Understanding the Disharmony between Dropout and Batch Normalization by Variance Shift

Xiang Li*[1,2], Shuo Chen[1], Xiaolin Hu†[3] and Jian Yang‡[1]

[1]PCALab, Nanjing University of Science and Technology [2]Momenta [3]Tsinghua University

Abstract

This paper first answers the question "why do the two most powerful techniques Dropout and Batch Normalization (BN) often lead to a worse performance when they are combined together in many modern neural networks, but cooperate well sometimes as in Wide ResNet (WRN)?" in both theoretical and empirical aspects. Theoretically, we find that Dropout shifts the variance of a specific neural unit when we transfer the state of that network from training to test. However, BN maintains its statistical variance, which is accumulated from the entire learning procedure, in the test phase. The inconsistency of variances in Dropout and BN (we name this scheme "variance shift") causes the unstable numerical behavior in inference that leads to erroneous predictions finally. Meanwhile, the large feature dimension in WRN further reduces the "variance shift" to bring benefits to the overall performance. Thorough experiments on representative modern convolutional networks like DenseNet, ResNet, ResNeXt and Wide ResNet confirm our findings. According to the uncovered mechanism, we get better understandings in the combination of these two techniques and summarize guidelines for better practices.

1. Introduction

Srivastava et al. [28] brought Dropout as a simple way to prevent neural networks from overfitting. It has been proved to be significantly effective over a large range of machine learning areas, such as image classification [26, 2], speech recognition [9, 5, 3] and even natural language processing [18, 15]. Before the birth of Batch Normalization (BN), it became a necessity of almost all the state-of-the-art networks and successfully boosted their performances against overfitting risks, despite its amazing simplicity.

Ioffe and Szegedy [17] demonstrated BN, a powerful

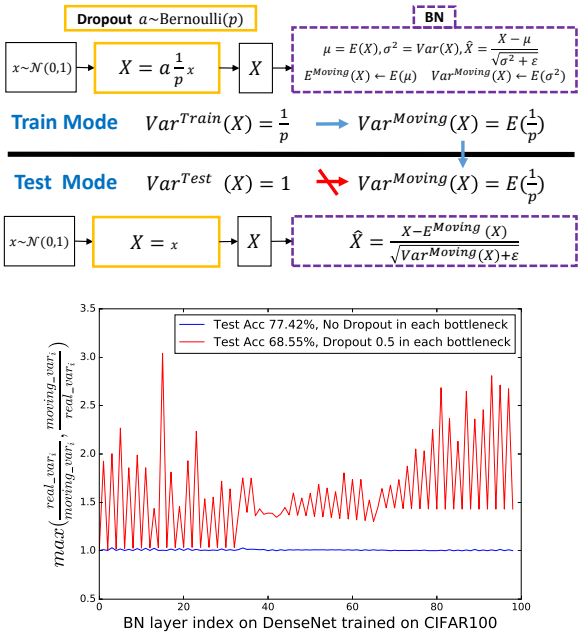

Figure 1. **Up:** a simplified mathematical illustration of "variance shift". In test mode, the neural variance of X is different from that in train mode caused by Dropout, yet BN attempts to treat that variance as the popular statistics accumulated from training. Note that p denotes the Dropout retain ratio and a comes from Bernoulli distribution which has probability p of being 1. **Down:** variance shift in experimental statistics on DenseNet trained on CIFAR100 dataset. The curves are both calculated from the *same training data*. "$moving\_var_i$" is the moving variance (take its mean value instead if it is a vector) that the i-th BN layer accumulates during the entire learning, and "$real\_var_i$" stands for the real variance of neural response before the i-th BN layer in inference.

*Xiang Li, Shuo Chen and Jian Yang are with PCA Lab, Key Lab of Intelligent Perception and Systems for High-Dimensional Information of Ministry of Education, and Jiangsu Key Lab of Image and Video Understanding for Social Security, School of Computer Science and Engineering, Nanjing University of Science and Technology, China. Xiang Li is also a visiting scholar at Momenta. Email: xiang.li.implus@njust.edu.cn

†Xiaolin Hu is with the Tsinghua National Laboratory for Information Science and Technology (TNList) Department of Computer Science and Technology, Tsinghua University, China.

‡Corresponding author.

skill that not only sped up all the modern architectures but also improved upon their strong baselines by acting as regularizers. Therefore, BN has been adopted in nearly all the recent network structures [31, 30, 13, 34] and demonstrates its great practicability and effectiveness.

However, the above two powerful methods always fail to obtain an extra reward when combined together practically [19]. In fact, a modern network even performs worse and unsatisfactorily when it is equipped with BN and Dropout simultaneously in their bottleneck blocks. [17] had already realized that BN eliminates the need for Dropout in some cases, and thus conjectured that BN provides similar regularization benefits as Dropout intuitively. More evidences are provided in recent architectures such as ResNet/PreResNet [10, 11], ResNeXt [32], DenseNet [16], where the best performances are all obtained by BN with the absence of Dropout. Interestingly, a recent study Wide ResNet (WRN) [33] show that it is positive for Dropout to be applied in the WRN's bottleneck blocks with a very large feature dimension. So far, previous clues leave us a mystery about the confusing and complicated relations between Dropout and BN. *Why do they conflict in most of the common modern architectures? Why do they cooperate friendly sometimes as in WRN?*

We discover the key to understand the disharmony between Dropout and BN is the inconsistent behaviors of neural variance [12] during the switch of networks' state. Considering one neural response X as illustrated in Fig. 1, when the state changes from training to test, Dropout will scale the response by its Dropout retain ratio (i.e. p) that actually changes the neural variance as in learning. However, BN still maintains its statistical moving variance of X, as in most of the common Deep Learning toolboxes' (e.g., tensorflow [1], pytorch [24] and mxnet [4]) implementations. This mismatch of variance could lead to a instability (see red curve in Fig. 1). As the signals go deeper, the numerical deviation on the final predictions may amplify, which drops the system's peformance. We name this scheme as "variance shift" for simplicity. Instead, without Dropout in every bottleneck block, the real neural variances in inference appear very closely to the moving ones accumulated by BN (see blue curve in Fig. 1), which is also preserved with a higher test accuracy.

Theoretically, we deduced the "variance shift" under two general conditions in modern networks' bottleneck blocks, and found a satisfied explanation for the aforementioned mystery between Dropout and BN. Furthermore, a large range of experimental statistics from four representative modern convolutional networks (i.e., PreResNet, ResNeXt, DenseNet, Wide ResNet) on CIFAR10/100 datasets verified our findings. Finally, we summarized the understandings based on our theory and experiments, which can serve as guidelines in practice.

2. Related Work and Preliminaries

Dropout [28] can be interpreted as a way of regularizing a neural network by adding noise to its hidden units. Specifically, it involves multiplying hidden activations by Bernoulli distributed random variables which take the value 1 with probability p ($0 \leq p \leq 1$) and 0 otherwise. Importantly, the test scheme is quite different from training. During training, the information flow goes through the dynamic sub-network. In the test phase, the neural responses are scaled by the Dropout retain ratio. In order to approximate an equally weighted geometric mean of the predictions of an exponential number of learned models that share parameters. Consider a feature vector $\boldsymbol{x} = (x_1 \dots x_d)$ with channel dimension d, $x_k = a_k x_k (k = 1 \dots d)$ during the training phase if we apply Dropout on $\boldsymbol{x}$, where $a_k \sim P$ comes from the Bernoulli distribution [7]:

$$P(a_k) = \begin{cases} 1-p, & a_k = 0 \\ p, & a_k = 1 \end{cases}, \tag{1}$$

and $\boldsymbol{a} = (a_1 \dots a_d)$ is a vector of *independent* Bernoulli random variables. At test time for Dropout, one should scale down the weights by multiplying them by a factor of p. As introduced in [28], another way to achieve the same effect is to scale up the retained activations by multiplying by $\frac{1}{p}$ at training time and not modifying the weights at test time. It is more popular on practical implementations, thus we employ this formula of Dropout in both analyses and experiments. Therefore, the hidden activation in the training phase is: $\widehat{x}_k = a_k \frac{1}{p} x_k$, whilst in inference it becomes simple like: $\widehat{x}_k = x_k$.

Batch Normalization (BN) [17] proposes a deterministic information flow by normalizing each neuron into zero mean and unit variance. Considering values of x (for clarity, $x \equiv x_k$) over a mini-batch: $\mathcal{B} = \{x^{(1)\dots(m)}\}$ with m instances, we have the form of "normalize" part:

$$\mu = \frac{1}{m}\sum_{i=1}^{m} x^{(i)}, \sigma^2 = \frac{1}{m}\sum_{i=1}^{m}(x^{(i)} - \mu)^2, \widehat{x}^{(i)} = \frac{x^{(i)} - \mu}{\sqrt{\sigma^2 + \epsilon}}, \tag{2}$$

where μ and σ^2 participate in the backpropagation. Note that we do not consider the "scale and shift" part in BN because the key of "variance shift" exists in its "normalize" part. The normalization of activations that depends on the mini-batch allows efficient training, but is neither necessary nor desirable during inference [25]. Therefore, BN accumulates the moving averages of neural means and variances during learning to track the accuracy of a model as it trains:

$$E^{Moving}(x) \leftarrow E_{\mathcal{B}}(\mu), Var^{Moving}(x) \leftarrow E'_{\mathcal{B}}(\sigma^2), \tag{3}$$

where $E_{\mathcal{B}}(\mu)$ denotes the expectation of μ from multiple training mini-batches $\mathcal{B}$ and $E'_{\mathcal{B}}(\sigma^2)$ denotes the expectation of the unbiased variance estimate (i.e., $\frac{m}{m-1} \cdot E_{\mathcal{B}}(\sigma^2)$)

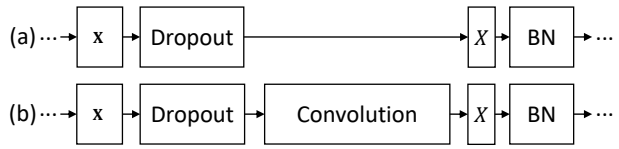

Figure 2. Two general cases for analyzing variance shift in modern networks' bottleneck blocks.

over multiple training mini-batches. They are all obtained by implementations of moving averages [17] and are fixed for linear transform during inference:

$$\widehat{x} = \frac{x - E^{Moving}(x)}{\sqrt{Var^{Moving}(x) + \epsilon}}. \tag{4}$$

3. Theoretical Analyses

From the preliminaries, one can notice that Dropout only ensures an "equally weighted geometric **mean** of the predictions of an exponential number of learned models" by the approximation from its test policy, as introduced in the original paper [28]. This scheme poses the variance of the hidden units unexplored in a Dropout model. Therefore, the central idea is to investigate the variance of the neural response before a BN layer, where the Dropout is previously applied. Following [8], we first start by studying the linear regime. Further, if a Dropout layer is applied after the last BN layer in this bottleneck block, it will be followed by the first BN layer in the next bottleneck block. Therefore, we only need to consider the cases where Dropout comes before BN. Meanwhile, we also need to consider the number of convolutional layers between Dropout and BN. 0 or 1 convolutional layer is obviously necessary for investigations, yet 2 or more convolutional layers can be attributed to the 1 case via similar analyses. To conclude, we have two cases generally, as shown in Fig. 2. *Importantly, the Wide ResNet with Dropout exactly follows the case (b) formulation.*

In case (a), the BN layer is directly subsequent to the Dropout layer and we only need to consider one neural response $X = a_k \frac{1}{p} x_k$, where $k = 1 \dots d$ in training phase and $X = x_k$ in test phase.

In case (b), the feature vector $\boldsymbol{x} = (x_1 \dots x_d)$ is passed into a convolutional layer (similar deduction can be conducted here if it is a fully connected layer) to form the neural response X. We also regard its corresponding weights to be $\boldsymbol{w} = (w_1 \dots w_d)$, hence we get $X = \sum_{i=1}^{d} w_i a_i \frac{1}{p} x_i$ for training and $X = \sum_{i=1}^{d} w_i x_i$ for testing.

For the ease of deduction, we assume that the inputs all come from the same distribution with mean c and variance v (i.e., $E(x_i) = c, Var(x_i) = v, v > 0$ for any $i = 1 \dots d$). We let the a_i and x_i be mutually independent, considering the property of Dropout. Due to the aforementioned definition, a_i and a_j are mutually independent as well.

3.1. Case (a)

By using the definition of the variance and following the paradigms above, we have that

$$\begin{aligned} &Var^{Train}(X) \\ &= \frac{1}{p^2} E(a_k^2) E(x_k^2) - \frac{1}{p^2} (E(a_k) E(x_k))^2 = \frac{1}{p}(c^2 + v) - c^2. \end{aligned} \tag{5}$$

In inference, BN keeps the moving average of variance (i.e., $E'_{\mathcal{B}}(\frac{1}{p}(c^2 + v) - c^2)$) fixed. That is, *BN wishes that the variance of neural response X, which comes from the input images initially, is supposed to be close to* $E'_{\mathcal{B}}(\frac{1}{p}(c^2 + v) - c^2)$. However, Dropout breaks the harmony when it comes to its test stage by having $X = x_k$ to get $Var^{Test}(X) = Var(x_k) = v$.

If putting $Var^{Test}(X)$ into the unbiased variance estimate, it becomes $E'_{\mathcal{B}}(v)$ which is obviously different from the popular statistic $E'_{\mathcal{B}}(\frac{1}{p}(c^2 + v) - c^2)$ of BN during training when Dropout ($p < 1$) is applied. Therefore, the shift ratio $\triangle$ is obtained by

$$\triangle(p) = \frac{Var^{Test}(X)}{Var^{Train}(X)} = \frac{v}{\frac{1}{p}(c^2 + v) - c^2}. \tag{6}$$

In case (a), the variance shift happens via a coefficient $\triangle(p) \leq 1$. Since modern neural networks carry a deep feedforward topologic structure, the deviate numerical manipulations can lead to more uncontrollable numerical outputs of subsequent layers (Fig. 1). It brings the chain reaction of amplified shift of variances (even affects the means further) in every BN layers sequentially, as the networks go deeper. We will show that it directly leads to a dislocation of final predictions and makes the system suffer from a performance drop later in the statistical experimental part (e.g., Figs. 4 and 5 in Section 4).

In this design (i.e., BN directly follows Dropout), if we want to alleviate the variance shift risks, i.e., $\triangle(p) \to 1$, the only thing we can do is to eliminate Dropout which means setting the Dropout retain ratio $p \to 1$. Fortunately, the architectures where Dropout brings benefits (e.g., in Wide ResNet) do not follow this type of arrangement. In fact, they adopt the case (b) in Fig. 2, which is more common in practice, and we will describe it in details as follows.

3.2. Case (b)

At this time, X is obtained by $\sum_{i=1}^{d} w_i a_i \frac{1}{p} x_i$ during training, where $\boldsymbol{w}$ denotes for the corresponding weights for $\boldsymbol{x}$, along with the Dropout applied. For the ease of deduction, we assume that in the very later epoch of training, the weights of $\boldsymbol{w}$ remains constant, giving that the gradients become significantly close to zero. Similarly, we can

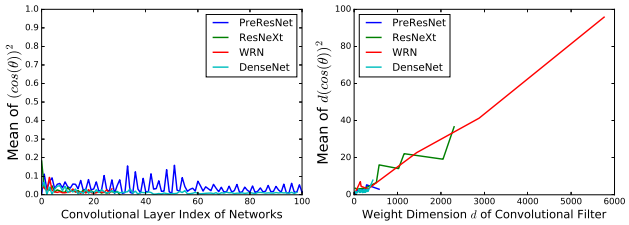

Figure 3. Statistical mean values of $(\cos\theta)^2$ and $d(\cos\theta)^2$. These four modern architectures are trained without Dropout on CIFAR100, respectively. We observe that $(\cos\theta)^2$ lies in $(0.01, 0.10)$ approximately in every network structure and various datasets. Interestingly, the term $d(\cos\theta)^2$ in WRN is significantly bigger than those on other networks mainly due to its larger channel width d.

expand $Var^{Train}(X)$ as:

$$\begin{aligned} Var^{Train}(X) &= Cov(\sum_{i=1}^{d} w_i a_i \frac{1}{p} x_i, \sum_{i=1}^{d} w_i a_i \frac{1}{p} x_i) \\ &= (\frac{1}{p}(c^2+v) - c^2)(\sum_{i=1}^{d} w_i^2 + \rho^{ax} \sum_{i=1}^{d} \sum_{j \neq i}^{d} w_i w_j), \end{aligned} \tag{7}$$

where $\rho_{i,j}^{ax} = \frac{Cov(a_i x_i, a_j x_j)}{\sqrt{Var(a_i x_i)}\sqrt{Var(a_j x_j)}} \in [-1, 1]$. For the ease of deduction, we simplify all the linear correlation coefficients to be the same as a constant $\rho^{ax} = \rho_{i,j}^{ax}, \forall i, j = 1 \dots d, i \neq j$. Similarly, $Var^{Test}(X)$ is obtained by

$$\begin{aligned} Var^{Test}(X) &= Cov(\sum_{i=1}^{d} w_i x_i, \sum_{i=1}^{d} w_i x_i) \\ &= v(\sum_{i=1}^{d} w_i^2 + \rho^{x} \sum_{i=1}^{d} \sum_{j \neq i}^{d} w_i w_j), \end{aligned} \tag{8}$$

where $\rho_{i,j}^{x} = \frac{Cov(x_i, x_j)}{\sqrt{Var(x_i)}\sqrt{Var(x_j)}} \in [-1, 1]$, and we also have a constant $\rho^{x} = \rho_{i,j}^{x}, \forall i, j = 1 \dots d$ and $i \neq j$. Since a_i and x_i, a_i and a_j are mutually independent, we can get the relation between ρ^{ax} and ρ^{x}:

$$\begin{aligned} \rho^{ax} = \rho_{i,j}^{ax} &= \frac{Cov(a_i x_i, a_j x_j)}{\sqrt{Var(a_i x_i)}\sqrt{Var(a_j x_j)}} \\ &= \frac{v}{\frac{1}{p}(c^2+v) - c^2} \rho_{i,j}^{x} = \frac{v}{\frac{1}{p}(c^2+v) - c^2} \rho^{x}. \end{aligned} \tag{9}$$

According to Eqs. (7), (8) and (9), the variance shift for case (b) can be written as:

$$\begin{aligned} \triangle(p, d) &= \frac{Var^{Test}(X)}{Var^{Train}(X)} \\ &= \frac{v + v\rho^{x}(d(\cos\theta)^2 - 1)}{\frac{1}{p}(c^2+v) - c^2 + v\rho^{x}(d(\cos\theta)^2 - 1)}, \end{aligned} \tag{10}$$

where $(\cos\theta)^2$ comes from the expression:

Table 1. Averaged means of $(\cos\theta)^2$ and $d(\cos\theta)^2$ over all the convolutional layers on four representative networks.

Networks	CIFAR10		CIFAR100	
	$(\cos\theta)^2$	$d(\cos\theta)^2$	$(\cos\theta)^2$	$d(\cos\theta)^2$
PreResNet-110 [11]	0.03546	2.91827	0.03169	2.59925
ResNeXt-29 [32]	0.02244	14.78266	0.02468	14.72835
WRN-28-10 [33]	0.02292	52.73550	0.02118	44.31261
DenseNet-BC [16]	0.01538	3.83390	0.01390	3.43325

$$\frac{(\sum_{i=1}^{d} w_i)^2}{d \cdot \sum_{i=1}^{d} w_i^2} = (\frac{\sum_{i=1}^{d} 1 \cdot w_i}{\sqrt{\sum_{i=1}^{d} 1^2}\sqrt{\sum_{i=1}^{d} w_i^2}})^2 = (\cos\theta)^2, \tag{11}$$

and θ denotes for the angle between vector $\boldsymbol{w}$ and vector $(1 \dots 1) \in R^d$. To empirically prove that $d(\cos\theta)^2$ scales approximately linear to d, here we made rich calculations w.r.t the terms $d(\cos\theta)^2$ and $(\cos\theta)^2$ on four modern architectures[1] trained on CIFAR10/100 datasets (Table 1 and Fig. 3.2). Based on Table 1 and Fig. 3.2, we observe that $(\cos\theta)^2$ lies in $(0.01, 0.10)$ stably in every type of the network whilst $d(\cos\theta)^2$ tends to increase in parallel when d grows. From Eq. (10), the inequation $\frac{Var^{Test}(X)}{Var^{Train}(X)} < 1$ holds obviously when $p < 1$. If we want $Var^{Test}(X)$ to be close with $Var^{Train}(X)$, we need this term

$$\triangle(p, d) = \frac{v\rho^{x} + \frac{v - v\rho^{x}}{\boldsymbol{d}(\cos\theta)^2}}{v\rho^{x} + \frac{\frac{1}{p}v - v\rho^{x} + (\frac{1}{p} - 1)c^2}{\boldsymbol{d}(\cos\theta)^2}} = \frac{v\rho^{x}(d(\cos\theta)^2 - 1) + v}{v\rho^{x}(d(\cos\theta)^2 - 1) + \frac{1}{\boldsymbol{p}}(c^2 + v) - c^2} \tag{12}$$

to approach 1. There are ***two ways*** to achieve $\triangle(p, d) \to 1$:

- $p \to 1$: maximizing the Dropout retain ratio p (ideally up to 1 which means Dropout is totally eliminated);
- $d \to \infty$: growing the width of channel exactly as the Wide ResNet did to enlarge d.

4. Statistical Experiments

We conduct extensive statistical experiments to check the correctness of above deduction in this section. Four modern architectures including DenseNet [16], PreResNet [11], ResNeXt [32] and Wide ResNet (WRN) [33] are adopted on CIFAR10 and CIFAR100 datasets.

Datasets. The two CIFAR datasets [20] consist of colored natural scene images, with 32×32 pixel each. The training and test sets contain 50k images and 10k images respectively. CIFAR10 (C10) has 10 classes and CIFAR100 (C100) has 100. For data preprocessing, we normalize the data by using the channel means and standard deviations. For data augmentation, we adopt a standard scheme that is widely used in [11, 16, 21, 23, 22, 27, 29]: the images are first zero-padded with 4 pixels on each side, then a 32×32

[1]For the convolutional filters which have larger than 1 filter size as $k \times k, k > 1$, we vectorise them by expanding its channel width d to $d \times k \times k$ while maintaining all the weights.

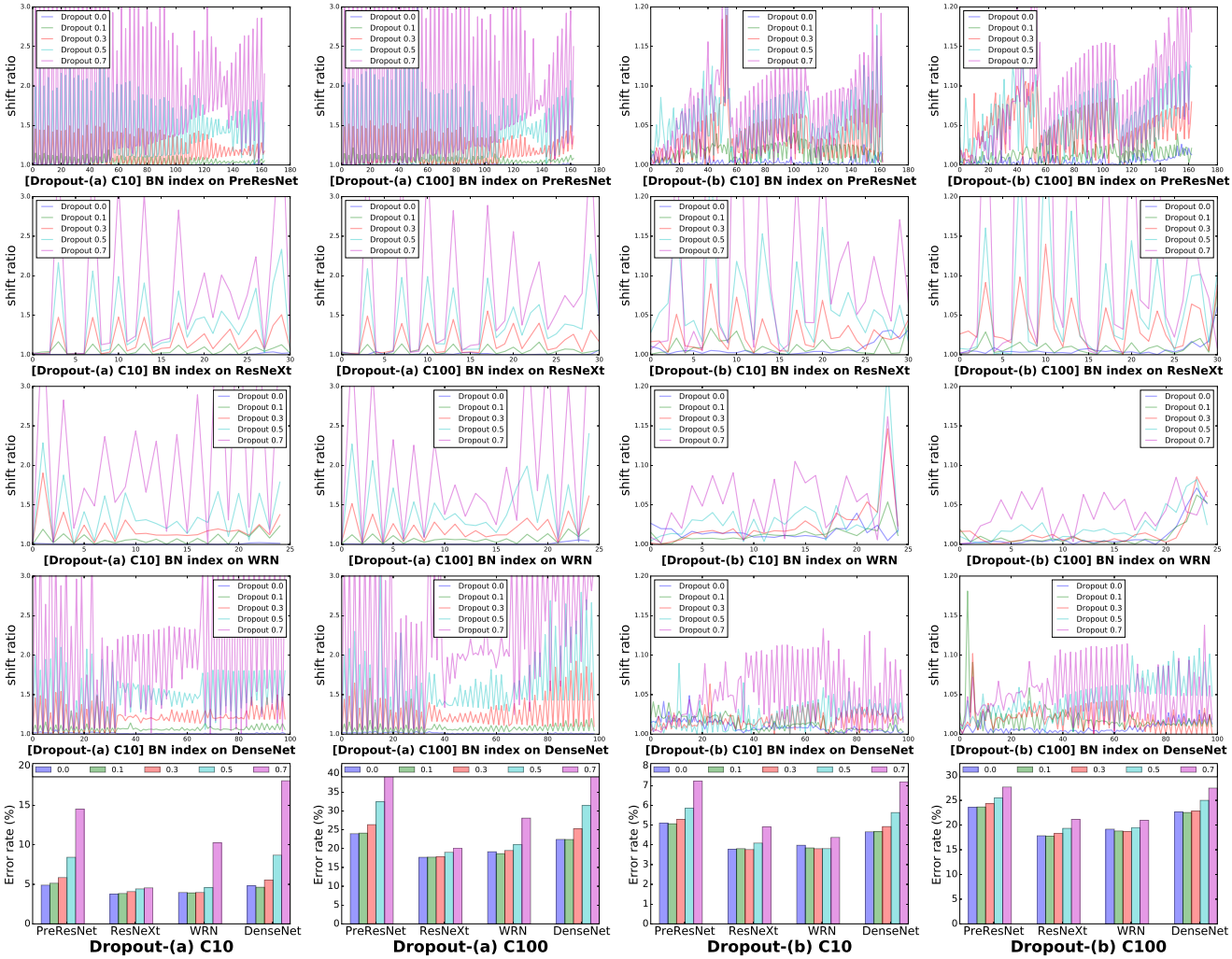

Figure 4. *See by columns.* Visualizations about "variance shift" on BN layers of four modern networks w.r.t: 1) Dropout type; 2) Dropout drop ratio; 3) dataset, along with their test error rates (the 5th row). Obviously, WRN is less influenced by Dropout (e.g., *in 3rd row and 4th column*) when the Dropout-(b) drop ratio ≤ 0.5, and thus it even enjoys an improvement with Dropout applied with BN in each bottleneck.

crop is randomly sampled from them and half of the images are horizontally flipped.

Networks with Dropout. The four modern architectures are all chosen from the open-source codes written in pytorch that can reproduce the results reported in previous papers. Specifically, there are PreResNet-110 [11], ResNeXt-29, 8 $\times$ 64 [32], WRN-28-10 [33] and DenseNet-BC (L=100, k=12) [16]. Since the BN layers are already developed as the indispensible components of their body structures, *we arrange Dropout that follows the two cases in Fig. 2:*

(a) We assign all the Dropout layers only and right before all the bottlenecks' last BN layers in these four networks, neglecting their possible Dropout implementations (as in DenseNet [16] and Wide ResNet [33]). We denote this design to be models of **Dropout-(a)**.

(b) We follow the assignment of Dropout in Wide ResNet [33], which finally improves WRNs' overall performances, to place the Dropout before the last Convolutional layer in every bottleneck block of PreResNet, ResNeXt and DenseNet. This scheme is denoted as **Dropout-(b)** models.

Statistics of variance shift. Assume a network $\mathcal{G}$ contains n BN layers in total. We arrange these BN layers from shallow to deep by giving them indices that range from 1 to n accordingly. The whole statistical manipulation is conducted by the following three steps:

(1) Calculate *moving_var*$_i, i \in \{1, ..., n\}$: when $\mathcal{G}$ is trained until convergence, each BN layer obtains the moving average of neural variance (the unbiased variance estimate) from the feature-map that it receives during the entire learning procedure. We denote that variance as *moving_var*. Since the *moving_var* for every BN layer is a vector (whose length is equal to the amount of channels of previous

Table 2. Averaged shift ratios over all the BN layers of Dropout-(b) models on CIFAR100 dataset. Smaller is better for stability.

Dropout ratios:	0.1	0.3	0.5	0.7
PreResNet-110 [11]	1.008945	1.040766	1.052092	1.076403
ResNeXt-29 [32]	1.006296	1.032514	1.058549	1.134871
WRN-28-10 [33]	**1.003485**	**1.006466**	**1.013873**	**1.033254**
DenseNet-BC [16]	1.013859	1.015065	1.036019	1.042925

feature-map), we leverage its mean value to represent *moving_var* instead, in purpose of an ease visualization. Further, we denote $moving\_var_i$ as the *moving_var* of i-th BN layer.

(2) Calculate $real\_var_i, i \in \{1, ..., n\}$: after training, we fix all the parameters of $\mathcal{G}$ and set its state to the evaluation mode (hence the Dropout will apply its inference policy and BN will freeze its moving averages of means and variances). The training data is again utilized for going through $\mathcal{G}$ within a certain of epochs, in order to get the real expectation of neural variances on the feature-maps before each BN layer. Data augmentation is also kept to ensure that every possible detail for calculating neural variances remains exactly the same with training. Importantly, we adopt the same moving average algorithm to accumulate the unbiased variance estimates. Similarly in (1), we let the mean value of real variance vector be $real\_var_i$ before the i-th BN layer.

(3) Get "shift ratio" $= \boldsymbol{max}(\frac{real\_var_i}{moving\_var_i}, \frac{moving\_var_i}{real\_var_i}), i \in [1, n]$: since we focus on the shift, the scalings are all kept above 1 by their reciprocals if possible in purpose of a better view. Various Dropout drop ratios $[0.0, 0.1, 0.3, 0.5, 0.7]$ are applied for comparisons in Fig. 4. The corresponding error rates are also included in each column. To be specific, we also calculate all the averaged shift ratios over the entire networks under drop ratio $0.1, 0.3, 0.5, 0.7$ to show the quantitive analyses based on Fig. 4 in Table 2. The results demonstrate that WRNs' shift ratios are considerably smaller than other counterparts in every Dropout setting.

The statistical experiments confirm our analyses. In these four columns of Fig. 4, we discover that when the drop ratio is relatively small (i.e., 0.1), the green curves go close to the blue ones (i.e., models without Dropout), thus their performances are comparable or even better to the baselines. It agrees with our previous deduction that whenever in (a) or (b) case, decreasing drop ratio $1 - p$ will alleviate the variance shift risks. Furthermore, in Dropout-(b) models (i.e., the last two columns) we find that, for WRNs, the curves with drop ratio $0.1, 0.3$ even 0.5 approach closer to the one with 0.0 than other networks, and they all outperform the baseline. It also aligns with our analyses since WRN has a significantly larger channel dimension d, and it ensures that a slightly larger p will not explode the neural variance too much. Furthermore, the statistics on Table 2 also support our previous deduction that WRN is less influenced by Dropout in terms of variance shift ratio, and its performance consistently improves when drop ratio < 0.5, whilst other models get stucked or perform even worse when drop ratio reaches 0.3 (last row in Fig. 4).

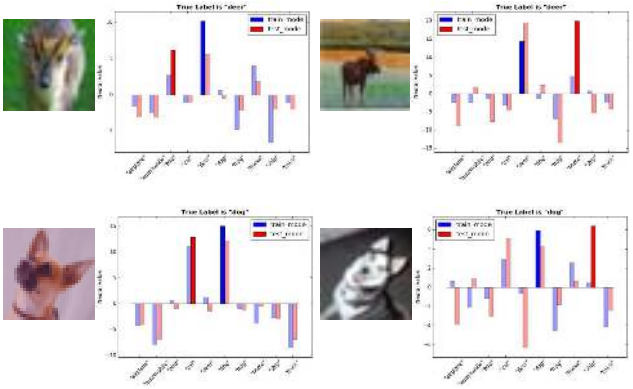

Figure 5. Examples of inconsistent neural responses between train mode and test mode of DenseNet Dropout-(a) 0.5 trained on CIFAR10 dataset. These samples are from the training data, whilst they are correctly classified by the model during learning yet erroneously judged in inference, despite all the fixed model parameters. Variance shift finally leads to the prediction shift that drops the performance.

Neural responses (of the last layer before softmax) for training data are unstable from training stage to test stage. To get a clearer understanding of the numerical disturbance that the variance shift brings finally, a bundle of images (from training data) are drawn with their neural responses before the softmax layer in both training stage and test stage (Fig. 5). From those pictures and their responses, we can find that with all the weights of networks fixed, only a mode transfer (from train to test) will change the distribution of the final responses even in the training set, and it leads to a wrong classification consequently. It proves that the predictions of training data differs between training stage and test stage when a network is equipped with Dropout and BN layers in their bottlenecks. Therefore, we confirm that the unstable numerical behaviors are the fundamental reasons for the performance drop.

Only an adjustment for moving means and variances will bring an improvement, despite all other parameters fixed. Given that the moving means and variances of BN will not match the real ones during test, we attempt to adjust these values by passing the training data again under the evaluation mode. In this way, the moving average algorithm [17] can also be applied. After shifting the moving statistics to the real ones by using the training data, we can have the model performed on the test set. From Table 3, All the Dropout-(a)/(b) 0.5 models outperform their baselines by having their moving statistics adjusted. Significant improvements (e.g., ~ 2 and ~ 4.5 gains for DenseNet on CIFAR10 and on CIFAR100 respectively) can be observed in Dropout-(a) models. It again verifies that the drop of performance could be attributed to the "variance shift": a more

Table 3. Adjust BN's moving mean/variance by running moving average algorithm on training data under test mode. These error rates (%) are all averaged from 5 parallel runnings with different random initial seeds. "-A" means the corresponding adjustment. For comparisons, we also list the performances of these models without Dropout. The best records are marked red.

C10	Dropout-(a)		Dropout-(b)		w/o Dropout
	0.5	0.5-A	0.5	0.5-A	
PreResNet	8.42	**6.42**	5.85	**5.77**	**5.02**
ResNeXt	4.43	**3.96**	4.09	**3.93**	**3.77**
WRN	4.59	**4.20**	3.81	**3.71**	3.97
DenseNet	8.70	**6.82**	5.63	**5.29**	**4.72**
C100	Dropout-(a)		Dropout-(b)		w/o Dropout
	0.5	0.5-A	0.5	0.5-A	
PreResNet	32.45	**26.57**	25.50	**25.20**	**23.73**
ResNeXt	19.04	**18.24**	19.33	**19.09**	**17.78**
WRN	21.08	**20.70**	19.48	**19.15**	19.17
DenseNet	31.45	**26.98**	25.00	**23.92**	**22.58**

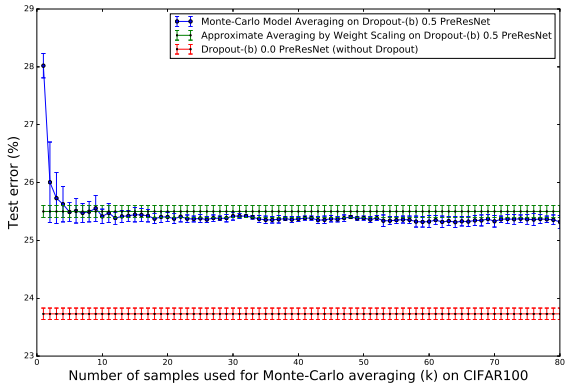

Figure 6. Monte-Carlo model averaging vs. weight scaling vs. no Dropout. The ensemble of models which avoid "variance shift" risks still underperforms the baseline trained without Dropout.

proper popular statistics with *smaller* variance shift could recall a bundle of erroneously classified samples back to the right ones. However, except for WRN, the performances of other architectures after adjusting statistics still *underperform* their counterparts without Dropout. This cue shows that for most structures, shifting moving statistics via training data can not make up for the performance gap.

Although Monte-Carlo model averaging can avoid "variance shift", it costs plenty of time and limits the performance.. The efficient test time procedure that the original Dropout [28] propose is to do an approximate model combination by scaling down the weights of the trained neural network. And it is exactly the central reason which is responsible for the variance shift risks, as it only ensures the stability of neural means, rather the variances. Therefore, a natural question comes out: what if we try to make predictions by sampling k neural nets using Dropout for each test case and average their predictions? Theoretically, applying Dropout in the test phase will avoid the "variance shift" yet slightly harm the performance. Although it is shown very expensive in [28], we are still interesting how many samples networks are needed to match the performance of the approximate averaging method or the baseline models without Dropout. Here we take the Dropout-(b) 0.5 PreResNet model as an example and do classification on CIFAR100 by averaging the predictions of k randomly sampled neural networks.

From Fig. 6, we can find that nearly 10 samples of networks can approach the results of weight scaling. And more rounds of runnings will give a slight gain in the end but can not reach the performance of the baseline without Dropout. To conclude, these sampled networks still cannot compensate the performance drop with such an expensive way in the test phase.

5. Strategy to Combine Them Better

Since we get a clear knowledge about the disharmony between Dropout and BN, we can easily develop an approach to combine them together, to see whether an extra improvement can be obtained. In this section, we introduce one possible solution that slightly modifies the formula of Dropout and make it less sensitive to variance, which can alleviate the shift problem and stabilize the numerical behaviors.

The drawbacks of vanilla Dropout lie in the weight scale during the test phase, which may lead to a large disturbance on statistical variance. This clue can push us to think: if we find a scheme that functions like Dropout but carries a lighter variance shift, we may stabilize the numerical behaviors of neural networks, thus the final performance will probably benefit from such stability. Here we take the case (a) as an example for investigations where the variance shift rate is $\frac{v}{\frac{1}{p}(c^2+v)-c^2} = p$ (we let $c = 0$ for simplicity in this discussion). That is, if we set the drop ratio $(1-p)$ as 0.1, the variance would be scaled by 0.9 when the network is switched from training to test. Inspired by the original Dropout [28], where the authors also proposed another form of Dropout that amounts to adding a Gaussian distributed random variable with zero mean and standard deviation equal to the activation of the unit, i.e., $x_i + x_i r$ and $r \sim \mathcal{N}(0,1)$, we further modify r as a uniform distribution that lies in $[-\beta, \beta]$, where $0 \leq \beta \leq 1$. Therefore, each hidden activation would be $X = x_i + x_i r_i$ and $r_i \sim \mathcal{U}(-\beta, \beta)$ [6]. We name this form of Dropout as "Uout" for simplicity. With the mutually independent distribution between x_i and r_i hold, we apply $X = x_i + x_i r_i, r_i \sim \mathcal{U}(-\beta, \beta)$ in training mode and $X = x_i$ in test. Similarly, in the simplified case

Table 4. Apply new form of Dropout (i.e. Uout) in Dropout-(b) models. These error rates (%) are all averaged from 5 parallel runnings with different random initial seeds. The numbers in brackets denote the values of β relating to the performances.

C10 β	0.0	[0.2, 0.3, 0.5]
PreResNet	5.02	**4.85** (0.2)
ResNeXt	3.77	**3.75** (0.3)
WRN	3.97	**3.79** (0.5)
DenseNet	4.72	**4.61** (0.5)
C100 β	0.0	[0.2, 0.3, 0.5]
PreResNet	23.73	**23.53** (0.3)
ResNeXt	17.78	**17.72** (0.2)
WRN	19.17	**18.87** (0.5)
DenseNet	22.58	**22.30** (0.5)

of $c = 0$, we can deduce the variance shift again as follows:

$$\frac{Var^{Test}(X)}{Var^{Train}(X)} = \frac{Var(x_i)}{Var(x_i + x_i r_i)} = \frac{v}{E((x_i + x_i r_i)^2)} = \frac{v}{E(x_i^2) + 2E(x_i^2)E(r_i) + E(x_i^2)E(r_i^2)} = \frac{3}{3 + \beta^2}. \tag{13}$$

Given β as 0.1, the new variance shift rate would be $\frac{300}{301} \approx 0.9966777$ which is much closer to 1.0 than the previous 0.9 in case (a). A list of experiments is hence employed based on those four modern networks under Dropout-(b) settings in Table 4. We search β in range of $[0.2, 0.3, 0.5]$ to find optimal results. We observe that "Uout" with larger ratios tends to perform favorably well, which indicates its superior stability. Except for ResNeXt, nearly all the architectures achieved up to $0.2 \sim 0.3$ increase of accuracy on both CIFAR10 and CIFAR100 dataset.

Beyond Uout, we discover that adding only one Dropout layer right before the *softmax* layer can avoid the variance shift risks since there are no following BN layers. We evaluate several state-of-the-art models on the ImageNet validation set (Table 5), and observe consistent improvements when drop ratio 0.2 is employed after the last BN layers on the large scale dataset. The benefits of doing so also confirm the effectiveness of our theory.

ImageNet drop ratio	top-1 err.		top-5 err.	
	0.0	0.2	0.0	0.2
ResNet-200 [10]	21.70	**21.48**	5.80	**5.55**
ResNeXt-101 [32]	20.40	**20.17**	5.30	**5.12**

Table 5. Error rates (%) on ImageNet validation set.

6. Summary of Guidelines

According to the analyses and experiments, we can get the following understandings as guidelines:

- In modern CNN architectures, the original Dropout and BN are not recommended to appear in the bottleneck part due to their variance shift conflict, except that we have a relatively large feature dimension. We also suggest the drop ratio < 0.5 since the deduction Eq. (12) and the experiments (Fig. 4) show higher drop ratio will still break the stability of neural responses in any case. To conclude, the shift risk depends on both the Dropout ratio and feature dimension.
- Adjusting the moving means and variances through training data is beneficial for improvements, but it can not compensate the entire loss in performance, compared to the baselines which are trained without Dropout. Moreover, the ensemble of predictions from networks which apply Dropout during test to avoid "variance shift" still underperforms these baselines.
- We understand why some recent models (e.g. Inception-v4 [30], SENet [14]) have adopted one Dropout layer after the last BN layer of the entire network, because it will not lead to the variance shift essentially based on our theory.
- We also discover that the form of Dropout can be modified, in purpose of reducing their variance shift to boost their performances even when they are in the bottleneck building blocks.

7. Conclusion

In this paper, we investigate the "variance shift" phenomenon when Dropout layers are applied with Batch Normalization on modern convolutional networks. We discover that due to their distinct test policies, neural variance will be improper and shifted as the information flows in inference, and it leads to the unexpected final predictions that drops the performance. These understandings can serve as practical guidelines for designing novel regularizers or getting better practices in the area of Deep Learning.

Acknowledgments

The authors would like to thank the editor and the anonymous reviewers for their critical and constructive comments and suggestions. This work was supported by the National Science Fund of China under Grant No. U1713208, Program for Changjiang Scholars and National Natural Science Foundation of China under Grant No. 61836014. It was also supported by NSF of China (No: 61602246), NSF of Jiangsu Province (No: BK20171430), the Fundamental Research Funds for the Central Universities (No: 30918011319), the open project of State Key Laboratory of Integrated Services Networks (Xidian University, ID: ISN19-03), the Summit of the Six Top Talents Program (No: DZXX-027), and the Young Elite Scientists Sponsorship Program by CAST (No: 2018QNRC001).

References

[1] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, et al. Tensorflow: A system for large-scale machine learning. In *OSDI*, volume 16, pages 265–283, 2016.

[2] Z. Akata, S. Reed, D. Walter, H. Lee, and B. Schiele. Evaluation of output embeddings for fine-grained image classification. In *CVPR*, pages 2927–2936, 2015.

[3] D. Amodei, S. Ananthanarayanan, R. Anubhai, J. Bai, E. Battenberg, C. Case, J. Casper, B. Catanzaro, Q. Cheng, G. Chen, et al. Deep speech 2: End-to-end speech recognition in english and mandarin. In *ICML*, pages 173–182, 2016.

[4] T. Chen, M. Li, Y. Li, M. Lin, N. Wang, M. Wang, T. Xiao, B. Xu, C. Zhang, and Z. Zhang. Mxnet: A flexible and efficient machine learning library for heterogeneous distributed systems. *arXiv preprint arXiv:1512.01274*, 2015.

[5] G. E. Dahl, D. Yu, L. Deng, and A. Acero. Context-dependent pre-trained deep neural networks for large-vocabulary speech recognition. *IEEE Transactions on audio, speech, and language processing*, 20(1):30–42, 2012.

[6] J. Friedman, T. Hastie, and R. Tibshirani. *The elements of statistical learning*, volume 1. Springer series in statistics New York, NY, USA:, 2001.

[7] Y. Gal, J. Hron, and A. Kendall. Concrete dropout. In *NeurIPs*, pages 3581–3590, 2017.

[8] X. Glorot and Y. Bengio. Understanding the difficulty of training deep feedforward neural networks. In *ICAIS*, pages 249–256, 2010.

[9] A. Hannun, C. Case, J. Casper, B. Catanzaro, G. Diamos, E. Elsen, R. Prenger, S. Satheesh, S. Sengupta, A. Coates, et al. Deep speech: Scaling up end-to-end speech recognition. *arXiv preprint arXiv:1412.5567*, 2014.

[10] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *CVPR*, pages 770–778, 2016.

[11] K. He, X. Zhang, S. Ren, and J. Sun. Identity mappings in deep residual networks. In *ECCV*, pages 630–645, 2016.

[12] D. Hendrycks and K. Gimpel. Adjusting for dropout variance in batch normalization and weight initialization. 2017.

[13] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017.

[14] J. Hu, L. Shen, and G. Sun. Squeeze-and-excitation networks. *arXiv preprint arXiv:1709.01507*, 2017.

[15] R. Hu, H. Xu, M. Rohrbach, J. Feng, K. Saenko, and T. Darrell. Natural language object retrieval. In *CVPR*, pages 4555–4564, 2016.

[16] G. Huang, Z. Liu, K. Q. Weinberger, and L. van der Maaten. Densely connected convolutional networks. *arXiv preprint arXiv:1608.06993*, 2016.

[17] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *ICML*, pages 448–456, 2015.

[18] Y. Kim, Y. Jernite, D. Sontag, and A. M. Rush. Character-aware neural language models. In *AAAI*, pages 2741–2749, 2016.

[19] G. Klambauer, T. Unterthiner, A. Mayr, and S. Hochreiter. Self-normalizing neural networks. In *NeurIPs*, pages 971–980, 2017.

[20] A. Krizhevsky and G. Hinton. Learning multiple layers of features from tiny images. 2009.

[21] G. Larsson, M. Maire, and G. Shakhnarovich. Fractalnet: Ultra-deep neural networks without residuals. *arXiv preprint arXiv:1605.07648*, 2016.

[22] C.-Y. Lee, S. Xie, P. Gallagher, Z. Zhang, and Z. Tu. Deeply-supervised nets. In *ICAIS*, pages 562–570, 2015.

[23] M. Lin, Q. Chen, and S. Yan. Network in network. *arXiv preprint arXiv:1312.4400*, 2013.

[24] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer. Automatic differentiation in pytorch. 2017.

[25] T. Salimans and D. P. Kingma. Weight normalization: A simple reparameterization to accelerate training of deep neural networks. In *NeurIPs*, pages 901–909, 2016.

[26] A. Shrivastava, T. Pfister, O. Tuzel, J. Susskind, W. Wang, and R. Webb. Learning from simulated and unsupervised images through adversarial training. In *CVPR*, volume 2, page 5, 2017.

[27] J. T. Springenberg, A. Dosovitskiy, T. Brox, and M. Riedmiller. Striving for simplicity: The all convolutional net. *arXiv preprint arXiv:1412.6806*, 2014.

[28] N. Srivastava, G. E. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *JMLR*, 15(1):1929–1958, 2014.

[29] R. K. Srivastava, K. Greff, and J. Schmidhuber. Training very deep networks. In *NeurIPS*, pages 2377–2385, 2015.

[30] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. A. Alemi. Inception-v4, inception-resnet and the impact of residual connections on learning. In *AAAI*, pages 4278–4284, 2017.

[31] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna. Rethinking the inception architecture for computer vision. In *CVPR*, pages 2818–2826, 2016.

[32] S. Xie, R. Girshick, P. Dollár, Z. Tu, and K. He. Aggregated residual transformations for deep neural networks. In *CVPR*, pages 5987–5995, 2017.

[33] S. Zagoruyko and N. Komodakis. Wide residual networks. *arXiv preprint arXiv:1605.07146*, 2016.

[34] X. Zhang, X. Zhou, M. Lin, and J. Sun. Shufflenet: An extremely efficient convolutional neural network for mobile devices. *arXiv preprint arXiv:1707.01083*, 2017.