Channel Equilibrium Networks for Learning Deep Representation

Wenqi Shao [*1] Shitao Tang [*2] Xingang Pan [3] Ping Tan [2] Xiaogang Wang [1] Ping Luo [4]

Abstract

Convolutional Neural Networks (CNNs) are typically constructed by stacking multiple building blocks, each of which contains a normalization layer such as batch normalization (BN) and a rectified linear function such as ReLU. However, this work shows that the combination of normalization and rectified linear function leads to inhibited channels, which have small magnitude and contribute little to the learned feature representation, impeding the generalization ability of CNNs. Unlike prior arts that simply removed the inhibited channels, we propose to "wake them up" during training by designing a novel neural building block, termed Channel Equilibrium (CE) block, which enables channels at the same layer to contribute equally to the learned representation. We show that CE is able to prevent inhibited channels both empirically and theoretically. CE has several appealing benefits. (1) It can be integrated into many advanced CNN architectures such as ResNet and MobileNet, outperforming their original networks. (2) CE has an interesting connection with the Nash Equilibrium, a well-known solution of a non-cooperative game. (3) Extensive experiments show that CE achieves state-of-the-art performance on various challenging benchmarks such as ImageNet and COCO.

1. Introduction

Normalization methods such as batch normalization (BN) (Ioffe & Szegedy, 2015), layer normalization (LN) (Ba et al., 2016) and instance normalization (IN) (Ulyanov et al., 2016) are important components for a wide range of tasks such as image classification (Ioffe & Szegedy, 2015), object detection (He et al., 2017a), and image generation (Miyato et al., 2018). They are often combined with rectified linear activation functions such as rectified linear unit (ReLU) (Glorot et al., 2011; Nair & Hinton, 2010), exponential linear unit (ELU) (Clevert et al., 2015) and leaky ReLU (LReLU) (Maas et al., 2013) and used in many recent advanced convolutional neural networks (CNNs). The combination of normalization and rectified unit becomes one of the most popular building block for CNNs.

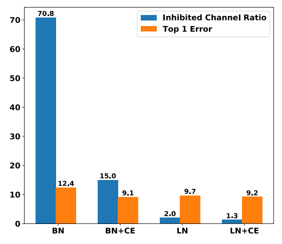
However, recent studies showed that the above building block leads to inhibited channels (as known as "channel collapse") after training a CNN, where a significant amount of feature channels always produce small values (Mehta et al., 2019) as shown in Fig.1(a&b). These inhibited channels contribute little to the learned feature representation, making the network more reliant on the remaining channels, which impedes its generalization ability as shown in (Morcos et al., 2018). For example, the lottery hypothesis (Frankle & Carbin, 2018) found that when a CNN is over-parameterized, it always contains unimportant ("dead") channels whose feature values are extremely small. Although these inhibited channels could be pruned in training to reduce the model size, it would lead to the limited generalization ability of the network (Yu et al., 2018; He et al., 2017b).

Instead of simply removing the inhibited "dead" channels, this work investigates an alternative to "wake them up" by proposing a novel neural building block, termed Channel Equilibrium (CE), to replace the ordinary combination of normalization and rectified units. CE encourages channels at the same layer of a network to contribute more equally in representation learning. With CE, all channels are useful in the learned representation, preventing CNNs from relying on specific channels and thus enhancing the generalization ability. For example, Fig.1 shows that CE not only reduces the number of inhibited channels but encourages all channels to contribute equally to network's prediction, when different combinations of normalization approaches and rectified units are presented, consistently improving their generalization to testing samples.

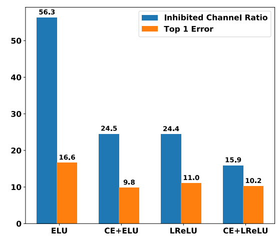
The main **contributions** of this work are three-fold. (1) We

[*]Equal contribution [1]Department of Electronic Engineering, The Chinese University of Hong Kong [2]Department of Computer Science, Simon Fraser University [3]Department of Information Engineering, The Chinese University of Hong Kong [4]Department of Computer Science,The University of Hong Kong. Correspondence to: Wenqi Shao <weqish@link.cuhk.edu.hk>, Ping Luo <pluo@cs.hku.hk>.

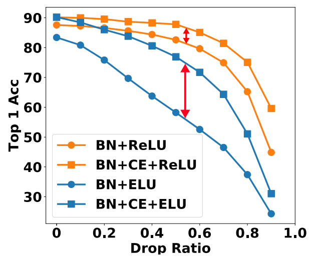
Proceedings of the 37th International Conference on Machine Learning, Vienna, Austria, PMLR 119, 2020. Copyright 2020 by the author(s).



(a) CE improves BN and LN.



(b) CE improves ELU and LReLU.



(c) Cumulative Ablation Curves

Figure 1. CE can improve many different normalization methods and rectified linear activation functions. For example, in (a-c), CE is used to train VGGNet (Simonyan & Zisserman, 2014) on CIFAR10 (Krizhevsky, 2009) with different normalizers and rectified units. (a) shows that the numbers of inhibited channels (*i.e.* channel features with values $< 10^{-2}$) are greatly reduced by applying CE with BN and LN, whose top-1 test errors are also reduced by using CE. In (a), ReLU is the activation. (b) shows similar phenomena where CE decreases inhibited channels and test errors compared to the ordinary ELU and LReLU functions. In (b), BN is the normalization method. (c) demonstrates that CE can encourage channels of BN+ReLU or ELU to contribute more equally to the network's prediction, by using cumulative ablation curve (Morcos et al., 2018), where accuracies are evaluated by randomly ablating channels (*i.e.* set their values to zeros) with an increasing ratio from '0' to '1'. When the ratio approaches '0.9', most channels are set to zeroed values, resulting in the worst accuracy. We see that CE presents a more gentle accuracy drop curve, implying that it makes the network less reliant on specific channels.

propose a novel neural building block for CNNs, termed Channel Equilibrium (CE), which encourages all channels to contribute equally to the learned feature representation. In theory, we show an interesting connection between CE and Nash Equilibrium, which is a well-known solution in game theory. (2) CE can significantly improve the generalization of existing networks with merely small computational overhead by plugging it into various advanced CNN architectures. For example, when CE is integrated into ResNet50 (He et al., 2016) and MobileNetv2 (Sandler et al., 2018), the resulting networks substantially outperform the original networks by 1.6% and 2.1% top-1 accuracy on ImageNet (Russakovsky et al., 2015), while merely introducing small extra computation. Specifically, the improvement of ResNet50+CE over ResNet50+BN is 60% larger than that of ResNet50+Squeeze-and-Excitation block (Hu et al., 2018) (*i.e.* 1.6% versus 1.0%). (3) The learned representation of CE can be well generalized to many other tasks such as object detection and segmentation. For example, CE trained with Mask RCNN (He et al., 2017a) using ResNet50 as backbone improves the AP metric on the MS-COCO dataset (Lin et al., 2014) by 3.4 compared to its counterpart. Models and code are available at `https://github.com/Tangshitao/CENet`.

2. Notation and Preliminary

This section presents the notations and backgrounds of normalization methods and rectified units.

Notations. We use regular letters to denote scalars such as 'x', and use bold letters to denote vectors (*e.g.* vector, matrix, and tensor) such as '$\boldsymbol{x}$'. For CNNs, we employ a 4D tensor, $\boldsymbol{x} \in \mathbb{R}^{N\times C\times H\times W}$, to represent the feature map in a layer, where N, C, H and W indicate sample size, channel size, height and width of a channel respectively. For example, x_{ncij} denotes a pixel at location (i,j) in the c-th channel of the n-th sample.

Overview. The recently advanced building block of CNNs consists of a normalization layer and a rectified linear function denoted as $g(\cdot)$. We have

$$\begin{aligned} &y_{ncij} = g(\tilde{x}_{ncij}), \text{ where} \\ &\tilde{x}_{ncij} = \gamma_c \bar{x}_{ncij} + \beta_c, \quad \bar{x}_{ncij} = (x_{ncij} - \mu_k)/\sigma_k. \end{aligned} \tag{1}$$

In Eqn.(1), y_{ncij} denotes the output value after applying rectified activation function and normalization method. $k \in \Omega = \{\text{IN}, \text{BN}, \cdots\}$ where Ω indicates a set of normalization methods. μ_k and σ_k are mean and standard deviation estimated by using the normalizer k. Moreover, $\tilde{x}_{ncij}$ and $\bar{x}_{ncij}$ respectively represent the features after normalization and standardization (*i.e.* with zeroed mean and unit standard deviation). For each channel, γ_c and β_c are two parameters, which re-scale and re-shift the standardized features $\bar{x}_{ncij}$. Furthermore, $g(\cdot)$ denotes a rectified linear function. For instance, we have $g(x) = x \cdot \mathbf{1}_{x\geq 0} + ax \cdot \mathbf{1}_{x<0}$. It represents ReLU (Nair & Hinton, 2010) when $a = 0$, while it represents leaky ReLU (LReLU) (Maas et al., 2013) when $a \in (0,1)$.

Inhibited Channels. Eqn.(1) shows that many normalization approaches perform an affine transformation by using the parameters γ_c and β_c for each channel. Previous work (Mehta et al., 2019) shows that after training, amounts of γ_c and y_{ncij} for all $i \in [H]$ and $j \in [W]$ would get small. We see this by treating $\bar{x}_{ncij}$ in Eqn.(1) as a standard Gaussian random variable following (Arpit et al., 2016). When the value of γ_c becomes small, Remark 1 tells us that the mean and the variance of the channel output y_{ncij} would also be small (proof is provided in Appendix Sec.A). In this case, the c-th channel becomes inhibited and contributes little to the representation learning. For evaluation, this paper treats those channels with magnitudes smaller than 10^{-2} as inhibited channels. We observe that inhibited channels largely emerge in many different combinations of normalizations and rectified units, including BN (Ioffe & Szegedy, 2015), IN (Ulyanov et al., 2016), LN (Ba et al., 2016), ReLU, ELU (Clevert et al., 2015) and LReLU (Maas et al., 2013) as shown in Fig.1(a&b). The existence of inhibited channels makes the network rely more on the remaining activated channels, impeding the generalization of CNNs (Morcos et al., 2018).

Remark 1. *Let a random variable $z \sim \mathcal{N}(0,1)$ and $y = max\{0, \gamma_c z+\beta_c\}$. Then we have $\mathbb{E}_z[y] = 0$ and $\mathbb{E}_z[y^2] = 0$ if and only if $\beta_c \leq 0$ and γ_c sufficiently approaches 0.*

Decorrelation. Although the above ELU and LReLU extend the ReLU activation function by making its negative part has a non-zero slope, they are not able to prevent inhibited channels. Different from these methods, this work prevents inhibited channels by decorrelation operation performed after the normalization layer. Typically, a decorrelation operator is expressed as the inverse square root of the covariance matrix, denoted as $\mathbf{\Sigma}^{-\frac{1}{2}}$ where $\mathbf{\Sigma}$ is the covariance matrix and is usually estimated over a minibatch of samples (Huang et al., 2018; 2019). This work discovers that decorrelating feature channels after normalization layer can increase the magnitude of all the feature channels, making all channels useful in the learned representation.

Furthermore, suppose that every single channel aims to contribute to the learned feature representation, we show that decorrelating feature channels after the normalization method can be connected with the Nash Equilibrium for each instance. In this sense, constructing a decorrelation operator for every single sample is also crucial for representation learning (Yang et al., 2019). As presented in the below section, the proposed Channel Equilibrium (CE) module is carefully designed by exploring a dynamic decorrelation operator conditioned on each instance sample.

3. Channel Equilibrium (CE) Block

This section introduces the CE block, which contains a branch of batch decorrelation (BD) and a branch of instance reweighting (IR). We show that the CE block can increase the magnitude of feature channels. We also show the connection between the CE block and the Nash Equilibrium.

In particular, a CE block is a computational unit that encourages all channels to contribute to the feature representation by decorrelating feature channels. Unlike previous methods (Huang et al., 2018; 2019) that decorrelated features after the convolutional layer given a minibatch of samples, CE conditionally decorrelates features after the normalization layer for each sample. Rewriting Eqn.(1) into a vector, we have the formulation of CE

$$\boldsymbol{p}_{nij} = \boldsymbol{D}_n^{-\frac{1}{2}}(\mathrm{Diag}(\boldsymbol{\gamma})\bar{\boldsymbol{x}}_{nij} + \boldsymbol{\beta}) \tag{2}$$

where $\boldsymbol{p}_{nij} \in \mathbb{R}^{C\times 1}$ is a vector of C elements that denote the output of CE for the n-th sample at location (i,j) for all channels. $\boldsymbol{D}_n^{-\frac{1}{2}}$ is a decorrelation operator and $\boldsymbol{D}_n$ is the covariance matrix defined in CE. The subscript n is the sample index, suggesting that the decorrelation operator is performed for each sample but not a minibatch of samples. In Eqn.(2), $\bar{\boldsymbol{x}}_{nij} \in \mathbb{R}^{C\times 1}$ is a vector by stacking elements from all channels of $\bar{x}_{ncij}$ into a column vector. $\boldsymbol{\gamma} \in \mathbb{R}^{C\times 1}$ and $\boldsymbol{\beta} \in \mathbb{R}^{C\times 1}$ are two vectors by stacking γ_c and β_c of all the channels respectively. $\mathrm{Diag}(\boldsymbol{\gamma})$ returns a diagonal matrix by using $\boldsymbol{\gamma}$ as diagonal elements.

To decorrelate the feature channels conditioned on each input, statistics of the channel dependency with respect to both the minibatch and each sample are embedded in the matrix $\boldsymbol{D}_n$. We achieve this by incorporating a covariance matrix $\mathbf{\Sigma}$ with an instance variance matrix, $\mathrm{Diag}(\boldsymbol{v}_n)$, where $\boldsymbol{v}_n \in \mathbb{R}^{C\times 1}$ denotes the adaptive instance variances for all channels. In this way, we have

$$\boldsymbol{D}_n = \lambda\mathbf{\Sigma} + (1-\lambda)\mathrm{Diag}(\boldsymbol{v}_n), \qquad \boldsymbol{v}_n = f(\tilde{\boldsymbol{\sigma}}_n^2), \tag{3}$$

where $\mathbf{\Sigma} \in \mathbb{R}^{C\times C}$ is estimated by a minibatch of samples after normalization,$\{\tilde{\boldsymbol{x}}_n\}_{n=1}^N$, $\tilde{\boldsymbol{\sigma}}_n^2 \in \mathbb{R}^{C\times 1}$ is a vector of variance of the n-th instance estimated by using $\tilde{\boldsymbol{x}}_n$ for all channels (Ulyanov et al., 2016), $f : \mathbb{R}^{C\times 1} \to \mathbb{R}^{C\times 1}$ models channel dependencies and returns an adaptive instance variance. And $\lambda \in (0,1)$ is a learnable ratio used to switch between the batch and the instance statistics.

Given Eqn.(3), the decorrelation operator $\boldsymbol{D}_n^{-\frac{1}{2}}$ can be relaxed by using the Jensen inequality for matrix functions (Pečarić, 1996). We have

$$\begin{aligned}\boldsymbol{D}_n^{-\frac{1}{2}} &= [\lambda\mathbf{\Sigma} + (1-\lambda)\mathrm{Diag}(\boldsymbol{v}_n)]^{-\frac{1}{2}} \\ &\preceq \lambda \underbrace{\mathbf{\Sigma}^{-\frac{1}{2}}}_{\text{batch decorrelation}} + (1-\lambda)\underbrace{[\mathrm{Diag}(\boldsymbol{v}_n)]^{-\frac{1}{2}}}_{\text{instance reweighting}},\end{aligned} \tag{4}$$

where $\boldsymbol{A} \preceq \boldsymbol{B}$ indicates $\boldsymbol{B}-\boldsymbol{A}$ is semi-definite positive. In fact, the term on the right hand side in Eqn.(4) is also a linear approximation of $\boldsymbol{D}_n^{-\frac{1}{2}}$ by Taylor expansion where the residual term depends on the spectrum of $\mathbf{\Sigma}$ and $\mathrm{Diag}(\boldsymbol{v}_n)$. The above relaxation is made because of two reasons. (1) *Reduce Computational Complexity*. It allows less computational cost for each training step since the relaxed form only needs to calculate the inverse of square root $\mathbf{\Sigma}^{-\frac{1}{2}}$ once, meanwhile the other branch $\mathrm{Diag}(\boldsymbol{v}_n)^{-\frac{1}{2}}$ is easy to compute. (2) *Accelerate Inference.* $\mathbf{\Sigma}^{-\frac{1}{2}}$ is a moving-average statistic in inference, which can be absorbed into the previous layer, thus enabling fast inference.

In the following descriptions, we treat $\mathbf{\Sigma}^{-\frac{1}{2}}$ in Eqn.(4) as batch decorrelation (BD) and treat $[\mathrm{Diag}(\boldsymbol{v}_n)]^{-\frac{1}{2}}$ as instance reweighting (IR). The former one performs decorrelation by using a covariance matrix estimated in an entire minibatch, while the latter one adjusts correlations among feature channels by reweighting each channel with the inverse square root of an adaptive variance for each instance. Integrating both of them yields a dynamic decorrelation operator conditioned on each instance in the CE block whose forward representation is illustrated in Fig.2(b).

3.1. Batch Decorrelation (BD)

Although many previous work (Huang et al., 2018; 2019; Pan et al., 2019) have investigated decorrelation (whitening) methods by using the covariance matrix, all of them are applied in the normalization layer. Their drawback is that the channel features after whitening are still scaled by γ channel-wisely in the normalization layer, thus producing inhibited channels. Instead, CE is applied after the normalization layer (after γ), which as will be shown, is able to explicitly prevent inhibited channels. We take batch normalization (BN) as an example to illustrate CE. Note that CE can be applied to any normalization methods and activation functions.

Consider a tensor $\bar{\boldsymbol{x}}$ after a BN layer, it can be reshaped as $\bar{\boldsymbol{x}} \in \mathbb{R}^{C\times M}$ and $M = N\cdot H\cdot W$. Then the covariance matrix $\mathbf{\Sigma}$ of the normalized features $\tilde{\boldsymbol{x}}$ can be written as (details in Sec.B of Appendix)

$$\mathbf{\Sigma} = \boldsymbol{\gamma}\boldsymbol{\gamma}^{\mathsf{T}} \odot \frac{1}{M}\bar{\boldsymbol{x}}\bar{\boldsymbol{x}}^{\mathsf{T}}, \tag{5}$$

where $\bar{\boldsymbol{x}}$ is a standardized feature with zero mean and unit variance and $\odot$ indicates elementwise multiplication. It is observed that each element Σ_{ij} represents the dependency between the i-th channel and the j-th channel, and it is scaled by $\gamma_i\gamma_j$ after normalization.

The BD branch requires computing $\mathbf{\Sigma}^{-\frac{1}{2}}$, which usually uses eigen-decomposition or SVD, thus involving heavy computations (Huang et al., 2018). Instead, we adopt an efficient Newton's Iteration to obtain $\mathbf{\Sigma}^{-\frac{1}{2}}$ (Bini et al., 2005; Higham, 1986). Given a covariance matrix $\mathbf{\Sigma}$, Newton's Iteration calculates $\mathbf{\Sigma}^{-\frac{1}{2}}$ by following the iterations,

$$\begin{cases} \mathbf{\Sigma}_0 = \boldsymbol{I} \\ \mathbf{\Sigma}_k = \frac{1}{2}(3\mathbf{\Sigma}_{k-1} - \mathbf{\Sigma}_{k-1}^3\mathbf{\Sigma}),\ k = 1,2,\cdots,T. \end{cases} \tag{6}$$

where k is the iteration index and T is the iteration number ($T = 3$ in our experiments). Note that the convergence of Eqn.(6) is guaranteed if $\|\boldsymbol{I} - \mathbf{\Sigma}\|_2 < 1$ (Bini et al., 2005). To satisfy this condition, $\mathbf{\Sigma}$ can be normalized by $\mathbf{\Sigma}/\mathrm{tr}(\mathbf{\Sigma})$ following (Huang et al., 2019), where $\mathrm{tr}(\cdot)$ is the trace operator. In this way, the normalized covariance matrix can be written as $\mathbf{\Sigma} = \frac{\boldsymbol{\gamma}\boldsymbol{\gamma}^{\mathsf{T}}}{\|\boldsymbol{\gamma}\|_2^2} \odot \frac{1}{M}\bar{\boldsymbol{x}}\bar{\boldsymbol{x}}^{\mathsf{T}}$. To sum up, in the training stage, the BD branch firstly calculates a normalized covariance matrix and then applies Newton's Iteration to obtain its inverse square root, reducing computational cost compared to the SVD decomposition. In the testing stage, BD can be merged into the convolutional layers, which merely adds small extra computation.

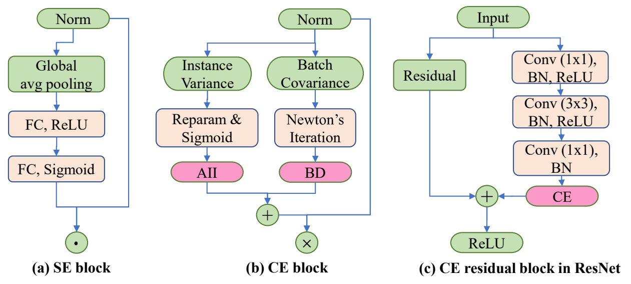


Figure 2. **Comparisons** of (a) SE block (Hu et al., 2018), (b) CE block and (c) CE residual block in ResNet. $\odot$ denotes broadcast element-wise multiplication, $\oplus$ denotes broadcast elementwise addition and $\otimes$ denotes matrix multiplication. The SE block in (a) is not able to equalize feature representation, and it has larger computations and lower performance than (b). In (b), CE has two lightweight branches, BD and IR. (c) shows CE can be easily stacked into many advanced networks such as ResNet with merely small extra computation.

3.2. Instance Reweighting (IR)

Other than the BD branch, the decorrelation operator is also desired for each sample (Yang et al., 2019). we achieve this by incorporating the BD with a branch of instance reweighting (IR) as shown in Eqn.(4).

Specifically, the input of IR is denoted as $\tilde{\boldsymbol{\sigma}}_n^2 \in \mathbb{R}^{C\times 1}$, which can be computed as below (details in Appendix Sec.B)

$$\tilde{\boldsymbol{\sigma}}_n^2 = \mathrm{diag}(\boldsymbol{\gamma}\boldsymbol{\gamma}^{\mathsf{T}}) \odot \frac{(\boldsymbol{\sigma}_{\mathrm{IN}}^2)_n}{\boldsymbol{\sigma}_{\mathrm{BN}}^2}, \tag{7}$$

where $\mathrm{diag}(\boldsymbol{\gamma}\boldsymbol{\gamma}^{\mathsf{T}}) \in \mathbb{R}^{C\times 1}$ extracts the diagonal of the given matrix, $(\boldsymbol{\sigma}_{\mathrm{IN}}^2)_n \in \mathbb{R}^{C\times 1}$ and $\boldsymbol{\sigma}_{\mathrm{BN}}^2 \in \mathbb{R}^{C\times 1}$ represent the variances estimated by using IN (Ulyanov et al., 2016) and BN (Ioffe & Szegedy, 2015) respectively. In Eqn.(7), the vector division is applied elementwisely. Similar to Eqn.(5), the input of IR is scaled by γ_c^2 for the c-th channel.

The IR branch returns an inverse square root of an adaptive instance inverse, denoted as $[\mathrm{Diag}(\boldsymbol{v}_n)]^{-\frac{1}{2}}$, which is used to adjusts correlations among feature channels. It needs to satisfy two requirements. First, note that $\boldsymbol{v}_n = f(\tilde{\boldsymbol{\sigma}_n^2})$ from Eqn.(3), while $\tilde{\boldsymbol{\sigma}}_n^2$ is just a vector of variances calculated within each channel. To adjust correlations by IR branch, the dependencies among channels should be embedded in transformation f for each sample. Second, the output of IR should have the same magnitude as the inverse square root of variance or covariance in the BD branch such that neither of them is dominant in CE.

To achieve the above, a reparameterization trick is employed to generate the inverse square root of instance variance. Let $s = \frac{1}{NC}\sum_{n,c}^{N,C}(\tilde{\boldsymbol{\sigma}}_n^2)_c$ be the estimate of variance for all channels and all samples in a minibatch, the transformation f in Eqn.(3) can be reparameterized as below,

$$[\mathrm{Diag}(\boldsymbol{v}_n)]^{-\frac{1}{2}} = \mathrm{Diag}(\mathrm{Sigmoid}(\tilde{\boldsymbol{\sigma}}_n^2;\boldsymbol{\theta}))\cdot s^{-\frac{1}{2}}, \tag{8}$$

where $s^{-\frac{1}{2}}$ represents the magnitude of the inverse square root of variance. And a subnetwork with the parameter of $\boldsymbol{\theta}$ is used to model the dependencies among channels by following the designs of the SE block (Hu et al., 2018) and GC block (Cao et al., 2019). Here we use a Sigmoid activation to generate a set of channel weights, which is used to control the strength of the inverse square root of variance for each channel. In this way, the output of the IR branch not only has the same magnitude as that of BD but also encodes channel dependencies. We provide detailed descriptions of the subnetwork in Appendix Sec.B.1.

3.3. Discussions

Network Architectures. Different from SE block in Fig.2(a) which only reweights feature channels by a bottleneck network(Hu et al., 2018), CE decorrelates incoming feature channels after the normalization layer by combining two branches, i.e. batch decorrelation (BD) and instance reweighting(IR), as shown in Fig.2(b). The CE block can be readily integrated into various advanced architectures, such as ResNet, VGGNet (Simonyan & Zisserman, 2014), ShuffleNetv2 (Ma et al., 2018) and MobileNetv2 (Sandler et al., 2018), by inserting it in block of normalization and rectified units.

The flexibility of the CE block makes it easy to construct a series of CENets. For example, we consider the residual networks (ResNet). The core unit of the ResNet is the residual block that consists of '1×1', '3×3' and '1×1' convolution layers, sequentially. The CE block is applied in the last '1×1' convolution layer by plugging the CE module before ReLU non-linearity, as shown in Fig.2(c). Following similar strategies, CE is further integrated into ShuffleNetv2 and MobileNetv2 to construct CE-ShuffleNetv2 and CE-MobileNetv2. whose diagrams are provided in Sec.E of Appendix. We also explore the integration strategy used to incorporate CE blocks into a network architecture in Sec.F of Appendix.

Magnitude of Gamma and Feature Channels. The CE block can prevent the inhibited channels through the BD branch. Remark 1 shows that inhibited channels are usually related to γ_c and the output y_{ncij} with small values. Here we discover that BD branch can increase the magnitude of gamma and channel features. To see this, by combining Eqn.(4) and Eqn.(2), the output of BD can be expressed as $\boldsymbol{p}_{nij}^{\mathrm{BD}} = \mathrm{Diag}(\boldsymbol{\Sigma}^{-\frac{1}{2}}\boldsymbol{\gamma})\bar{x}_{nij} + \boldsymbol{\Sigma}^{-\frac{1}{2}}\boldsymbol{\beta}$. Compared with Eqn.(1), an equivalent gamma for BD branch can be defined as $\hat{\boldsymbol{\gamma}} = \boldsymbol{\Sigma}^{-\frac{1}{2}}\boldsymbol{\gamma}$. The proposition 1 shows that BD increases the magnitude of $\hat{\boldsymbol{\gamma}}$ and feature channels in a feed-forward way. Therefore, it is effective to prevent inhibited channels. The proof of proposition 1 is provided in Sec.C of Appendix.

Proposition 1. *Let $\boldsymbol{\Sigma}$ be covariance matrix of feature maps after batch normalization. Then, (1) assume that $\boldsymbol{\Sigma}_k = \boldsymbol{\Sigma}^{-\frac{1}{2}}, \forall k = 1,2,3,\cdots,T$, we have $|\hat{\gamma_c}| > |\gamma_c|, \forall c \in [C]$. (2) Denote $\boldsymbol{\rho} = \frac{1}{M}\bar{\boldsymbol{x}}\bar{\boldsymbol{x}}^{\mathsf{T}}$ in Eqn.(5) and $\tilde{\boldsymbol{x}}_{nij} = \mathrm{Diag}(\boldsymbol{\gamma})\bar{\boldsymbol{x}}_{nij} + \boldsymbol{\beta}$. Assume $\boldsymbol{\rho}$ is full-rank, then $\left\|\boldsymbol{\Sigma}^{-\frac{1}{2}}\tilde{\boldsymbol{x}}_{nij}\right\|_2 > \|\tilde{\boldsymbol{x}}_{nij}\|_2$*

Connection with Nash Equilibrium. We understand normalization and ReLU block from a perspective in game theory (Leshem & Zehavi, 2009). In this way, an interesting connection between the proposed CE block and the well-known Nash Equilibrium can be built. To be specific, for every underlying sample, we treat the output p_{cij} in Eqn.(2) as the transmit power allocated to neuron (i,j) for the c-th channel. Here the subscript 'n' is omitted for clarity. Then each neuron is associated with a maximum information rate which determines the maximum transmit power available to the neuron (Cover & Thomas, 2012). In strategic games (Osborne & Rubinstein, 1994), each channel wants to maximize its benefit. In the context of CNN, we suppose that every channel obtains its output by maximizing the sum of the maximum information rate of all neurons.

Furthermore, considering the dependencies among channels, the channels are thought to play a non-cooperative game, named Gaussian interference game, which admits a unique Nash Equilibrium solution (Laufer et al., 2006). When all the outputs are activated (larger than 0), this Nash Equilibrium solution has an explicit expression, the linear proxy of which has the same form with the expression of CE in Eqn.(2). It shows that decorrelating features after the normalization layer can be connected with Nash Equilibrium, implying that the proposed CE block indeed encourages every channel to contribute to the network's computation. Note that the Nash Equilibrium solution can be derived for every single sample, implying that the decorrelation operation should be performed conditioned on each instance sample. This is consistent with our design of the CE block. We present detailed explanations about the connection between CE and Nash Equilibrium in Sec.D of the Appendix.

4. Related Work

Sparsity in ReLU. An attractive property of ReLU (Sun et al., 2015; Nair & Hinton, 2010) is sparsity, which brings potential advantages such as information disentangling and linear separability. However, (Lu et al., 2019) and (Mehta et al., 2019) pointed out that some ReLU neurons may become inactive and output 0 values for any input. Previous work tackled this issue by designing new activation functions, such as ELU (Clevert et al., 2015) and Leaky ReLU (Maas et al., 2013). Recently, Lu et al. (2019) also tried to solve this problem by modifying the initialization scheme. Different from these work, CE focus on explicitly preventing inhibited channel in a feed-forward way by encouraging channels at the same layer to contribute equally to learned feature representation.

Normalization and decorrelation. There are many practices on normalizer development, such as batch normalization (BN) (Ioffe & Szegedy, 2015), group normalization (GN) (Wu & He, 2018), examplar normalization (Zhang et al., 2020), switchable normalization (Luo et al., 2018) and its sparse version SSN (Shao et al., 2019). A normalization scheme is typically applied after a convolution layer and contains two stages: standardization and rescaling. Another type of normalization methods not only standardizes but also decorrelates features, like DBN (Huang et al., 2018), IterNorm (Huang et al., 2019) and Decorrelated BN (DBN) (Huang et al., 2018). Despite their success in stabilizing the training, little is explored about the relationship between these methods and inhibited channels. Fig.1 shows that inhibited channels emerge in VGGNet where 'BN+ReLU' or 'LN+ReLU' is used. Unlike previous decorrelated normalizations where decorrelation operation is applied after a convolution layer, our CE explicitly decorrelates features after normalization and is designed to prevent inhibited channels emerging in the block of normalization and rectified units.

5. Experiments

We extensively evaluate the proposed CE on two basic vision tasks, image classification on ImageNet (Russakovsky et al., 2015) and object detection/segmentation on COCO (Lin et al., 2014).

5.1. Image Classification on ImageNet

We first evaluate CE on the ImageNet benchmark. The models are trained on the $\sim$ 1.28M training images and evaluate on the 50,000 validation images. The top-1 and top-5 accuracies are reported. We are particularly interested in whether the proposed CE has better generalization to testing samples in various modern CNNs such as ResNets (He et al., 2016), MobileNetv2 (Sandler et al., 2018), ShuffleNetv2 (Ma et al., 2018) compared with the SE block (Hu et al., 2018). The training details are illustrated in Sec.G of the Appendix.

Performance comparison on ResNets. We evaluate CE on representative residual network structures including ResNet18, ResNet50 and ResNet101. The CE-ResNet is compared with baseline (plain ResNet) and SE-ResNet. For fair comparisons, we use publicly available code and re-implement baseline models and SE modules with their respective best settings in a unified Pytorch framework. To save computation, the CE blocks are selectively inserted into the last normalization layer of each residual block. Specifically, for ResNet18, we plug the CE block into each residual block. For ResNet50, CE is inserted into all residual blocks except for those layers with 2048 channels. For ResNet101, the CE blocks are employed in the first seven residual blocks.

Improved generalization on ResNets. As shown in Table 1, our proposed CE outperforms the BN baseline and SE block by a large margin with little increase of GFLOPs. Concretely, CE-ResNet18, CE-ResNet50 and CE-ResNet101 obtain top-1 accuracy increase of 1.5%, 1.6% and 1.0% compared with the corresponding plain ResNet architectures, confirming the improved generalization on testing samples. Note that the shallower network, i.e. CE-ResNet50, even outperforms the deeper network, i.e. plain ResNet101 (78.0), suggesting that the learned features under CE blocks are more representative. We plot training and validation error during the training process for ResNet50, SE-ResNet50 and CE-ResNet50 in Fig.4(a). Compared to ResNet50 and SE-ResNet50, CE-ResNet50 obtains lower training error and validation error than that of SE-ResNet50, implying that CE improves the generalization ability of the network.

Comparable computational cost. We also analyze the complexity of BN, SE, and CE in terms of GFLOPs, GPU and CPU running time. The definition of GFLOPs follows (Sandler et al., 2018), $i.e.$, the number of multiply-adds. We evaluate the inference time[1] with a mini-batch of 32. In terms of GFLOPs, the CE-ResNet18, CE-ResNet50, CE-ResNet101 has only 0.55%, 0.48% and 0.25% relative increase in GFLOPs compared with plain ResNet. Additionally, the CPU and GPU inference time of CENet is nearly the same with SENet.

Improved generalization on light-weight networks. We further investigate the efficacy of our proposed CE in two representative light-weight networks, MobileNetv2 and ShuffleNetv2. The results of the comparison are given in Table 2. It is seen that CE blocks bring conspicuous improvements in top-1 and top-5 accuracies on test examples at a minimal increase in computational burden. For MobileNetv2 $1\times$, CE even improves top-1 accuracy of baseline by 2.1%, showing that CE enables the network to generalize well in testing samples.

Comparison with normalization methods using decorrelation. Many normalization approaches also use decorre-

[1]The CPU type is Intel Xeon CPU E5-2682 v4, and the GPU is NVIDIA GTX1080TI. The implementation is based on Pytorch

	ResNet18			ResNet50			ResNet101		
	Baseline	SE	CE	Baseline	SE	CE	Baseline	SE	CE
Top-1	70.4	71.4	**71.9**	76.6	77.6	**78.2**	78.0	78.5	**79.0**
Top-5	89.4	90.4	**90.8**	93.0	93.7	**94.1**	94.1	94.1	**94.6**
GFLOPs	1.82	1.82	1.83	4.14	4.15	4.16	7.87	7.88	7.89
CPU (s)	3.69	3.69	4.13	8.61	11.08	11.06	15.58	19.34	17.05
GPU (s)	0.003	0.005	0.006	0.005	0.010	0.009	0.011	0.040	0.015

Table 1. Comparisons with baseline and SENet on ResNet-18, -50, and -101 in terms of accuracy, GFLOPs, CPU and GPU inference time on ImageNet. The top-1,-5 accuracy of our CE-ResNet is higher than SE-ResNet while the computational cost in terms of GFLOPs, GPU and CPU inference time remain nearly the same.

	MobileNetv2 $1\times$			ShuffleNetv2 $0.5\times$			ShuffleNetv2 $1\times$		
	top-1	top-5	GFLOPs	top-1	top-5	GFLOPs	top-1	top-5	GFLOPs
Baseline	72.5	90.8	0.33	59.2	82.0	0.05	69.0	88.6	0.15
SE	73.5	91.7	0.33	60.2	82.4	0.05	70.7	89.6	0.15
CE	**74.6**	**91.7**	0.33	**60.5**	**82.7**	0.05	**71.2**	**89.8**	0.16

Table 2. Comparisons with baseline and SE on lightweight networks, MobileNetv2 and ShuffleNetv2, in terms of accuracy and GFLOPs on ImageNet. Our CENet improves the top-1 accuracy by a large margin compared with SENet with nearly the same GFLOPs.

Backbone	ResNet50			ResNet18		
Method	DBN	IterNorm	CE	DBN	IterNorm	CE
Top-1	76.9	77.1	**78.2**	71.0	71.1	**71.9**

Table 3. Comparison between the proposed CE and other normalization method using decorrelation on ImageNet dataset. CE achieves higher top-1 accuracy on both ResBet50 and ResNet18.

lation operation such as Decorrelated BN (DBN) (Huang et al., 2018) and IterNorm (Huang et al., 2019) to stabilize the course of training . However, all of them are applied after convolution layer. Thus, the inhibited channels still emerge due to the use of affine transformation (*i.e.* γ and β. Specifically, the VGGNet with IterNorm trained on CIFAR10 has 66.2% inhibited channels at the same training setting in Fig.1. Instead, our proposed CE decorrelates features after normalization layer conditioned on each instance, which has been proved to be able to prevent inhibited channels. Here we show that CE obtains a gain of performance on ImageNet with ResNet18 and ResNet50 compared with DBN and IterNorm. The results are repoted in Table 3.

5.2. Analysis of CE

In this section, we investigate the robustness of CE against label corruptions (Zhang et al., 2016). We demonstrate that CE encourages channels to contribute equally to the learned feature representation and reduces correlations among feature channels. More experimental results are presented in Appendix Sec.F.

CE improves generalization ability in corrupted label setting. We have shown in Sec.5.1 that CE has a better generalization to testing samples that are drawn from the same distributions of training ones. Here we show the robustness of CE when the labels of training samples are randomly corrupted with different corruption ratios (Zhang et al., 2016). We train VGGNet with BN and CE on CIFAR10 (Krizhevsky, 2009) under the same training settings

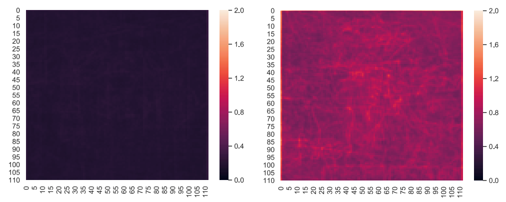


Figure 3. **Left** & **Right** show the magnitude of feature channels after BN and CE layer, respectively. The ℓ_2 norm of feature channels at each location (i,j) after the first BN or CE layer of the trained VGGNet is visualized. CE increase the magnitude of channel features.

in Fig.1. Especially, VGGNet with CE is trained to obtain the same training error of VGGNet with BN. The top-1 test errors of VGGNet with BN and CE are plotted in Fig.4(e). It shows that CE consistently improves the generalization ability under a wide range of corruption label ratios.

CE encourages channels to contribute more equally to the learned feature representation. We demonstrate this in two ways. Firstly, by applying a decorrelation operator, neurons across C channels after CE block have a larger magnitude at every location (i,j). We use ℓ_2 norm to measure the magnitude of feature channels. The average of the magnitude for each location (i,j) after CE blocks are calculated over a random minibatch of samples. Results are obtained by training BN-VGGNet and CE-VGGNet. Fig.3 shows that neurons across channels in CE-VGGNet have a larger magnitude than those in BN-VGGNet, meaning that CE makes more channels useful in the feature representation.

Secondly, the importance of feature channels to the network's prediction is more equal. We investigate this by using a cumulative ablation method (Morcos et al., 2018). Typically, the importance of a single channel to the network's computation can be measured by the relative performance drop once that channel is removed (clamping activity a feature map to zero). If the importance of channels to the

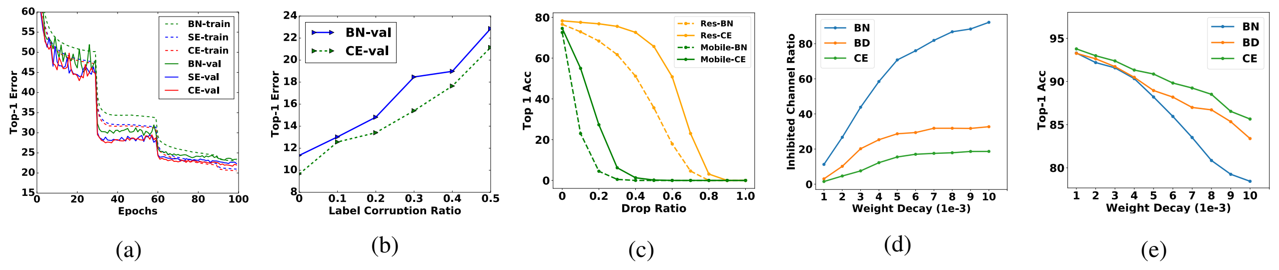


Figure 4. (a) shows the training and validation error curves on ImageNet with ResNet50 as backbone for BN, SE and CE. CE improves both training error and validation error. (b) shows Robustness test of CE at 5-level corruption labels on CIFAR10 dataset where '0' corruption indicates no corrupted labels. CE gives improved test error over baselines in all label corruption ratios. (c) shows cumulative ablation curves for MobileNetv2 and ResNet50 on ImageNet dataset respectively. We randomly ablate channels with an increasing fraction in the first normalization layers. CE also helps to equalize the importance of channels on ImageNet. (d) & (e) are inhibited channel ratio and top-1 accuracy curves when training VGGNet on CIFAR-10 under different weight decays. Compared to networks trained with BN, networks trained with the proposed BD and CE can effectively prevent inhibited channels and retain a higher performance as strength of weight decay increases.

network's prediction is more equal, the network will rely less on some specific channels and thus the performance will drop more gently. With this method, we see how ResNet50 and MobileNetv2 $1\times$ with CE blocks respond to the cumulative random ablation of channels on ImageNet. We plot the ablation ratio versus the top-1 accuracy in Fig.4(c). It can be observed that the CE block is able to resist the cumulative random ablation of channels on both ResNet50 and MobileNetv2 compared with the original networks, showing that CE can effectively make channels contribute more equally to the network's prediction.

CE mitigates the inhibited channels, which is robust to different strength of weight decay. (Mehta et al., 2019) revealed that the number of inhibited channels increases as the strength of weight decay grows. As shown in Fig.4(d), the number of inhibited channel in CE-VGGNet trained on CIFAR10 is conspicuously reduced under all weight decays compared with BN-VGGNet. We also note that the BD branch in the CE block is also able to prevent inhibited channels, which is consistent with proposition 1. CE achieves the lower inhibited channel ratio than BD, implying that IR also helps to prevent inhibited channel. Fig.4(e) further shows that the top-1 accuracy of VGGNet with BN drops significantly as the weight decay increases, but CE can alleviate accuracy drop, implying that excessive inhibited channels impede network' generalization to testing samples.

CE reduces correlations among feature channels. By design, CE decorrelates feature channels by the BD branch, which is then used to generate a decorrelation operator conditioned on each sample by combining with the IR branch. We investigate the effect of reducing correlations among feature channels of BN (Baseline), IR, BD, and CE by applying VGGNet. As shown in Fig.5, the correlations among feature channels at different depths of the network are remarkably decreased when CE, IR, and BD are used, implying that CE can reduce the redundancy in feature channels. We also observe that CE can learn adaptive correlations at different depths of the network by combining BD and IR. Note that in deeper layers of the network, the decorrelation of CE behaves more similar to decorrelation of IR compared with BD, showing that decorrelating feature channels for each instance is useful in higher layers.

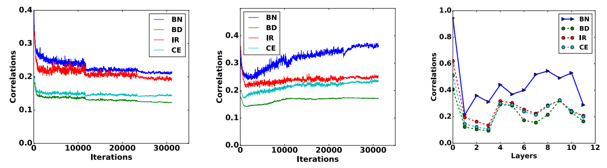


Figure 5. **Left** & **Middle** show the correlations for the output response maps in shallow (Left) and deeper (Middle) CE layers during the whole training period. **Right** shows the curves of correlations at different layers. Results are obtained by applying VGGNet as backbone. All of CE, IR and BD can achieve lower correlations among feature channels than BN baseline.

5.3. Ablation Study

This section investigates the influence of hyper-parameters in CE.

Effect of different λ. From Eqn.(2), we see that λ represents the strength of two branches, i.e. BD and IR. To explore how λ in Eqn.(2) affects the final performance, we consider four different settings of λ: (a) fix λ at 0, in this setting, only IR branch in CE is used; (b) fix λ at 1, where CE turns into BD branch; and (c) treat λ as a learnable variable, which is the proposed CE. The above settings are compared on ResNet50 trained on ImageNet. The top-1 accuracy is reported in Table 4. It is observed that the ResNet50 with BD and IR are 0.4 and 0.7 higher than the plain ResNet-50 respectively. However, when they are combined, the top-1 accuracy improves by 1.6, higher than the combined accuracy increase (1.1), which demonstrates that they benefit

λ	Baseline	$\lambda = 1$	$\lambda = 0$	learnable
top-1	76.6	77.0	77.3	**78.2 (+1.6)**

Table 4. Effect of different λ. Note that $\lambda = [0, 1, learnable]$ represent IR branch, BD branch and the proposed CE, respectively. We use ResNet-50 as the basic structure. The proposed CE achieves the best performance, which shows that training λ by SGD to learn the ratio of BD and IR branch is crucial for CE .

from each other. Therefore, training λ by SGD to learn the ratio of BD and IR branch is crucial for CE.

Effect of the number of CE block. To investigate this, we train VGGNet (Simonyan & Zisserman, 2014) on CIFAR10 (Krizhevsky, 2009) by stacking CE after 100%, 75% and 50% BN layers. Their top-1 acc are 89.7, 89.6 and 89.4 respectively, showing that more CEs lead to better performance. However, the downside of CE is that too many CE blocks in the network would bring much computational cost. In practice, we should carefully trade off the performance and computation burden by integrating

Effect of the location of CE block. Firstly, we investigate how the location of CE in the bottleneck affects the performance. To this end, we put CE in different positions of a bottleneck in ResNet50, which consists of three ”Conv-BN-ReLU” basic blocks. The channel of the third block is 4 times than that of the second one. We compare the performance of CE-ResNet50 by putting CE in the second block (CE2-ResNet50) or the third block (CE3-ResNet50). The top-1 accuracy of CE3-ResNet-50 outperforms CE2-ResNet50 by 0.3 (i.e. 78.2 v.s. 77.9), which indicates that our CE block benefits from a larger number of channels. Secondly, we verify that CE should be stacked after the normalization layer. We show this by comparing two basic blocks (i.e. B1= ’conv+CE+Norm+ReLU’ and B2=’conv+Norm+CE+ReLU) by training VGGNet on CIFAR10. B1 and B2 obtain top-1 acc of 88.8% and 89.6%, implying that decorrelation should be performed after normalization layer as in CE.

5.4. Object Detection and Instance Segmentation on COCO

We assess the generalization of our CE block on detection/segmentation track using the COCO2017 dataset (Lin et al., 2014). We train our model on the union of 80k training images and 35k validation images and report the performance on the mini-val 5k images. Mask-RCNN is used as the base detection/segmentation framework. The standard COCO metrics of Average Precision (AP) for bounding box detection (APbb) and instance segmentation (APm) is used to evaluate our methods. In addition, we adopt two common training settings for our models, (1) freezing the vanilla BN and CE layer and (2) updating parameters with the synchronized version. For vanilla BN and CE layers, all the gamma, beta parameters, and the tracked running statistics are frozen. In contrast, for the synchronized version, the running mean and variance for batch normalization and the covariance for CE layers are computed across multiple GPUs. The gamma and beta parameters are updated during training while $\tilde{F}$ and λ are frozen to prevent overfitting. We use MMDetection training framework with ResNet50/ResNet101 as basic backbones and all the hyper-parameters are the same as (Chen et al., 2019). Table 5 shows the detection and segmentation results. The results show that compared with vanilla BN, our CE block can consistently improve the performance. For example, our fine-tuned CE-ResNet50 is 2.2 AP higher in detection and 2.7 AP higher in segmentation. For the sync BD version, CE-ResNet50 gets **42.0** AP in detection and **37.5** AP in segmentation, which is the best performance for ResNet50 to the best of our knowledge. To sum up, these experiments demonstrate the generalization ability of CE blocks in other tasks.

Backbone	AP^b	$AP^b_{.5}$	$AP^b_{.75}$	AP^m	$AP^m_{.5}$	$AP^m_{.75}$
ResNet50	38.6	59.5	41.9	34.2	56.2	36.1
+CE	40.8	**62.7**	44.3	36.9	59.2	39.4
+SyncCE	**42.0**	62.6	**46.1**	**37.5**	**59.5**	**40.3**
ResNet101	40.3	61.5	44.1	36.5	58.1	39.1
+CE	**41.6**	**62.8**	**45.8**	**37.4**	**59.4**	**40.0**

Table 5. Detection and segmentation results in COCO using Mask-RCNN We use the pretrained CE-ResNet50 model (78.2) and CE-ResNet101 (79.0) in ImageNet to train our model. CENet can consistently improve both box AP and segmentation AP by a large margin.

6. Conclusion

In this paper, we presented an effective and efficient network block, termed as Channel Equilibrium (CE). We show that CE encourages channels at the same layer to contribute equally to learned feature representation, enhancing the generalization ability of the network. Specifically, CE can be stacked between the normalization layer and the Rectified units, making it flexible to be integrated into various CNN architectures. The superiority of CE blocks has been demonstrated on the task of image classification and instance segmentation. We hope that the analyses of CE could bring a new perspective for future work in architecture design.

Acknowledgements Thank Xinjiang Wang, Yao Zhou, Chenyan Wu and Xiaobao Guo for their helpful discussions. Ping Luo is partially supported by the HKU Seed Funding for Basic Research, , Start-up Fund and Donation from SenseTime. This work was supported in part by SenseTime Group Limited and in part by the General Research Fund through the Research Grants Council of Hong Kong under Grants CUHK14202217, CUHK14203118, CUHK14205615, CUHK14207814, CUHK14208619, CUHK14213616.

References

Arpit, D., Zhou, Y., Kota, B. U., and Govindaraju, V. Normalization propagation: A parametric technique for removing internal covariate shift in deep networks. *arXiv preprint arXiv:1603.01431*, 2016.

Ba, J. L., Kiros, J. R., and Hinton, G. E. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

Bini, D. A., Higham, N. J., and Meini, B. Algorithms for the matrix pth root. *Numerical Algorithms*, 39(4):349–378, 2005.

Cao, Y., Xu, J., Lin, S., Wei, F., and Hu, H. Gcnet: Non-local networks meet squeeze-excitation networks and beyond. *arXiv preprint arXiv:1904.11492*, 2019.

Chen, K., Wang, J., Pang, J., Cao, Y., Xiong, Y., Li, X., Sun, S., Feng, W., Liu, Z., Xu, J., et al. Mmdetection: Open mmlab detection toolbox and benchmark. *arXiv preprint arXiv:1906.07155*, 2019.

Clevert, D.-A., Unterthiner, T., and Hochreiter, S. Fast and accurate deep network learning by exponential linear units (elus). *arXiv preprint arXiv:1511.07289*, 2015.

Cover, T. M. and Thomas, J. A. *Elements of information theory*. John Wiley & Sons, 2012.

Frankle, J. and Carbin, M. The lottery ticket hypothesis: Finding sparse, trainable neural networks. *arXiv preprint arXiv:1803.03635*, 2018.

Glorot, X., Bordes, A., and Bengio, Y. Deep sparse rectifier neural networks. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pp. 315–323, 2011.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

He, K., Gkioxari, G., Dollár, P., and Girshick, R. Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pp. 2961–2969, 2017a.

He, Y., Zhang, X., and Sun, J. Channel pruning for accelerating very deep neural networks. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 1389–1397, 2017b.

Higham, N. J. Newton's method for the matrix square root. *Mathematics of Computation*, 46(174):537–549, 1986.

Hu, J., Shen, L., and Sun, G. Squeeze-and-excitation networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 7132–7141, 2018.

Huang, L., Yang, D., Lang, B., and Deng, J. Decorrelated batch normalization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 791–800, 2018.

Huang, L., Zhou, Y., Zhu, F., Liu, L., and Shao, L. Iterative normalization: Beyond standardization towards efficient whitening. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4874–4883, 2019.

Ioffe, S. and Szegedy, C. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

Krizhevsky, A. Learning multiple layers of features from tiny images. *Technical report*, 2009.

Laufer, A., Leshem, A., and Messer, H. Game theoretic aspects of distributed spectral coordination with application to dsl networks. *arXiv preprint cs/0602014*, 2006.

Leshem, A. and Zehavi, E. Game theory and the frequency selective interference channel. *IEEE Signal Processing Magazine*, 26(5):28–40, 2009.

Lin, T.-Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., and Zitnick, C. L. Microsoft coco: Common objects in context. In *European conference on computer vision*, pp. 740–755. Springer, 2014.

Lu, L., Shin, Y., Su, Y., and Karniadakis, G. E. Dying relu and initialization: Theory and numerical examples. *arXiv preprint arXiv:1903.06733*, 2019.

Luo, P., Ren, J., Peng, Z., Zhang, R., and Li, J. Differentiable learning-to-normalize via switchable normalization. *arXiv preprint arXiv:1806.10779*, 2018.

Ma, N., Zhang, X., Zheng, H.-T., and Sun, J. Shufflenet v2: Practical guidelines for efficient cnn architecture design. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 116–131, 2018.

Maas, A. L., Hannun, A. Y., and Ng, A. Y. Rectifier nonlinearities improve neural network acoustic models. In *Proc. icml*, volume 30, pp. 3, 2013.

Mehta, D., Kim, K. I., and Theobalt, C. On implicit filter level sparsity in convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 520–528, 2019.

Miyato, T., Kataoka, T., Koyama, M., and Yoshida, Y. Spectral normalization for generative adversarial networks. *arXiv preprint arXiv:1802.05957*, 2018.

Morcos, A. S., Barrett, D. G., Rabinowitz, N. C., and Botvinick, M. On the importance of single directions for generalization. *arXiv preprint arXiv:1803.06959*, 2018.

Nair, V. and Hinton, G. E. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th international conference on machine learning (ICML-10)*, pp. 807–814, 2010.

Osborne, M. J. and Rubinstein, A. *A course in game theory*. MIT press, 1994.

Pan, X., Zhan, X., Shi, J., Tang, X., and Luo, P. Switchable whitening for deep representation learning. *Proceedings of the IEEE International Conference on Computer Vision*, 2019.

Pečarić, J. Power matrix means and related inequalities. *Mathematical Communications*, 1(2):91–110, 1996.

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., et al. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 115(3): 211–252, 2015.

Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., and Chen, L.-C. Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4510–4520, 2018.

Shao, W., Meng, T., Li, J., Zhang, R., Li, Y., Wang, X., and Luo, P. Ssn: Learning sparse switchable normalization via sparsestmax. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 443–451, 2019.

Simonyan, K. and Zisserman, A. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

Sun, Y., Wang, X., and Tang, X. Deeply learned face representations are sparse, selective, and robust. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2892–2900, 2015.

Ulyanov, D., Vedaldi, A., and Lempitsky, V. Instance normalization: The missing ingredient for fast stylization. *arXiv preprint arXiv:1607.08022*, 2016.

Wu, Y. and He, K. Group normalization. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 3–19, 2018.

Yang, J., Ren, Z., Gan, C., Zhu, H., and Parikh, D. Cross-channel communication networks. In *Advances in Neural Information Processing Systems*, pp. 1295–1304, 2019.

Yu, J., Yang, L., Xu, N., Yang, J., and Huang, T. Slimmable neural networks. *arXiv preprint arXiv:1812.08928*, 2018.

Zhang, C., Bengio, S., Hardt, M., Recht, B., and Vinyals, O. Understanding deep learning requires rethinking generalization. *arXiv preprint arXiv:1611.03530*, 2016.

Zhang, R., Peng, Z., Wu, L., Li, Z., and Luo, P. Exemplar normalization for learning deep representation. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2020.