecting the learning rate. It may be the most important hyperparameter for the model.

Normally the value of learning rate is between 0.0 and 1.0. Learning rate controls how quickly the model is adapted or response to the given problem. If the learning rate is small, time required to train the model increases because at each iteration weight that is in the model is going to change slowly. It needs more and more CPU calculations. In that case, it gets stuck in the process. So, choosing a small learning rate raises new problems. However, a small learning rate gives a more accurate solution to the problem.

If the learning rate is too large, time takes to train the model small because weights are changed by a large amount of size case to overshooting. We don't need much CPU power in this case. But finally what we need is a more optimal solution, but if we choose large learning rate, we may miss the optimal solution as indicated in Figure 7.

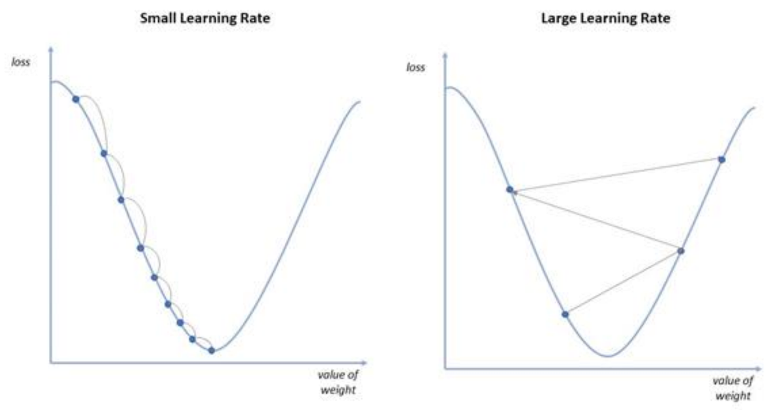

Figure 7: Small learning rate vs large learning rate

Answered by: Madushanka S.D.W. (E/16/222)

2.11 What are the algorithms that can be used to stop the neural network from overfitting?

Basically, overfitting occurs when a model tries to predict a trend in data that is too noisy. One of main reason is due to an overly complex model with too many parameters. If the model produces good results on seen data (training set) but performs poorly on the unseen data (test set) and that's because of overfitting. To overcome this, we can use following techniques.

1. Simplify the model

   One of the steps is decreasing the model complexity. If the model is too complex the model fits to the training data set, it give best accuracy in training data set but not in unseen data set. If the model complexity is low, the model gives wrong output. The best thing is to choose middle complexity.

To reduce the complexity, we can reduce the number of neurons by removing the layers. We need to do training several times with several complexity and choose best complexity for the model.

2. Early stopping

   Early stopping is a form of regularization while training a model with an iterative method, such as gradient descent. Since all the neural networks learn exclusively by using gradient descent, early stopping is a technique applicable to all the problems. In each iteration, model fits more and more with training data set, but it’s up to some point, after that point generalization error increases.

   Therefore, in early stopping technique what we do is stopping the iteration after some point as shown in figure 8.

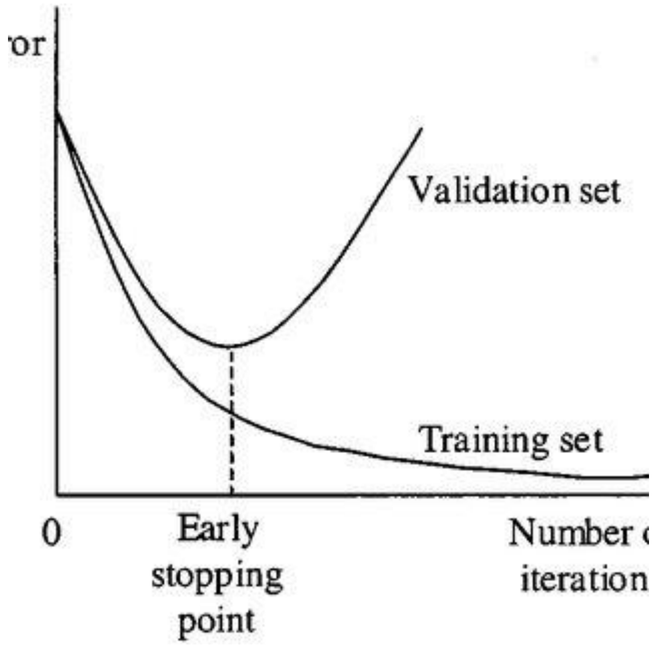

Figure 8: Generalization error variation with number of iterations

- Use Data Augmentation

In the case of neural networks, data augmentation simply means increasing size of the data set. For example, in an image data set, normally what is done to increase data set is translation, rotation, scaling, changing brightness of current data set. By increasing data set size, model gets more and more generalize.

- Use Dropouts

Dropout modifies the network itself. It randomly drops neurons from the neural network during training in each iteration. It is like training different neural networks. So, different networks act different way in overfitting problem, caused to reduce the overall overfitting.

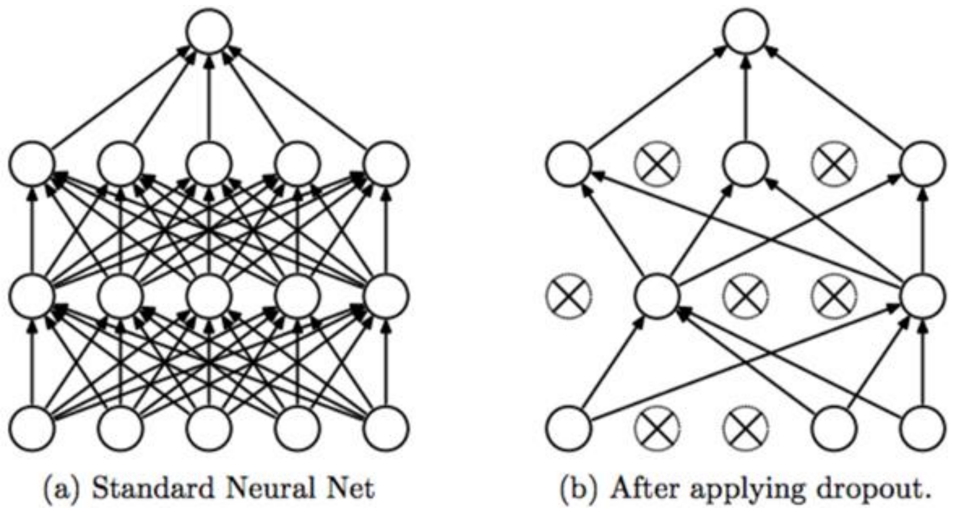

Figure 9: Before and after using dropout in neural networks

Answered by: Madushanka S.D.W. (E/16/222)

3 Speaker Notes

3.1 Explanation on Analysis of Highway Networks (Speaker Notes by Chamath Amarasinghe – E/16/022)

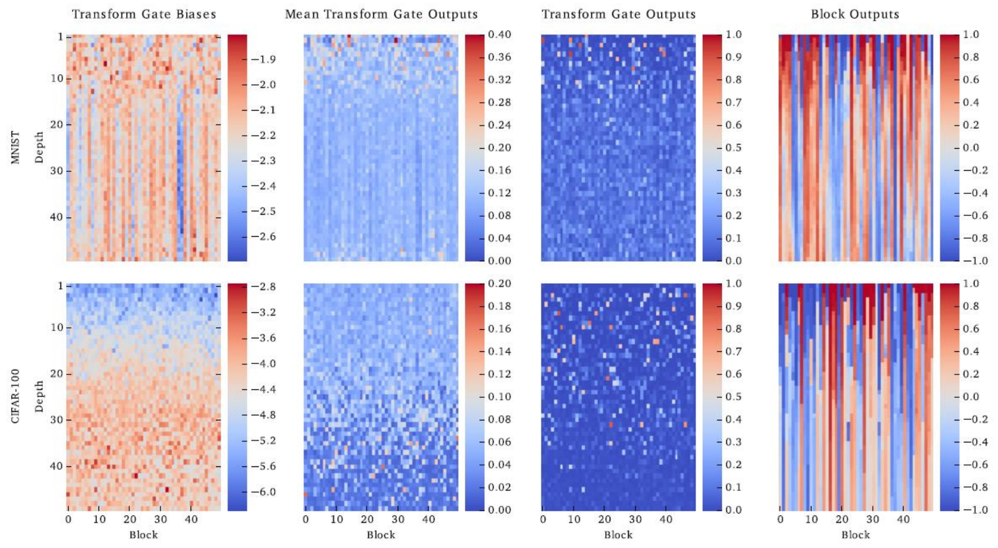

Figure 10: Visualization of best 50 hidden-layer highway networks trained on MNIST (top row) and CIFAR-100 (bottom row)

For each transform gate, the bias, mean activity overall training samples, and activity for a single random sample are shown in the figures above. The last column shows block outputs for the same single sample.

The biases in the CIFAR-100 network increase with depth, generating a gradient. The large negative biases at low depths are utilized to make the gates more selective, not to shut them down. This results in an extremely sparse transform gate activity for a single example (column 3).

Most transform gates are active on average in the CIFAR-100 scenario, but they are quite selective in the single example. This means that only a few blocks conduct alteration for each sample, but various blocks are used by different samples.

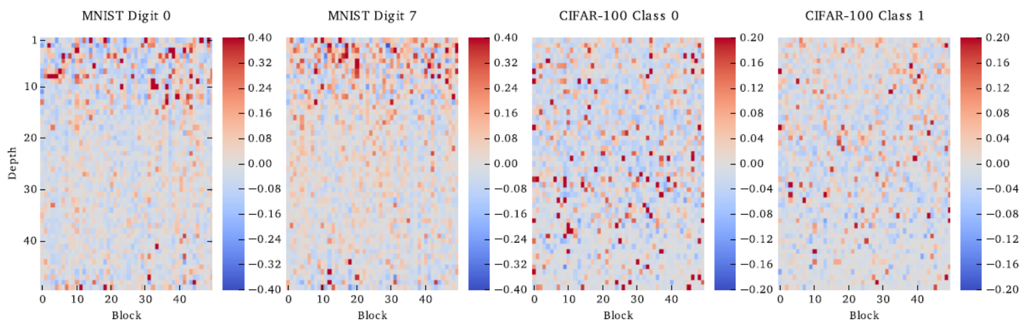

Figure 11: Visualization showing the extent to which the mean transform gate activity for certain classes differs from the mean activity overall training samples

Significant discrepancies can be found within the first 15 levels for MNIST digits 0 and 7. It can be observed that the differences between CIFAR classes 0 and 1 are sparser and spread-out over-all layers.

References

[1] Srivastava, R.K., Greff K. and Schmidhuber, J., 2015. Training very deep networks. Advances in neural information processing systems, 28, pp.2377-2385.

[2] Review: Highway Networks - Gating Function To Highway https://towardsdatascience.com/review-highway-networks-gating-function-to-highway-image-classification-5a33833797b5

[3] Klaus Greff, Rupesh Kumar Srivastava, Jan Koutník, Bas R. Steunebrink, Jürgen Schmidhuber. LSTM: A Search Space Odyssey IEEE Transactions on Neural Networks and Learning Systems (Volume: 28, Issue: 10, Oct. 2017) Pages: 2222 – 2232 https://arxiv.org/pdf/1505.00387.pdf

[4] Five techniques to Prevent Overfitting in Neural Networks https://www.kdnuggets.com/2019/12/5-techniques-prevent-overfitting-neural-networks.html

[5] Training very deep networks reading group presentation of the research paper from Srivastava, R.K., Greff K. and Schmidhuber, J., 2015. Training very deep networks. Reading group presentation by Amarasinghe D.L.C., Amasith K.T.D., Madushanka S.D.W., Marzook F.S. https://www.youtube.com/watch?v=3-HGbVxdgnI