Training Very Deep Networks

Rupesh Kumar Srivastava **Klaus Greff** **Jürgen Schmidhuber**

The Swiss AI Lab IDSIA / USI / SUPSI
{rupesh, klaus, juergen}@idsia.ch

Abstract

Theoretical and empirical evidence indicates that the depth of neural networks is crucial for their success. However, training becomes more difficult as depth increases, and training of very deep networks remains an open problem. Here we introduce a new architecture designed to overcome this. Our so-called highway networks allow unimpeded information flow across many layers on *information highways*. They are inspired by Long Short-Term Memory recurrent networks and use adaptive gating units to regulate the information flow. Even with hundreds of layers, highway networks can be trained directly through simple gradient descent. This enables the study of extremely deep and efficient architectures.

1 Introduction & Previous Work

Trong phần **1. Introduction & Previous Work**, tác giả giới thiệu về những thành tựu gần đây trong **học máy giám sát** nhờ vào việc sử dụng các mạng nơ-ron sâu và lớn. Họ nhấn mạnh rằng **độ sâu của mạng** (số lượng các lớp tính toán kế tiếp nhau) đóng vai trò quan trọng nhất trong thành công này.

Điểm chính:

- **Mạng nơ-ron sâu** đã giúp tăng độ chính xác trong các nhiệm vụ phân loại hình ảnh. Ví dụ, trong vài năm, độ chính xác của top-5 trên tập dữ liệu ImageNet đã tăng từ khoảng 84% lên 95% nhờ việc sử dụng các mạng sâu hơn với các receptive field nhỏ hơn.
- Mạng nơ-ron sâu có thể biểu diễn các lớp hàm số hiệu quả hơn so với các mạng nông. Điều này rõ ràng nhất đối với các mạng hồi quy (recurrent nets). Ví dụ, bài toán n-bit parity có thể được giải quyết một cách tự nhiên bởi mạng hồi quy với chỉ 3 đơn vị và 5 trọng số, thay vì mạng feedforward với một số lượng lớn đầu vào.

Để xử lý các khó khăn khi huấn luyện các mạng sâu, các nhà nghiên cứu đã tập trung phát triển các phương pháp như:

- **Bộ tối ưu hóa tốt hơn** và **chiến lược khởi tạo được thiết kế kỹ lưỡng**, đặc biệt là các phương pháp khởi tạo bảo toàn phương sai, đã được áp dụng rộng rãi cho mạng có độ sâu vừa phải.
- Một số phương pháp khác, như **kết nối skip** để cải thiện dòng chảy thông tin, và **huấn luyện mạng sâu** bằng cách sử dụng các mục tiêu từ mạng nông hơn, cũng đã được thử nghiệm.

Vấn đề của mạng sâu:

- Mặc dù đã có nhiều giải pháp, việc huấn luyện mạng sâu vẫn gặp khó khăn, đặc biệt là với vấn đề gradient biến mất (vanishing gradients) trong các mạng hồi quy chuẩn. Các kiến trúc mạng feedforward thông thường gặp khó khăn trong việc lan truyền các kích hoạt và gradient, làm cho việc tối ưu mạng sâu trở nên khó khăn hơn.

Giải pháp của tác giả:

Để khắc phục vấn đề này, tác giả lấy cảm hứng từ các mạng **LSTM** và đề xuất **mạng highway**. Ý tưởng là sửa đổi kiến trúc của các mạng feedforward rất sâu để làm cho dòng thông tin qua các lớp dễ dàng hơn nhờ một cơ chế "gating" thích nghi, cho phép thông tin **chảy qua nhiều lớp mà không bị suy giảm**. Các "paths" này được gọi là **information highways** (đường cao tốc thông tin), tạo nên mạng highway, khác với các mạng thông thường.

Đóng góp chính:

Tác giả chỉ ra rằng các mạng "highway" rất sâu có thể được huấn luyện trực tiếp bằng **stochastic gradient descent (SGD)**, trong khi các mạng thông thường trở nên khó tối ưu khi tăng độ sâu. Ngoài ra, các thí nghiệm của họ cũng chỉ ra rằng **mạng highway** dễ dàng tổng quát hóa tốt với dữ liệu chưa được thấy trước đó.

To overcome this, we take inspiration from Long Short Term Memory (LSTM) recurrent networks [29, 30]. We propose to modify the architecture of very deep feedforward networks such that information flow across layers becomes much easier. This is accomplished through an LSTM-inspired adaptive gating mechanism that allows for computation paths along which information can flow across many layers without attenuation. We call such paths *information highways*. They yield *highway networks*, as opposed to traditional 'plain' networks.[1]

Our primary contribution is to show that extremely deep highway networks can be trained directly using stochastic gradient descent (SGD), in contrast to plain networks which become hard to optimize as depth increases (Section 3.1). Deep networks with limited computational budget (for which a two-stage training procedure mentioned above was recently proposed [25]) can also be directly trained in a single stage when converted to highway networks. Their ease of training is supported by experimental results demonstrating that highway networks also generalize well to unseen data.

2 Highway Networks

Notation We use boldface letters for vectors and matrices, and italicized capital letters to denote transformation functions. $\mathbf{0}$ and $\mathbf{1}$ denote vectors of zeros and ones respectively, and $\mathbf{I}$ denotes an identity matrix. The function $\sigma(x)$ is defined as $\sigma(x) = \frac{1}{1+e^{-x}}, x \in \mathbb{R}$. The dot operator $(\cdot)$ is used to denote element-wise multiplication.

A *plain* feedforward neural network typically consists of L layers where the l^{th} layer ($l \in \{1, 2, ..., L\}$) applies a non-linear transformation H (parameterized by $\mathbf{W_{H,l}}$) on its input $\mathbf{x_l}$ to produce its output $\mathbf{y_l}$. Thus, $\mathbf{x_1}$ is the input to the network and $\mathbf{y_L}$ is the network's output. Omitting the layer index and biases for clarity,

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

H is usually an affine transform followed by a non-linear activation function, but in general it may take other forms, possibly convolutional or recurrent. For a highway network, we additionally define two non-linear transforms $T(\mathbf{x}, \mathbf{W_T})$ and $C(\mathbf{x}, \mathbf{W_C})$ such that

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

We refer to T as the *transform* gate and C as the *carry* gate, since they express how much of the output is produced by transforming the input and carrying it, respectively. For simplicity, in this paper we set $C = 1 - T$, giving

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

The dimensionality of $\mathbf{x}, \mathbf{y}, H(\mathbf{x}, \mathbf{W_H})$ and $T(\mathbf{x}, \mathbf{W_T})$ must be the same for Equation 3 to be valid. Note that this layer transformation is much more flexible than Equation 1. In particular, observe that for particular values of T,

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

Similarly, for the Jacobian of the layer transform,

[1] This paper expands upon a shorter report on Highway Networks [31]. More recently, a similar LSTM-inspired model was also proposed [32].

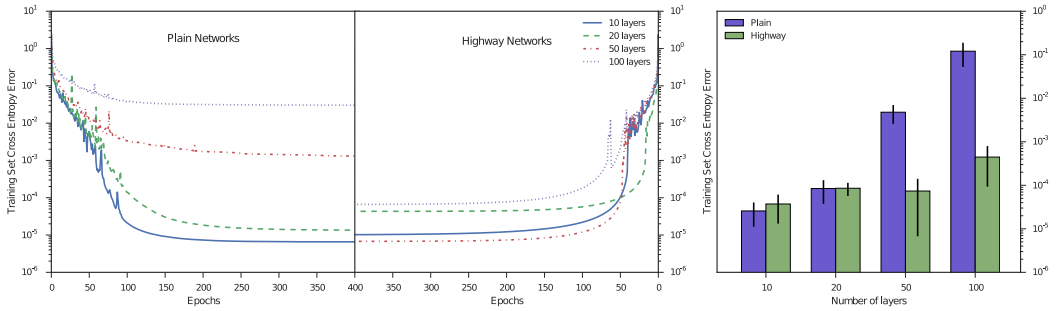

Figure 1: Comparison of optimization of plain networks and highway networks of various depths. *Left:* The training curves for the best hyperparameter settings obtained for each network depth. *Right:* Mean performance of top 10 (out of 100) hyperparameter settings. Plain networks become much harder to optimize with increasing depth, while highway networks with up to 100 layers can still be optimized well. Best viewed on screen (larger version included in Supplementary Material).

$$\frac{d\mathbf{y}}{d\mathbf{x}} = \begin{cases} \mathbf{I}, & \text{if } T(\mathbf{x}, \mathbf{W_T}) = \mathbf{0}, \\ H'(\mathbf{x}, \mathbf{W_H}), & \text{if } T(\mathbf{x}, \mathbf{W_T}) = \mathbf{1}. \end{cases} \tag{5}$$

Thus, depending on the output of the transform gates, a highway layer can smoothly vary its behavior between that of H and that of a layer which simply passes its inputs through. Just as a plain layer consists of multiple computing units such that the i^{th} unit computes $y_i = H_i(\mathbf{x})$, a highway network consists of multiple **blocks** such that the i^{th} block computes a **block state** $H_i(\mathbf{x})$ and **transform gate output** $T_i(\mathbf{x})$. Finally, it produces the **block output** $y_i = H_i(\mathbf{x}) * T_i(\mathbf{x}) + x_i * (1 - T_i(\mathbf{x}))$, which is connected to the next layer.[2]

2.1 Constructing Highway Networks

As mentioned earlier, Equation 3 requires that the dimensionality of $\mathbf{x}, \mathbf{y}, H(\mathbf{x}, \mathbf{W_H})$ and $T(\mathbf{x}, \mathbf{W_T})$ be the same. To change the size of the intermediate representation, one can replace $\mathbf{x}$ with $\hat{\mathbf{x}}$ obtained by suitably sub-sampling or zero-padding $\mathbf{x}$. Another alternative is to use a plain layer (without highways) to change dimensionality, which is the strategy we use in this study.

Convolutional highway layers utilize weight-sharing and local receptive fields for both H and T transforms. We used the same sized receptive fields for both, and zero-padding to ensure that the block state and transform gate feature maps match the input size.

2.2 Training Deep Highway Networks

We use the transform gate defined as $T(\mathbf{x}) = \sigma({\mathbf{W_T}}^T\mathbf{x} + \mathbf{b_T})$, where $\mathbf{W_T}$ is the weight matrix and $\mathbf{b_T}$ the bias vector for the transform gates. This suggests a simple initialization scheme which is independent of the nature of H: b_T can be initialized with a negative value (e.g. -1, -3 etc.) such that the network is initially biased towards *carry* behavior. This scheme is strongly inspired by the proposal [30] to initially bias the gates in an LSTM network, to help bridge long-term temporal dependencies early in learning. Note that $\sigma(x) \in (0, 1), \forall x \in \mathbb{R}$, so the conditions in Equation 4 can never be met exactly.

In our experiments, we found that a negative bias initialization for the transform gates was sufficient for training to proceed in very deep networks for various zero-mean initial distributions of W_H and different activation functions used by H. In pilot experiments, SGD did not stall for networks with more than 1000 layers. Although the initial bias is best treated as a hyperparameter, as a general guideline we suggest values of -1, -2 and -3 for convolutional highway networks of depth approximately 10, 20 and 30.

[2]Our pilot experiments on training very deep networks were successful with a more complex block design closely resembling an LSTM block "unrolled in time". Here we report results only for a much simplified form.

Network	Highway Networks		Maxout [20]	DSN [24]
	10-layer (width 16)	10-layer (width 32)		
No. of parameters	39 K	151 K	420 K	350 K
Test Accuracy (in %)	99.43 (99.4±0.03)	99.55 (99.54±0.02)	99.55	99.61

Table 1: Test set classification accuracy for pilot experiments on the MNIST dataset.

Network	**No. of Layers**	**No. of Parameters**	**Accuracy (in %)**
Fitnet Results (reported by Romero et. al.[25])			
Teacher	5	∼9M	90.18
Fitnet A	11	∼250K	89.01
Fitnet B	19	∼2.5M	91.61
Highway networks			
Highway A (Fitnet A)	11	∼236K	89.18
Highway B (Fitnet B)	19	∼2.3M	**92.46 (92.28±0.16)**
Highway C	32	∼1.25M	91.20

Table 2: CIFAR-10 test set accuracy of convolutional highway networks. Architectures tested were based on *fitnets* trained by Romero et. al. [25] using two-stage hint based training. Highway networks were trained in a single stage without hints, matching or exceeding the performance of fitnets.

3 Experiments

All networks were trained using SGD with momentum. An exponentially decaying learning rate was used in Section 3.1. For the rest of the experiments, a simpler commonly used strategy was employed where the learning rate starts at a value λ and decays according to a fixed schedule by a factor γ. λ, γ and the schedule were selected once based on validation set performance on the CIFAR-10 dataset, and kept fixed for all experiments. All convolutional highway networks utilize the rectified linear activation function [16] to compute the block state H. To provide a better estimate of the variability of classification results due to random initialization, we report our results in the format *Best (mean ± std.dev.)* based on 5 runs wherever available. Experiments were conducted using Caffe [33] and Brainstorm (`https://github.com/IDSIA/brainstorm`) frameworks. Source code, hyperparameter search results and related scripts are publicly available at `http://people.idsia.ch/~rupesh/very_deep_learning/`.

3.1 Optimization

To support the hypothesis that highway networks do not suffer from increasing depth, we conducted a series of rigorous optimization experiments, comparing them to plain networks with normalized initialization [16, 17].

We trained both plain and highway networks of varying varying depths on the MNIST digit classification dataset. All networks are *thin*: each layer has 50 blocks for highway networks and 71 units for plain networks, yielding roughly identical numbers of parameters (≈5000) per layer. In all networks, the first layer is a fully connected plain layer followed by 9, 19, 49, or 99 fully connected plain or highway layers. Finally, the network output is produced by a *softmax* layer. We performed a random search of 100 runs for both plain and highway networks to find good settings for the following hyperparameters: initial learning rate, momentum, learning rate exponential decay factor & activation function (either *rectified linear* or *tanh*). For highway networks, an additional hyperparameter was the initial value for the transform gate bias (between -1 and -10). Other weights were initialized using the same normalized initialization as plain networks.

The training curves for the best performing networks for each depth are shown in Figure 1. As expected, 10 and 20-layer plain networks exhibit very good performance (mean loss $< 1e^{-4}$), which significantly degrades as depth increases, even though network capacity increases. Highway networks do not suffer from an increase in depth, and 50/100 layer highway networks perform similar to 10/20 layer networks. The 100-layer highway network performed more than 2 orders of magnitude better compared to a similarly-sized plain network. It was also observed that highway networks consistently converged significantly faster than plain ones.

Mục tiêu của tác giả không phải là đạt được kết quả tốt nhất có thể, mà là **chứng minh rằng các mạng sâu hơn** có thể được huấn luyện mà không phải đánh đổi về **tính dễ dàng trong huấn luyện** hoặc **khả năng tổng quát hóa** (generalization ability).

so sánh vs các kin trúckha

Do đó, các thí nghiệm của họ chỉ thực hiện trong những điều kiện thiết lập thông thường, chẳng hạn như **normalization tương phản toàn cục** (global contrast normalization), **dịch chuyển nhỏ** (small translations), và **lật hình ảnh** (mirroring of images).

Network	CIFAR-10 Accuracy (in %)	CIFAR-100 Accuracy (in %)
Maxout [20]	90.62	61.42
dasNet [36]	90.78	66.22
NiN [35]	91.19	64.32
DSN [24]	92.03	65.43
All-CNN [37]	**92.75**	66.29
Highway Network	92.40 (92.31±0.12)	**67.76 (67.61±0.15)**

Table 3: Test set accuracy of convolutional highway networks on the CIFAR-10 and CIFAR-100 object recognition datasets with typical data augmentation. For comparison, we list the accuracy reported by recent studies in similar experimental settings.

3.2 Pilot Experiments on MNIST Digit Classification

As a sanity check for the generalization capability of highway networks, we trained 10-layer convolutional highway networks on MNIST, using two architectures, each with 9 convolutional layers followed by a softmax output. The number of filter maps (width) was set to 16 and 32 for all the layers. We obtained test set performance competitive with state-of-the-art methods with much fewer parameters, as show in Table 1.

3.3 Experiments on CIFAR-10 and CIFAR-100 Object Recognition

3.3.1 Comparison to Fitnets

Fitnet training Maxout networks can cope much better with increased depth than those with traditional activation functions [20]. However, Romero et. al. [25] recently reported that training on CIFAR-10 through plain backpropogation was only possible for maxout networks with a depth up to 5 layers when the number of parameters was limited to ∼250K and the number of multiplications to ∼30M. Similar limitations were observed for higher computational budgets. Training of deeper networks was only possible through the use of a two-stage training procedure and addition of soft targets produced from a pre-trained shallow teacher network (hint-based training).

We found that it was easy to train highway networks with numbers of parameters and operations comparable to those of fitnets in a single stage using SGD. As shown in Table 2, Highway A and Highway B, which are based on the architectures of Fitnet A and Fitnet B, respectively, obtain similar or higher accuracy on the test set. We were also able to train thinner and deeper networks: for example a 32-layer highway network consisting of alternating receptive fields of size 3x3 and 1x1 with ∼1.25M parameters performs better than the earlier teacher network [20].

3.3.2 Comparison to State-of-the-art Methods

It is possible to obtain high performance on the CIFAR-10 and CIFAR-100 datasets by utilizing very large networks and extensive data augmentation. This approach was popularized by Ciresan et. al. [5] and recently extended by Graham [34]. Since our aim is only to demonstrate that deeper networks can be trained without sacrificing ease of training or generalization ability, we only performed experiments in the more common setting of global contrast normalization, small translations and mirroring of images. Following Lin et. al. [35], we replaced the fully connected layer used in the networks in the previous section with a convolutional layer with a receptive field of size one and a global average pooling layer. The hyperparameters from the last section were re-used for both CIFAR-10 and CIFAR-100, therefore it is quite possible to obtain much better results with better architectures/hyperparameters. The results are tabulated in Table 3.

4 Analysis

Figure 2 illustrates the inner workings of the best[3] 50 hidden layer fully-connected highway networks trained on MNIST (top row) and CIFAR-100 (bottom row). The first three columns show

[3]obtained via random search over hyperparameters to minimize the best training set error achieved using each configuration

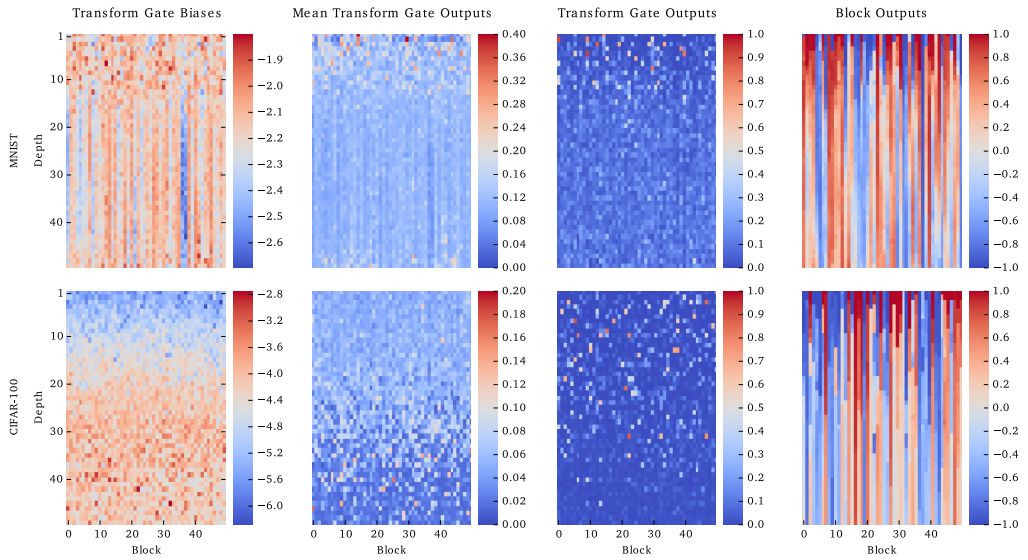

Figure 2: Visualization of best 50 hidden-layer highway networks trained on MNIST (top row) and CIFAR-100 (bottom row). The first hidden layer is a plain layer which changes the dimensionality of the representation to 50. Each of the 49 highway layers (y-axis) consists of 50 blocks (x-axis). The first column shows the transform gate biases, which were initialized to -2 and -4 respectively. In the second column the mean output of the transform gate over all training examples is depicted. The third and fourth columns show the output of the transform gates and the block outputs (both networks using tanh) for a single random training sample. Best viewed in color.

the bias, the mean activity over all training samples, and the activity for a single random sample for each transform gate respectively. Block outputs for the same single sample are displayed in the last column.

The transform gate biases of the two networks were initialized to -2 and -4 respectively. It is interesting to note that contrary to our expectations most biases decreased further during training. For the CIFAR-100 network the biases increase with depth forming a gradient. Curiously this gradient is inversely correlated with the average activity of the transform gates, as seen in the second column. This indicates that the strong negative biases at low depths are not used to shut down the gates, but to make them more selective. This behavior is also suggested by the fact that the transform gate activity for a single example (column 3) is very sparse. The effect is more pronounced for the CIFAR-100 network, but can also be observed to a lesser extent in the MNIST network.

The last column of Figure 2 displays the block outputs and visualizes the concept of "information highways". Most of the outputs stay constant over many layers forming a pattern of stripes. Most of the change in outputs happens in the early layers (≈ 15 for MNIST and ≈ 40 for CIFAR-100).

4.1 Routing of Information

One possible advantage of the highway architecture over hard-wired shortcut connections is that the network can learn to dynamically adjust the routing of the information based on the current input. This begs the question: does this behaviour manifest itself in trained networks or do they just learn a static routing that applies to all inputs similarly. A partial answer can be found by looking at the mean transform gate activity (second column) and the single example transform gate outputs (third column) in Figure 2. Especially for the CIFAR-100 case, most transform gates are active on average, while they show very selective activity for the single example. This implies that for each sample only a few blocks perform transformation but different blocks are utilized by different samples.

This data-dependent routing mechanism is further investigated in Figure 3. In each of the columns we show how the average over all samples of one specific class differs from the total average shown in the second column of Figure 2. For MNIST digits 0 and 7 substantial differences can be seen

Trong phần **4.1 Routing of Information**, tác giả giải thích một lợi ích tiềm năng của kiến trúc **highway network** là khả năng học cách điều chỉnh linh hoạt việc định tuyến thông tin dựa trên đầu vào hiện tại. Câu hỏi được đặt ra ở đây là liệu hành vi này có biểu hiện trong các mạng đã được huấn luyện hay không, hoặc liệu mạng chỉ học một tuyến đường tĩnh áp dụng giống nhau cho mọi đầu vào.

Tóm li, tác gi cho thy rng c ch nh tuyn thông tin trong mng "highway" là linh hot và ph thuc vào d liu, giúp mng này hc cách iu chnh cho tng

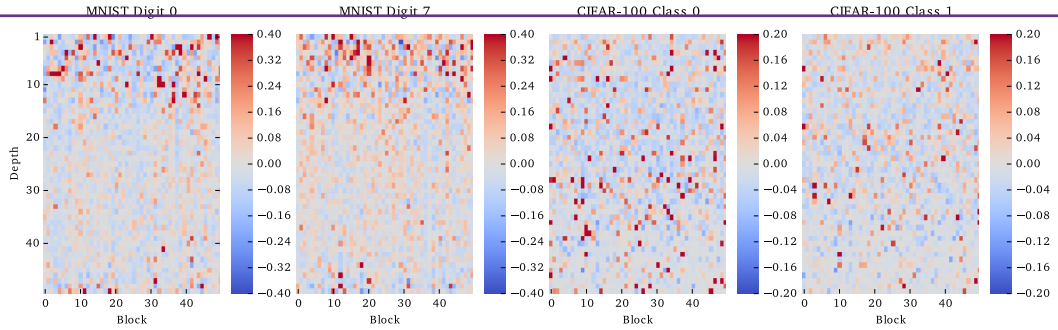

Figure 3: Visualization showing the extent to which the mean transform gate activity for certain classes differs from the mean activity over all training samples. Generated using the same 50-layer highway networks on MNIST on CIFAR-100 as Figure 2. Best viewed in color.

within the first 15 layers, while for CIFAR class numbers 0 and 1 the differences are sparser and spread out over all layers. In both cases it is clear that the mean activity pattern differs between classes. The gating system acts not just as a mechanism to ease training, but also as an important part of the computation in a trained network.

4.2 Layer Importance

Trong phần **4.2 Layer Importance**, tác giả thảo luận về **tầm quan trọng của các lớp** trong mạng "highway". Ban đầu, các "transform gate" được đặt nghiêng về trạng thái đóng, nghĩa là các lớp chủ yếu sao chép lại các kích hoạt của lớp trước đó. Câu hỏi đặt ra là liệu quá trình huấn luyện có thực sự thay đổi hành vi này hay không, hoặc liệu mạng cuối cùng có hoạt động tương đương với một mạng có ít lớp hơn.

Để kiểm tra vấn đề này, tác giả đã tiến hành một thử nghiệm bằng cách **loại bỏ từng lớp riêng biệt** trong mạng đã được huấn luyện (bằng cách tắt tất cả các cổng "transform gate" của lớp đó và buộc lớp chỉ sao chép đầu vào của nó). Kết quả được thể hiện ở **Hình 4**, trong đó mạng được đánh giá dựa trên hiệu suất sau khi đóng một lớp.

Kết quả:

- **MNIST**: Khi loại bỏ một lớp sớm trong mạng, sai số tăng đáng kể. Tuy nhiên, các lớp từ 15 đến 45 dường như không có ảnh hưởng nhiều đến hiệu suất cuối cùng. Khoảng 60% các lớp không đóng góp nhiều vào kết quả cuối cùng. Điều này được giải thích là do **MNIST là một bộ dữ liệu đơn giản** và không yêu cầu độ sâu mạng lớn.
- **CIFAR-100**: Đối với bộ dữ liệu phức tạp hơn như CIFAR-100, hiệu suất của mạng giảm đáng kể khi bất kỳ một trong số 40 lớp đầu tiên bị loại bỏ. Điều này cho thấy rằng với các vấn đề phức tạp, mạng "highway" có khả năng **tận dụng tất cả các lớp**. Trái lại, với các vấn đề đơn giản hơn, mạng sẽ giữ nhiều lớp không hoạt động.

5 Discussion

Trong phần **5. Discussion**, tác giả thảo luận về những lợi ích và hạn chế của các phương pháp tiếp cận khác khi đối mặt với các khó khăn do **chiều sâu mạng** gây ra, đặc biệt được đề cập trong **Section 1**.

Các phương pháp tiếp cận khác:

Tác giả đề cập rằng những phương pháp thay thế, như học cách chuyển thông tin qua các mạng nơ-ron bằng các tương tác lặp đi lặp lại (competitive interactions), có thể giúp cải thiện và mở rộng mạng để giải quyết các vấn đề phức tạp hơn. Tuy nhiên, những phương pháp này vẫn gặp phải các hạn chế khi mạng sâu hơn khoảng 20 lớp, ngay cả khi sử dụng các kỹ thuật khởi tạo cẩn thận.

- **Khởi tạo hiệu quả** là khó khăn vì chúng cần phải được thiết lập phù hợp với các hàm kích hoạt khác nhau.
- Một kỹ thuật khác, **Deep supervision**, cũng không cải thiện được nhiều và thậm chí có thể gây hại cho mạng nông nhưng sâu.

MNIST:

- Đối với bộ dữ liệu đơn giản như MNIST, ta thấy rằng việc loại bỏ một trong những lớp đầu tiên (lớp từ 1 đến 10) gây ra **sự tăng đáng kể trong sai số**.
- Tuy nhiên, khi loại bỏ các lớp từ lớp thứ 15 trở đi, sai số hầu như không thay đổi nhiều. Điều này củng cố giả thuyết rằng các lớp sau không đóng góp nhiều vào hiệu suất đối với MNIST, một bài toán tương đối đơn giản.

CIFAR-100:

- Trái ngược với MNIST, kết quả của CIFAR-100 cho thấy rằng việc loại bỏ bất kỳ lớp nào trong số 40 lớp đầu tiên đều gây ra sự **giảm sút đáng kể trong hiệu suất**.
- Hiệu suất chỉ ổn định khi đã loại bỏ các lớp từ lớp 40 trở đi, điều này cho thấy mạng phải sử dụng toàn bộ các lớp để đạt được hiệu suất tốt nhất với dữ liệu phức tạp hơn như CIFAR-100.

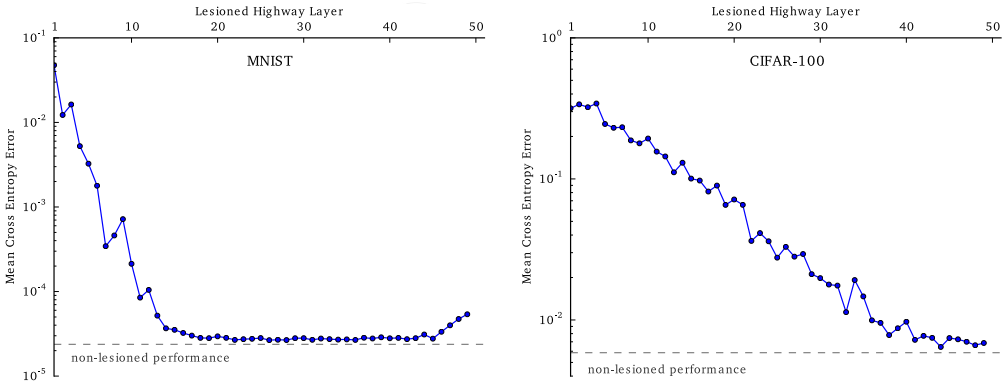

Figure 4: Lesioned training set performance (y-axis) of the best 50-layer highway networks on MNIST (left) and CIFAR-100 (right), as a function of the lesioned layer (x-axis). Evaluated on the full training set while forcefully closing all the transform gates of a single layer at a time. The non-lesioned performance is indicated as a dashed line at the bottom.

Mạng Highway Network:

Ngược lại, mạng "highway network" rất sâu có thể được huấn luyện dễ dàng hơn bằng các phương pháp gradient descent do kiến trúc đặc trưng của nó. Mạng này không phụ thuộc vào các phép biến đổi phi tuyến cụ thể (như các phép biến đổi hồi quy hoặc tích chập) và cũng không yêu cầu một **cơ chế khởi tạo** phức tạp. Cơ chế "gating" (cổng) bổ sung giúp định tuyến thông tin thông qua các kết nối nhân (multiplicative connections), khác với các kết nối "skip" cố định.

Sự phản đối có thể có:

Một mối lo ngại là nhiều lớp có thể không được sử dụng nếu các cổng "transform gate" bị đóng lại. Tuy nhiên, các thí nghiệm cho thấy rằng điều này **không ảnh hưởng tiêu cực đến mạng**. Mạng "highway" sâu và hẹp vẫn có thể đạt độ chính xác tương đương hoặc thậm chí vượt trội so với các mạng nông nhưng rộng (như mạng maxout). Điều này cho thấy các lớp không bị bỏ phí và vẫn thực hiện những tính toán quan trọng.

Tác giả cũng chỉ ra rằng, nhờ cấu trúc của mạng highway, ta có thể **đánh giá trực tiếp sự đóng góp của từng lớp**, như được mô tả trong **Hình 4**. Đây là lần đầu tiên một mạng "highway network" cho phép ta phân tích độ sâu tính toán cần thiết cho một vấn đề cụ thể, điều mà không dễ dàng thực hiện được với các mạng thông thường.

References

[1] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems*, 2012.

[2] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. *arXiv:1409.4842 [cs]*, September 2014.

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

[4] DC Ciresan, Ueli Meier, Jonathan Masci, Luca M Gambardella, and Jürgen Schmidhuber. Flexible, high performance convolutional neural networks for image classification. In *IJCAI*, 2011.

[5] Dan Ciresan, Ueli Meier, and Jürgen Schmidhuber. Multi-column deep neural networks for image classification. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2012.

[6] Dong Yu, Michael L. Seltzer, Jinyu Li, Jui-Ting Huang, and Frank Seide. Feature learning in deep neural networks-studies on speech recognition tasks. *arXiv preprint arXiv:1301.3605*, 2013.

[7] Sepp Hochreiter and Jurgen Schmidhuber. Bridging long time lags by weight guessing and "long short-term memory". *Spatiotemporal models in biological and artificial systems*, 37:65–72, 1996.

[8] Johan Håstad. *Computational limitations of small-depth circuits*. MIT press, 1987.

[9] Johan Håstad and Mikael Goldmann. On the power of small-depth threshold circuits. *Computational Complexity*, 1(2):113–129, 1991.

[10] Monica Bianchini and Franco Scarselli. On the complexity of neural network classifiers: A comparison between shallow and deep architectures. *IEEE Transactions on Neural Networks*, 2014.

[11] Guido F Montufar, Razvan Pascanu, Kyunghyun Cho, and Yoshua Bengio. On the number of linear regions of deep neural networks. In *Advances in Neural Information Processing Systems*. 2014.

[12] James Martens and Venkatesh Medabalimi. On the expressive efficiency of sum product networks. *arXiv:1411.7717 [cs, stat]*, November 2014.

[13] James Martens and Ilya Sutskever. Training deep and recurrent networks with hessian-free optimization. *Neural Networks: Tricks of the Trade*, pages 1–58, 2012.

[14] Ilya Sutskever, James Martens, George Dahl, and Geoffrey Hinton. On the importance of initialization and momentum in deep learning. pages 1139–1147, 2013.

[15] Yann N Dauphin, Razvan Pascanu, Caglar Gulcehre, Kyunghyun Cho, Surya Ganguli, and Yoshua Bengio. Identifying and attacking the saddle point problem in high-dimensional non-convex optimization. In *Advances in Neural Information Processing Systems 27*, pages 2933–2941. 2014.

[16] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *International Conference on Artificial Intelligence and Statistics*, pages 249–256, 2010.

[17] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on ImageNet classification. *arXiv:1502.01852 [cs]*, February 2015.

[18] David Sussillo and L. F. Abbott. Random walk initialization for training very deep feedforward networks. *arXiv:1412.6558 [cs, stat]*, December 2014.

[19] Andrew M. Saxe, James L. McClelland, and Surya Ganguli. Exact solutions to the nonlinear dynamics of learning in deep linear neural networks. *arXiv:1312.6120 [cond-mat, q-bio, stat]*, December 2013.

[20] Ian J. Goodfellow, David Warde-Farley, Mehdi Mirza, Aaron Courville, and Yoshua Bengio. Maxout networks. *arXiv:1302.4389 [cs, stat]*, February 2013.

[21] Rupesh K. Srivastava, Jonathan Masci, Sohrob Kazerounian, Faustino Gomez, and Jürgen Schmidhuber. Compete to compute. In *Advances in Neural Information Processing Systems*, pages 2310–2318, 2013.

[22] Tapani Raiko, Harri Valpola, and Yann LeCun. Deep learning made easier by linear transformations in perceptrons. In *International Conference on Artificial Intelligence and Statistics*, pages 924–932, 2012.

[23] Alex Graves. Generating sequences with recurrent neural networks. *arXiv:1308.0850*, 2013.

[24] Chen-Yu Lee, Saining Xie, Patrick Gallagher, Zhengyou Zhang, and Zhuowen Tu. Deeply-supervised nets. pages 562–570, 2015.

[25] Adriana Romero, Nicolas Ballas, Samira Ebrahimi Kahou, Antoine Chassang, Carlo Gatta, and Yoshua Bengio. FitNets: Hints for thin deep nets. *arXiv:1412.6550 [cs]*, December 2014.

[26] Jürgen Schmidhuber. Learning complex, extended sequences using the principle of history compression. *Neural Computation*, 4(2):234–242, March 1992.

[27] Geoffrey E. Hinton, Simon Osindero, and Yee-Whye Teh. A fast learning algorithm for deep belief nets. *Neural computation*, 18(7):1527–1554, 2006.

[28] Sepp Hochreiter. *Untersuchungen zu dynamischen neuronalen Netzen*. Masters thesis, Technische Universität München, München, 1991.

[29] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, November 1997.

[30] Felix A. Gers, Jürgen Schmidhuber, and Fred Cummins. Learning to forget: Continual prediction with LSTM. In *ICANN*, volume 2, pages 850–855, 1999.

[31] Rupesh Kumar Srivastava, Klaus Greff, and Jürgen Schmidhuber. Highway networks. *arXiv:1505.00387 [cs]*, May 2015.

[32] Nal Kalchbrenner, Ivo Danihelka, and Alex Graves. Grid long Short-Term memory. *arXiv:1507.01526 [cs]*, July 2015.

[33] Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross Girshick, Sergio Guadarrama, and Trevor Darrell. Caffe: Convolutional architecture for fast feature embedding. *arXiv:1408.5093 [cs]*, 2014.

[34] Benjamin Graham. Spatially-sparse convolutional neural networks. *arXiv:1409.6070*, September 2014.

[35] Min Lin, Qiang Chen, and Shuicheng Yan. Network in network. *arXiv:1312.4400*, 2014.

[36] Marijn F Stollenga, Jonathan Masci, Faustino Gomez, and Jürgen Schmidhuber. Deep networks with internal selective attention through feedback connections. In *NIPS*. 2014.

[37] Jost Tobias Springenberg, Alexey Dosovitskiy, Thomas Brox, and Martin Riedmiller. Striving for simplicity: The all convolutional net. *arXiv:1412.6806 [cs]*, December 2014.

[38] Rupesh Kumar Srivastava, Jonathan Masci, Faustino Gomez, and Jürgen Schmidhuber. Understanding locally competitive networks. In *International Conference on Learning Representations*, 2015.

A Highway Networks Implementation

Notation We use boldface letters for vectors and matrices, and italicized capital letters to denote transformation functions. $\mathbf{0}$ and $\mathbf{1}$ denote vectors of zeros and ones respectively, and $\mathbf{I}$ denotes an identity matrix. The dot operator $(\cdot)$ is used to denote element-wise multiplication.

In most modular and efficient implementations, neural networks are represented as a series of simpler *operations* chained together. Let's assume that some non-linear transformations H, T and C are already defined so that for input $\mathbf{x}$ and some transformation parameters (to be learned) $\mathbf{W_H}, \mathbf{W_T}, \mathbf{W_C}$:

$$\begin{aligned} \mathbf{H} &= H(\mathbf{x}, \mathbf{W_H}), \\ \mathbf{T} &= T(\mathbf{x}, \mathbf{W_T}), \\ \mathbf{C} &= C(\mathbf{x}, \mathbf{W_C}). \end{aligned} \tag{6}$$

Typically H would be an affine transformation followed by a non-linear activation function such as *tanh* or *rectified linear* for a feedforward network, but in general it may take convolutional, recurrent or other forms. Similarly, T and C can take any form but should typically map inputs to values in (0, 1), since they are interpreted as gates.

We define a **Highway** operation simply in terms of $\mathbf{x}, \mathbf{T}, \mathbf{H}$ and $\mathbf{C}$:

$$\mathbf{y} = \mathbf{H} \cdot \mathbf{T} + \mathbf{x} \cdot \mathbf{C}, \tag{7}$$

which essentially implements what's usually called the *forward pass* of the operation using element-wise multiplication and addition operations.

In this paper, we have set $\mathbf{C} = 1 - \mathbf{T}$ for simplicity, giving

$$\mathbf{y} = \mathbf{H} \cdot \mathbf{T} + \mathbf{x} \cdot (\mathbf{1} - \mathbf{T}). \tag{8}$$

Backward pass: The **Highway** operation utilizes no additional parameters on its own, so during backpropagation, only the derivatives of $\mathbf{x}, \mathbf{T}, \mathbf{H}$ need to be computed. These are simply:

$$\begin{aligned} \mathbf{dH} &= \mathbf{T} \cdot \mathbf{dy}, \\ \mathbf{dT} &= (\mathbf{H} - \mathbf{x}) \cdot \mathbf{dy}, \\ \mathbf{dx} &= (\mathbf{1} - \mathbf{T}) \cdot \mathbf{dy}. \end{aligned} \tag{9}$$

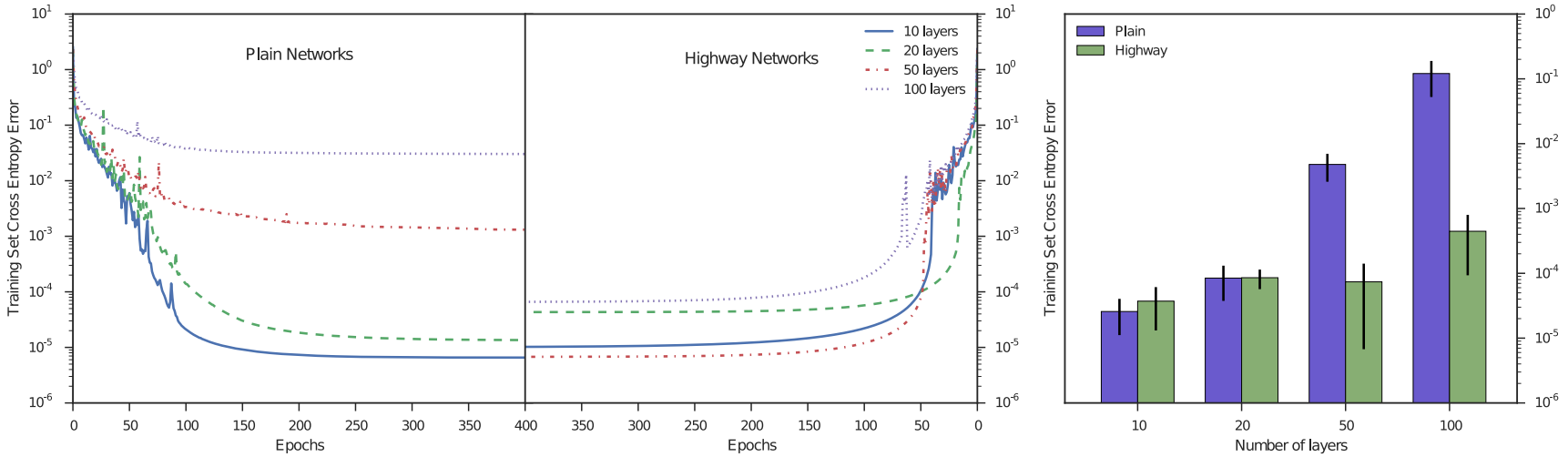

Figure 5: Comparison of optimization of plain networks and highway networks of various depths. All networks were optimized using SGD with momentum. *Left:* The training curves for the best hyperparameter settings obtained for each network depth. *Right:* Mean performance of top 10 (out of 100) hyperparameter settings. Plain networks become much harder to optimize with increasing depth, while highway networks with up to 100 layers can still be optimized well.