CHANNEL EQUILIBRIUM NETWORKS

Anonymous authors
Paper under double-blind review

ABSTRACT

Convolutional Neural Networks (CNNs) typically treat normalization methods such as batch normalization (BN) and rectified linear function (ReLU) as building blocks. Previous work showed that this basic block would lead to channel-level sparsity (i.e. channel of zero values), reducing computational complexity of CNNs. However, over-sparse CNNs have many collapsed channels (i.e. many channels with undesired zero values), impeding their learning ability. This problem is seldom explored in the literature. To recover the collapsed channels and enhance learning capacity, we propose a building block, Channel Equilibrium (CE), which takes the output of a normalization layer as input and switches between two branches, batch decorrelation (BD) branch and adaptive instance inverse (AII) branch. CE is able to prevent implicit channel-level sparsity in both experiments and theory. It has several appealing properties. First, CE can be stacked after many normalization methods such as BN and Group Normalization (GN), and integrated into many advanced CNN architectures such as ResNet and MobileNet V2 to form a series of CE networks (CENets), consistently improving their performance. Second, extensive experiments show that CE achieves state-of-the-art results on various challenging benchmarks such as ImageNet and COCO. Third, we show an interesting connection between CE and Nash Equilibrium, a well-known solution of a non-cooperative game. The models and code will be released soon.

1 INTRODUCTION

Normalization is a useful technique for a wide range of learning tasks such as image classification (He et al., 2016; Ioffe & Szegedy, 2015), object detection (Ren et al., 2015; He et al., 2017a; Wu & He, 2018), and image generation (Goodfellow et al., 2014; Miyato et al., 2018). In recent years, we have witnessed a lot of effort to improve normalization, such as batch normalization (BN) (Ioffe & Szegedy, 2015), group normalization (GN) (Wu & He, 2018), and switchable normalization (SN) (Luo et al., 2018). These normalization methods are often used together with the ReLU activation function (Glorot et al., 2011; Nair & Hinton, 2010), which has become the most widely-used building block of modern CNNs, i.e. the 'BN+ReLU' block, and has been widely adopted in many advanced CNN architectures, such as Inception (Szegedy et al., 2017; 2016), ResNet (He et al., 2016), DenseNet (Huang et al., 2017), and ResNeXt (Xie et al., 2017).

However, recent work has disclosed a critical problem of 'BN+ReLU', known as channel collapse as investigated in (Mehta et al., 2019), which refers to the problem when certain channels become inactive and always output 0 value for any input. Such sparsity typically leads to reduced effective learning capacity of the network (Mehta et al., 2019; Lu et al., 2019). Common normalizers such as BN and IterNorm (Huang et al., 2019) rescale normalized/whitened features by learning additional parameters γ in a channel-wise way. The magnitude of γ implies the importance of different channels, which has been widely employed in network slimming (Liu et al., 2017; Yu et al., 2018) and channel pruning (He et al., 2017b). According to the lottery hypothesis (Frankle & Carbin, 2018), one over-parameterized network always contains unimportant channels that may become inactive (small γ) after training.

A key observation is that the dependency (covariance) matrix of features (denoted as Σ) after normalization is scaled by $\gamma\gamma^{\mathsf{T}}$. Therefore, a decorrelation operation ($\Sigma^{-\frac{1}{2}}$) can not only effectively eliminate the influence of the magnitude of γ, but even equalize the magnitude of resulting features in a feed-forward way (Barlow et al., 1961; Bengio & Bergstra, 2009). This operator enables all the channels to contribute equally to the feature representation learning process. Fig.1 shows decorrelating features after normalization using batch decorrelation (BD) can effectively prevent channel-level sparsity and improves the learning capacity of the network.

This work aims to alleviate the channel collapse problem by encouraging different channels to play an equal role in learning a feature representation. To this end, we introduce a building block, termed as Channel Equilibrium (CE), to conditionally decorrelate features after normalization layer and explicitly enhance the representation capability of a neural network in a feed-forward way.

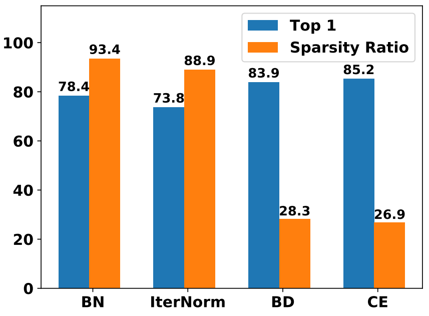

Figure 1: Top-1 accuracy and sparsity ratio[1] of batch normalization, iterative normalization (Huang et al. (2019)), the proposed batch decorrelation (BD) and channel equilibrium (CE) when training VGGNet (Simonyan & Zisserman, 2014) on CIFAR 10. Both BD and CE can effectively prevent channel-level sparsity.

As is shown in Fig.2b, CE takes the output of a normalization layer as input and switches between two complementary branches, including a branch of batch decorrelation and a branch of adaptive instance inverse. We will show that BD explicitly prevents channel-level sparsity by a batch-estimated covariance matrix, and AII helps CE learn preciser feature representation by learning an adaptive instance variance. As shown in Fig.1, equipped with CE, the VGGNet (Simonyan & Zisserman, 2014) is able to effectively mitigate undesired over-sparsity and improve recognition performance. Specifically, CE can be inserted between the normalization layer and the activation function, making it flexible to be integrated into many advanced CNN architectures such as ResNet50 and MobileNet V2. These networks can be upgraded into CE-Networks by replacing the 'BN-ReLU' building block using 'BN-CE-ReLU', only increasing the computational complexity marginally. Extensive experiments show that CENets consistently outperform their counterparts. For example, the CE-ResNet50 and CE-MobileNet V2 achieve 78.3% and 74.6% top-1 accuracy on ImageNet respectively, outperforming the plain networks by 1.7% and 2.1% with nearly the same FLOPs. We also show that CE with synchronization, which estimates the covariance matrix across multiple GPUs, increases the AP metric on the MS-COCO dataset to 42.0, surpassing its counterpart by 3.4.

Overall, the main **contributions** of this work are three-fold. First, we introduce an efficient feed-forward propagation that can largely prevent filter collapse and enhance representation capacity of CNNs. Second, CE blocks can be stacked after common normalization methods such as BN and GN and plugged into many advanced architectures, consistently improving their performance by a large margin. Third, CENet can be easily transferred to many other tasks like object detection/segmentation.

2 Related Work

Channel equalization. The success of the two most commonly used regularization techniques, i.e. BN (Ioffe & Szegedy, 2015) and Dropout (Srivastava et al., 2014), is attributed to channel or neuron equalization. For example, Mianjy et al. (2018) showed that Dropout makes the norm of incoming/outgoing weight vectors of all the hidden nodes equal, indicating a kind of equalization between neurons. Meanwhile, Morcos et al. (2018) pointed out that BN implicitly discourages single direction reliance. Intuitively, equalizing different channels enhances the generalization of learned feature representation. Note that the squeeze-and-excitation (SE) network (Hu et al., 2018) is a pioneer work that investigates network design by explicitly modeling interdependencies between channels. However, SE selectively emphasizes informative features and suppresses less useful ones. By contrast, this work proposes to explicitly expand representational power of all the channels by CE blocks, which, as will be shown, can be linked with Nash Equilibrium. More related work on sparsity in ReLU and normalization methods are provided in Sec.A of Appendix.

[1]The sparsity ratio is defined as the average percentage of values less than 1e-3 in the first six feature maps of VGGNet.

3 METHOD

In this section, we first review the normalization method and then introduce the proposed Channel Equilibrium (CE) block. CE contains two complementary branches, i.e. batch decorrelation (BD) and adaptive instance inverse (AII). We show how BD and AII benefit from each other through parameter γ and how CE is linked with Nash Equilibrium.

Notations. For CNNs, we use $x \in \mathbb{R}^{N\times C\times H\times W}$ to represent the feature in a layer, specifically, x_{ncij} denotes a pixel (i,j) in the c-th channel of the n-th sample. Sometimes, we ignore the subscript 'n' and denote it as x_{cij} for clarity of notation. $x_{nij} \in \mathbb{R}^C$ is obtained by stacking all elements in all channels of x_{ncij} into a column vector. $\mathrm{Diag}(\cdot)$ returns a matrix with the given diagonal and zero off-diagonal entries, and $\mathrm{diag}(\cdot)$ extracts the diagonal of the given matrix. $\gamma, \beta \in \mathbb{R}^C$ are normalization parameters.

3.1 OVERVIEW OF NORMALIZATION

Normalization is usually employed after convolution layers to stabilize the training of CNNs. Given a feature $x \in \mathbb{R}^{N\times C\times H\times W}$, a normalizer first standardizes it to $\bar{x}$, and then maps it to $\tilde{x}$ by an affine transformation as written in the following,

$$\tilde{x}_{ncij} = \gamma_c \bar{x}_{ncij} + \beta_c, \qquad \bar{x}_{ncij} = (x_{ncij} - \mu_s)/\sigma_s \tag{1}$$

where $s \in \Omega = \{\mathrm{IN}, \mathrm{BN}, \mathrm{LN}, \cdots\}$ indicates a normalizer and μ_s, σ_s are the mean and standard deviation of the given normalizer. From Eqn.(1), we claim that normalization would result in an unequal feature representation in a channel basis. To see this, firstly we know that common-used normalizers like IN and BN are performed channel-wisely. Consequently, dependencies between channels are not considered, resulting in weak correlation between some channels. In addition, normalization parameters γ and β are usually computed channel-wise and the magnitude of them indicates the importance of channels. Previous work (Frankle & Carbin, 2018; Mehta et al., 2019) revealed that channel level sparsity emerges with small-values of γ. Typically, the degradation of some channels causes reduced effective learning capacity of the network (Mehta et al., 2019; Lu et al., 2019). To mitigate such disequilibrium between channels, we propose a building block in the next section, namely Channel Equilibrium (CE).

3.2 CHANNEL EQUILIBRIUM (CE) BLOCK

A Channel Equilibrium (CE) block is a computational unit which aims to equalize feature representation capacity among channels. Towards this goal, the idea of decorrelation is adopted. Previous methods (Huang et al., 2018; 2019) decorrelate features by a single batch estimated covariance matrix Σ. However, as revealed in existing work, channel dependency is specific to each input (Hu et al., 2018). Inspired by this, we bring in an adaptive instance variance, S_n, on the diagonal of the covariance matrix Σ,

$$D_n = \lambda\Sigma + (1-\lambda)\mathrm{Diag}(S_n), \qquad S_n = F(\sigma^2(\tilde{x}_n)), \tag{2}$$

where the subscript n is the sample index, $\lambda \in (0,1)$ is a trainable ratio used to switch between batch and instance statistics, $F : \mathbb{R}^C \to \mathbb{R}^C$ is a transformation conditioned on the current input $\tilde{x}$ and $\sigma^2(\tilde{x}_n)$ computes instance variance of $\tilde{x}_n$ within each channel. On the issue of channel disequilibrium as discussed in sec.3.1, CE block works by decorrelating feature maps after normalization using $D_n^{-\frac{1}{2}}$. We further utilize the Jensen inequality for matrix functions (Pečarić, 1996) to obtain a relaxed decorrelation operator $D_n^{-\frac{1}{2}}$:

$$D_n^{-\frac{1}{2}} = \left[\lambda\Sigma + (1-\lambda)\mathrm{Diag}(S_n)\right]^{-\frac{1}{2}} \preceq \lambda\Sigma^{-\frac{1}{2}} + (1-\lambda)\left[\mathrm{Diag}(S_n)\right]^{-\frac{1}{2}}, \tag{3}$$

where $A \preceq B$ indicates $B - A$ is semi-definite positive. We introduce this relaxation for the following two reasons. (1) It reduces computation in the training stage, because the relaxed form only needs to calculate the inverse of square root $\Sigma^{-\frac{1}{2}}$ once, and the other branch $\mathrm{Diag}(S_n)^{-\frac{1}{2}}$ is easy to compute. (2) It makes inference fast, since $\Sigma^{-\frac{1}{2}}$ is a moving-average statistic in inference and can be absorbed into previous layer. Note that Eqn.(3) transforms the combination of covariance and adaptive instance variance into the combination of their inverse square roots.

In the following, we refer $\Sigma^{-\frac{1}{2}}$ in Eqn.(3) as batch decorrelation (BD) and refer $[\mathrm{Diag}(S_n)]^{-\frac{1}{2}}$ as adaptive instance inverse (AII). The former decorrelates channels by a batch covariance, while the latter adjusts the extend of inverse for each channel and instance in an adaptive manner. Integrating both of them yields the forward representation of CE block:

$$p_{nij} = D_n^{-\frac{1}{2}}(\mathrm{Diag}(\gamma)\bar{x}_{nij} + \beta) \tag{4}$$

where $p_{nij} \in \mathbb{R}^C$ denotes the output of CE, as illustrated in Fig.2b. Since CE is performed after the normalization layer, we take BN as an example to introduce these two branches in the following sections.

3.2.1 Batch Decorrelation (BD)

Although a lot of previous work (Huang et al., 2018; 2019; Pan et al., 2019) has investigated whitening using covariance matrices, all of them are applied after the convolution layer and thus cannot prevent filter-level sparsity, as shown in Fig.1. Instead, we apply decorrelation after the normalization layer to address channel collapse. Consider a tensor $\tilde{x}$ after a BN layer, we reshape it to $\tilde{x} \in \mathbb{R}^{C\times M}$ and $M = N \cdot H \cdot W$. Then the covariance matrix Σ of $\tilde{x}$ can be written as (details are presented in Sec. B of Apeendix)

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

where $\bar{x}$ is a standardized feature with zero mean and unit variance and $\odot$ indicates elementwise multiplication. Eqn.(5) implies that the covariance matrix Σ of $\tilde{x}$ can be decomposed into two parts. The first part depends on normalization parameter γ and the second part becomes correlation matrix of $\tilde{x}$. We observe that Σ_{ij}, which represents dependency between i-th channel and j-th channel, is scaled by $\gamma_i\gamma_j$ after a BN layer is applied.

The Batch Decorrelation branch requires computing $\Sigma^{-\frac{1}{2}}$, which is usually related to eigen-decomposition or SVD and involves heavy computation (Huang et al., 2018). Instead, here we adopt an efficient approach, i.e., Newton's Iteration to obtain $\Sigma^{-\frac{1}{2}}$ (Bini et al., 2005; Higham, 1986). Given covariance matrix Σ, Newton's Iteration calculates $\Sigma^{-\frac{1}{2}}$ by the following iterations:

$$\begin{cases} \Sigma_0 = I \\ \Sigma_k = \frac{1}{2}(3\Sigma_{k-1} - \Sigma_{k-1}^3\Sigma),\ k = 1, 2, \cdots, T. \end{cases} \tag{6}$$

where T is the iteration number ($T = 3$ in our experiments). Note that the convergence of Eqn.(6) is guaranteed if $\|\Sigma\|_2 < 1$ (Bini et al., 2005). To this end, we normalize Σ as $\Sigma/\mathrm{tr}(\Sigma)$ where $\mathrm{tr}(\cdot)$ is the trace operator (Huang et al., 2019). In this way, the normalized covariance matrix is written as $\Sigma = \frac{\gamma\gamma^{\mathsf{T}}}{\|\gamma\|_2^2} \odot \frac{1}{M}\bar{x}\bar{x}^{\mathsf{T}}$. To sum up, the batch decorrelation branch firstly calculates a normalized covariance matrix and then applies Newton's Iteration to obtain its inverse square root, reducing much computational cost compared with SVD decomposition in the training stage. Furthermore, BD branch can be merged into convolutional layers in the inference stage, which adds no extra computation.

3.2.2 Adaptive instance Inverse (AII)

Channel dependencies are specific to each sample. Consequently, a conditional decorrelation is desired for each sample. The adaptive instance inverse (AII) branch only uses diagonal entries to model channel dependencies, as shown in Eqn.(2). Since a diagonal matrix can be inverted easily, this approach can avoid the computation of Eqn.(3).

To construct the AII branch, we analyze its input (the output of a BN layer), which is formulated as $\tilde{x}_{ncij} = \gamma_c\bar{x}_{ncij} + \beta_c$. The input of AII is the instance variance of each channel (details are provided in Appendix Sec. B),

$$\sigma_{nc}^2 = \frac{\gamma_c^2(\sigma_{\mathrm{IN}}^2)^{nc}}{(\sigma_{\mathrm{BN}}^2)^c} \tag{7}$$

where σ_{IN}^2 and σ_{BN}^2 represent the variances in IN and BN respectively. The ratio of them measures the relative fluctuation of how much the instance statistic are deviated from the batch-estimated statistic. Similar to Eqn.(5), the input of AII is also scaled by γ_c^2. The AII branch takes σ_{nc}^2 as

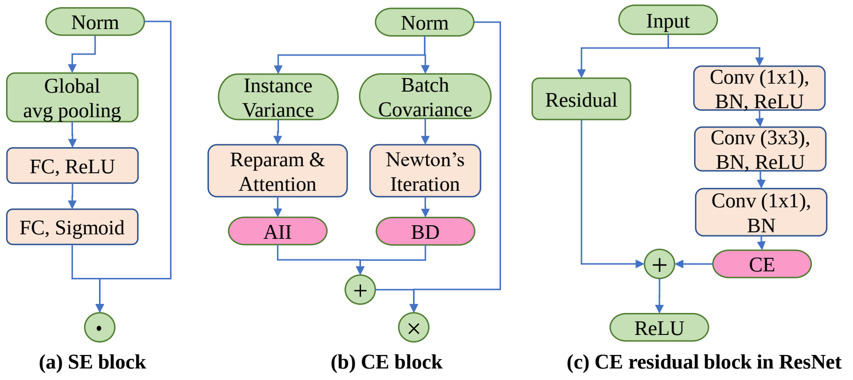

Figure 2: Illustrations of SE block (Hu et al., 2018), CE block and CE residual block in ResNet. $\odot$ denotes broadcast element-wise multiplication, $\bigoplus$ denotes broadcast elementwise addition and $\otimes$ denotes matrix multiplication. (b) shows CE has two lightweight branches, BN and AII. (c) shows CE can be easily stacked into many advanced networks such as ResNet with merely small extra computation.

input and computes an adaptive instance inverse, i.e. $[\text{Diag}(S_n)]^{-\frac{1}{2}}$. Here, a reparameterization trick is employed to generate adaptive instance inverse such that AII has the same philosophy as inverse square root of variance or covariance. Let s be an estimate of variance, the AII branch can be reparameterized as below,

$$[\text{Diag}(S_n)]^{-\frac{1}{2}} = \text{Diag}(\tilde{F}(\sigma_n^2)) \cdot s^{-\frac{1}{2}}, \tag{8}$$

$$\tilde{F}(\sigma_n^2) = \delta_2(W_2\delta_1(\text{LN}(W_1\sigma_n^2))), \;\; s = \sigma^2(\tilde{x}), \tag{9}$$

where δ_1 and δ_2 are ReLU and sigmoid activation function respectively, $W_1 \in \mathbb{R}^{\frac{C}{r}\times C}$ and $W_2 \in \mathbb{R}^{C\times\frac{C}{r}}$ and r is reduction ratio, $s \in \mathbb{R}$ denotes variance of all elements in $\tilde{x}$, which is a batch statistic in training and is obtained using moving average in inference. $\tilde{F}(\sigma_n^2) \in (0,1)^C$ is treated as a gating mechanism in order to control the strength of instance inverse for each channel. Like (Hu et al. (2018); Cao et al. (2019)), we also use a bottleneck architecture to limit model complexity. We add layer normalization (LN) inside the bottleneck transform (before ReLU) to ease optimization. It is seen from Eqn.(8) that $s^{-\frac{1}{2}}$ represents the quantity of inverse square root of variance and $\tilde{F}(\sigma_n^2)$ regulates the extend of variance inverse. Basically, $\tilde{F}$ maps the instance variance to a set of channel weights. In this sense, the AII branch intrinsically introduces dynamics conditioned on each input.

3.3 DISCUSSIONS

Instantiations. Our CE block can be integrated into various advanced architectures, such as ResNet, VGGNet, ShuffleNet or MobileNet, by inserting it between the normalization layer and ReLU non-linearity. We first describe the basic CE block in Fig.2b. As discussed earlier, CE processes incoming features after the normalization layer by combining two branches, i.e. batch decorrelation (BD) and adaptive instance inverse (AII). Compared with SE block in Fig.2a, our proposed CE clock combines both instance and batch statistics, and it can consequently model dependencies between channels better.

We can construct a series of CENets by integrating our CE block into various networks. 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 unit is employed in the last '1×1' convolution layer by plugging the CE module before ReLU non-linearity, as shown in Fig.2c. As for CE-MobileNet V2, since the last '1×1' convolution layer in the bottleneck of MobileNet V2 is not followed by a ReLU activation, we insert CE after the normalization layer and before the ReLU6 non-linearity of the '3×3' convolution layer. Following similar strategies, CE is further integrated into ShuffleNet V2 to construct CE-ShuffleNet V2. We provide extensive experiments evaluating all these CENets in Sec.4 and analysis of computation details in training and inference in Sec.F of Appendix.

	ResNet18			ResNet50			ResNet101		
	Baseline	SE	CE	Baseline	SE	CE	Baseline	SE	CE
Top-1	70.4	71.4	**71.9**	76.6	77.6	**78.3**	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.

Equivalent γ. This section shows how BD and AII benefit from each other through parameter γ. First, we disclose the mechanism in preventing channel-level sparsity behind the BD branch. Previous work (He et al., 2017b; Yu et al., 2018; Frankle & Carbin, 2018) revealed that γ in BN can be used to identify some unimportant channels, implying that the representational power of feature map largely depends on the magnitude of γ. Note that an equivalent scale parameter in the BD branch can be expressed as $\hat{\gamma} = \Sigma_T^{-\frac{1}{2}}\gamma$. The proposition 1 shows that BD explicitly increases the magnitude of $\hat{\gamma}$ in a feed-forward way, improving representational capacity in a channel basis. We present the proof of proposition 1 in Sec.C of Appendix. Furthermore, the original γ in BN is also implicitly enlarged. As discussed in Eqn.(5), a sufficient small γ_i can cause degradation of the covariance matrix and then the convergence of Newton's iteration (Bini et al., 2005) cannot be guaranteed. As a result, once the network converges, γ is not supposed to degrade. This will in turn bring many benefits to the AII branch. As shown in Eqn.(7), the input of AII is proportional to γ, meaning that the features fed into the AII branch are enlarged as γ increases. In this way, a bottleneck architecture in AII can learn more compact global information and model channel dependencies better.

Proposition 1. *Let Σ be covariance matrix of feature maps after batch normalization. Assume that $\Sigma_k = \Sigma^{-\frac{1}{2}}$, $\forall k = 2, 3, \cdots, T$, then $\|\hat{\gamma}\|_1 > \|\gamma\|_1$. Especially, we have $|\hat{\gamma_i}| > |\gamma_i|$.*

Connection with Nash Equilibrium. We show an interesting connection between the proposed CE block and the well-known Nash Equilibrium in game theory (Leshem & Zehavi, 2009). To be specific, we bring novel insights on normalization from an optimization perspective. Suppose each channel computes its output by maximizing capacity available to itself under some constraints. Especially, we restrict that each channel has a maximum budget and all the outputs are non-negative. Further, if we consider dependencies between 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). In Sec.D of Appendix, we present the detailed construction of Gaussian interference game in the context of CNNs. It is worth noting that when all the outputs are active (larger than 0), this Nash Equilibrium solution has an explicit expression. Under some mild approximations, it can be shown that the explicit Nash Equilibrium solution can surprisingly match the representation of CE in Eqn.(4). It shows that decorrelating features after normalization layer can be connected with Nash Equilibrium, implying that the proposed CE block performs a mechanism on channel equalization. We present detailed explanations about the connection between CE and Nash Equilibrium in Sec.D of Appendix.

4 EXPERIMENTS

We evaluate our methods on two basic vision tasks, image classification on ImageNet and object detection/segmentation on COCO, where we demonstrate the effectiveness of the CE block.

4.1 IMAGE CLASSIFICATION ON IMAGENET

We first evaluate CE on the ImageNet benchmark. The training details are illustrated in Sec.F of Appendix.

	MobileNet V2			ShuffleNet V2 $0.5\times$			ShuffleNet V2 $1\times$		
	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, MobileNet V2 and ShuffleNet V2, 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.

	BN		GN		IN		LN	
	top-1	top-5	top-1	top-5	top-1	top-5	top-1	top-5
Baseline	76.6	93.0	75.6	92.8	74.2	91.9	71.6	89.9
Baseline+CE	**78.3**	**94.1**	**76.2**	**92.9**	**76.0**	**92.7**	**73.3**	**91.3**
Increase	+1.7	+1.1	+0.6	+0.1	+1.8	+0.8	+1.7	+1.4

Table 3: CE improves top-1 and top-5 accuracy of various normalization methods on ImageNet with ResNet50 as backbone.

Performance comparison on ResNet. We evaluate on representative residual network structures including ResNet18, ResNet50 and ResNet101. The CE-ResNet is then 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.

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%, 1.7% and 1.0% compared with the corresponding plain ResNet architectures. The CE-ResNet50 even outperforms the plain ResNet101 (78.0). We plot training and validation loss during the training process for ResNet50, SE-ResNet50 and CE-ResNet50 in Sec.E of Appendix.

We also analyze the complexity of BN, SE, and CE in terms of GFLOPs, GPU and CPU running time. We evaluate the inference time[2] with a mini-batch of 32. In term of GFLOPs, the CE-ResNet18, CE-ResNet50, CE-ResNet101 has only 0.242% and 0.241% relative increase in GFLOPs compared with plain ResNet. Additionally, the CPU and GPU inference time of CENet is nearly the same with SENet.

Performance comparison on light-weight networks. We further verify the effectiveness of our proposed CE in two representative light-weight networks, MobileNet V2 and ShuffleNet V2. The results of comparison are listed in Table 2. It is seen that CE blocks bring conspicuous improvements in performance at a minimal increase in computational burden on mobile settings. For MobileNet V2, we see that CE blocks even improves top-1 accuracy of baseline by 2.1%.

Other Normalizers. In addition to BN, CE is also effective for other normalization technologies, since the channel-wise affine transformation is employed in any well-known normalizers. To prove this, we conduct experiments using ResNet-50 under different normalizers including batch normalization (BN), group normalization (GN), instance normalization (IN), and layer normalization (LN). For these experiments, we stack CE block after the above normalizers to see whether CE helps other normalization methods. As shown in Table 3, our CE generalize well over different normalization technology, improving the performance by 0.6-1.8 top-1 accuracy.

4.2 Analysis of CE

In this section, we first demonstrate that CE is able to equalize the importance of all channels and then analyze the effects of BD and AII branches separately on CIFAR10 and ImageNet datasets.

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

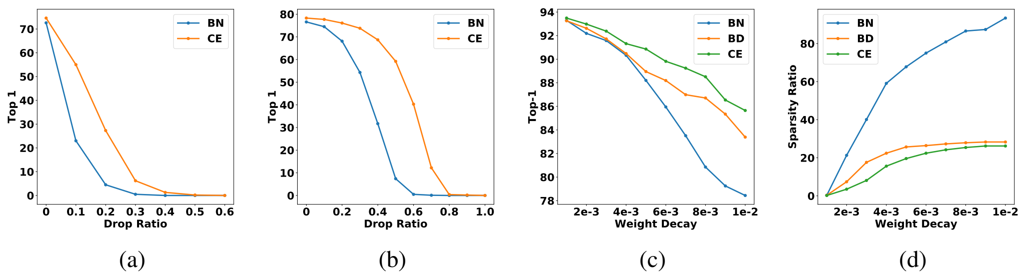

Figure 3: (a) and (b) show dropout ratios versus top-1 accuracy for MobileNet V2 and ResNet50 respectively. We randomly drop some channels in the first normalization layers. We can see that when dropping the same number of channels, the top-1 accuracy of our CE-ResNet50 and CE-MobileNet V2 is consistently higher than plain networks. For (c) and (d), we train VGGNet in CIFAR-10 under different weight decays. It is observed that the sparsity ratio is highly linked with top-1 accuracy. Our CE-VGG can reduce sparsity ratio, and thus reduce accuracy drop.

CE is effective in channel equalization. Here, we verify whether CE can equalize channels by an ablation approach used in 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). In this regard, the more reliant a network is on a small set of channels, the more quickly the accuracy will drop if those channels are ablated. On the contrary, if the importance of channels to the network's computation are more equal, the accuracy will drop more gently. With this powerful technique, we see how ResNet50 and MobileNet V2 with CE blocks respond to cumulative random ablation of channels. We plot the ablation ratio versus the top-1 accuracy in Fig.3a and Fig.3b. As we can see, our CE block is able to resist the cumulative random ablation of channels on both ResNet50 and MobileNet V2, showing that CE can effectively equalize the importance of channels. For example, the top-1 accuracy of our CE-ResNet50 is 1.7 higher than the original ResNet50 if no channels are ablated, but when 60% channels are ablated, CE-ResNet50 still obtain 40.5 top-1 accuracy, while the original ResNet50 gets only 0.5 top-1 accuracy.

BD is able to mitigate channel-level sparsity. As proved in Proposition 1, equivalent γ in the BD branch is explicitly enlarged, leading to the expansion of representational power of all channels. Here we investigate this property experimentally with a single BD branch. The layer-wise sparsity ratios are measured by the percentage of γ whose values are less than 1e-3. Fig.3d shows sparsity ratio under a wide range of weight decay for BN, BD and CE. It is observed that the top-1 accuracy of VGGNet with BN drops significantly as the weight decay increases, but BD can reduce accuracy drop. For example, when the weight decay is 1e-3, the top-1 of BD is only 0.1 higher than BN, but when the weight decay reaches to 1e-2, BD is 4.95 higher. Moreover, the sparsity ratio of CE is even lower than BD while the top-1 accuracy is higher, which can demonstrate that CE can strengthen the effect of sparsity alleviation compared with single BD.

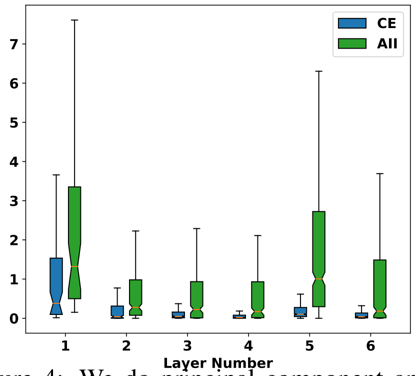

Figure 4: We do principal component analysis (PCA) on the input of AII sub-network, the variance of each channel. This figure show the box chart of principal components. CENet has lower means and variances than AIINet, indicating the input of AII sub-network in CENet is more equal and informative.

AII helps CE learn preciser feature representation. First, as discussed in Sec.3.3, AII benefits from BD such that the features fed into AII branch are more informative. To see this, we train ResNet50 with a single AII or CE branch, termed AII-ResNet50 or CE-ResNet50. We do principal component analysis (PCA) on the inputs of AII branch in AII-ResNet50 and the counterpart in CE-ResNet50 and plot the box chart of principal components. As is shown in Fig.4, the input of AII

Method	Plain ResNet50	BD-ResNet50	AII-ResNet50	CE-ResNet50
top-1	76.6	77.0 (+0.4)	77.3 (+0.7)	**78.3 (+1.7)**

Table 4: Results of batch covariance decorrelation, adaptive variance inverse and channel equilibrium. We use ResNet-50 as the basic structure. The top-1 accuracy increase (1.7) of CE-ResNet is higher than combined top-1 accuracy increase (1.1) of BD-ResNet and AII-ResNet, indicating the effects of BD and AII branch is complementary.

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-ResNet50	40.8	**62.7**	44.3	36.9	59.2	39.4
SyncCE-ResNet50	**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-ResNet101	**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 (78.3) 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.

branch in CE-ResNet50 gets much lower means and variances, meaning that the input feature has more valid basis and thus more informative. More analysis is provided in Sec.E of Appendix.

BD and AII are complementary. Here, we verify that BD and AII are complementary to each other. We train plain ResNet50, BD-ResNet50, AII-ResNet50, and CE-ResNet50 for comparison. The top-1 accuracy is reported in Table 4. It is observed that the BD-ResNet50 and AII-ResNet50 are 0.4 and 0.7 higher than the plain ResNet-50 respectively. However, when they are combined, the top-1 accuracy improves by 1.7, higher than combined accuracy increase (1.1), which demonstrates that they benefit from each other.

4.3 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 batch normalization and channel equilibrium 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). Fig.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.

5 Conclusion

In this paper, we presented a novel network block, termed as Channel Equilibrium (CE). The CE block conditionally decorrelates feature maps after normalization layer by switching between batch decorrelation branch and adaptive instance inverse branch. In both experiment and theory, we show that CE is able to explicitly enhance the representation capability of a neural network in a feed-forward way. Specifically, CE can be stacked between the normalization layer and the ReLU function, making it flexible to be integrated into many advanced CNN architectures. The superiority of CE blocks has been demonstrated on the task of image classification and instance segmentation. We hope that the analysis of channel equalization in CE could bring a new perspective for future work in architecture design.

REFERENCES

Horace B Barlow et al. Possible principles underlying the transformation of sensory messages. *Sensory communication*, 1:217–234, 1961.

Yoshua Bengio and James S Bergstra. Slow, decorrelated features for pretraining complex cell-like networks. In *Advances in neural information processing systems*, pp. 99–107, 2009.

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

Yue Cao, Jiarui Xu, Stephen Lin, Fangyun Wei, and Han Hu. Gcnet: Non-local networks meet squeeze-excitation networks and beyond. *arXiv preprint arXiv:1904.11492*, 2019.

Kai Chen, Jiaqi Wang, Jiangmiao Pang, Yuhang Cao, Yu Xiong, Xiaoxiao Li, Shuyang Sun, Wansen Feng, Ziwei Liu, Jiarui Xu, et al. Mmdetection: Open mmlab detection toolbox and benchmark. *arXiv preprint arXiv:1906.07155*, 2019.

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

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

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pp. 2672–2680, 2014.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pp. 1026–1034, 2015.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pp. 2961–2969, 2017a.

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

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

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

Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4700–4708, 2017.

Lei Huang, Dawei Yang, Bo Lang, and Jia Deng. Decorrelated batch normalization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 791–800, 2018.

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

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

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

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

Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pp. 740–755. Springer, 2014.

Zhuang Liu, Jianguo Li, Zhiqiang Shen, Gao Huang, Shoumeng Yan, and Changshui Zhang. Learning efficient convolutional networks through network slimming. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 2736–2744, 2017.

Lu Lu, Yeonjong Shin, Yanhui Su, and George Em Karniadakis. Dying relu and initialization: Theory and numerical examples. *arXiv preprint arXiv:1903.06733*, 2019.

Ping Luo, Jiamin Ren, Zhanglin Peng, Ruimao Zhang, and Jingyu Li. Differentiable learning-to-normalize via switchable normalization. *arXiv preprint arXiv:1806.10779*, 2018.

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

Dushyant Mehta, Kwang In Kim, and Christian Theobalt. 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.

Poorya Mianjy, Raman Arora, and Rene Vidal. On the implicit bias of dropout. *arXiv preprint arXiv:1806.09777*, 2018.

Takeru Miyato, Toshiki Kataoka, Masanori Koyama, and Yuichi Yoshida. Spectral normalization for generative adversarial networks. *arXiv preprint arXiv:1802.05957*, 2018.

Ari S Morcos, David GT Barrett, Neil C Rabinowitz, and Matthew Botvinick. On the importance of single directions for generalization. *arXiv preprint arXiv:1803.06959*, 2018.

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

Xingang Pan, Xiaohang Zhan, Jianping Shi, Xiaoou Tang, and Ping Luo. Switchable whitening for deep representation learning. *Proceedings of the IEEE International Conference on Computer Vision*, 2019.

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

Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pp. 91–99, 2015.

Ramprasaath R Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. Grad-cam: Visual explanations from deep networks via gradient-based localization. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 618–626, 2017.

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

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958, 2014.

Yi Sun, Xiaogang Wang, and Xiaoou Tang. 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.

Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2818–2826, 2016.

Christian Szegedy, Sergey Ioffe, Vincent Vanhoucke, and Alexander A Alemi. Inception-v4, inception-resnet and the impact of residual connections on learning. In *Thirty-First AAAI Conference on Artificial Intelligence*, 2017.

Dmitry Ulyanov, Andrea Vedaldi, and Victor Lempitsky. Instance normalization: The missing ingredient for fast stylization. *arXiv preprint arXiv:1607.08022*, 2016.

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

Saining Xie, Ross Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1492–1500, 2017.

Jiahui Yu, Linjie Yang, Ning Xu, Jianchao Yang, and Thomas Huang. Slimmable neural networks. *arXiv preprint arXiv:1812.08928*, 2018.

Appendix

A More 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 PReLU (He et al., 2015) and Leaky ReLU (Maas et al., 2013). Recently, Lu et al. (2019) also tried to solve this problem by modifying initialization scheme. Different from these work, we focus on filter level sparsity and explicitly prevent it in a feed-forward way by the proposed CE blocks.

Normalization and decorrelation. There are many practices on normalizer development, such as Batch Normalization (BN) (Ioffe & Szegedy, 2015), Group normalization (GN) (Wu & He, 2018) and Switchable Normalization (Luo et al., 2018). 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 switchable whitening (Pan et al., 2019). Despite their success in performance improvement, little is known about their implicit sparsity. Fig.1 shows that channel-level sparsity emerges in VGGNet where 'BN+ReLU' or 'IterNorm+ReLU' are used. Unlike previous decorrelated normalizations where decorrelation operation is applied after a convolution layer, our CE explicitly decorrelates features after normalization. As shown in Fig.1, both CE and BD are effective to alleviate such sparsity.

B Computation details in 'BN-CE-ReLU' block

As discussed before, CE processes incoming features after normalization layer by combining two branches, i.e. batch decorrelation and adaptive instance inverse. The former computes a covariance matrix and the latter calculates instance variance. We now take 'BN-CE-ReLU' block as an example to show the computation details of statistics in it. Given a tensor $x \in \mathbb{R}^{N\times C\times H\times W}$, the mean and variance in IN (Ulyanov et al., 2016) are calculated as:

$$\mu_{\text{IN}}^{nc} = \frac{1}{HW}\sum_{i,j}^{H,W} x_{ncij}, \quad (\sigma_{\text{IN}}^2)^{nc} = \frac{1}{HW}\sum_{i,j}^{H,W}(x_{ncij} - \mu_{\text{IN}}^{nc})^2 \tag{10}$$

Hence, we have $\mu_{\text{IN}}, \sigma_{\text{IN}}^2 \in \mathbb{R}^{N\times C}$. Then, the statistics in BN can be reformulated as follows:

$$\begin{aligned}
\mu_{\text{BN}}^{c} &= \frac{1}{NHW}\sum_{n,i,j}^{N,H,W} x_{ncij} = \frac{1}{N}\sum_{i}^{N}\frac{1}{HW}\sum_{i,j}^{H,W} x_{ncij} \\
(\sigma_{\text{BN}}^2)^{c} &= \frac{1}{NHW}\sum_{n,i,j}^{N,H,W}(x_{ncij} - \mu_{\text{BN}}^{c})^2 \\
&= \frac{1}{N}\sum_{n}^{N}\frac{1}{HW}\sum_{i,j}^{H,W}(x_{ncij} - \mu_{\text{IN}}^{nc} + \mu_{\text{IN}}^{nc} - \mu_{\text{BN}}^{c})^2 \\
&= \frac{1}{N}\sum_{n}^{N}(\frac{1}{HW}\sum_{i,j}^{H,W}(x_{ncij} - \mu_{\text{IN}}^{nc})^2 + (\mu_{\text{IN}}^{nc} - \mu_{\text{BN}}^{c})^2) \\
&= \frac{1}{N}\sum_{n}^{N}(\sigma_{\text{IN}}^2)^{nc} + \frac{1}{N}\sum_{n}^{N}(\mu_{\text{IN}}^{nc} - \mu_{\text{BN}}^{c})^2)
\end{aligned} \tag{11}$$

Then, we have $\mu_{\text{BN}} = \mathbb{E}[\mu_{\text{IN}}]$ and $\sigma_{\text{BN}}^2 = \mathbb{E}[\sigma_{\text{IN}}^2] + \text{D}[\mu_{\text{IN}}]$, where $\mathbb{E}[\cdot]$ and $\text{D}[\cdot]$ denote expectation and variance operators over N samples. Further, the input of AII is instance variance of features after BN, which can be calculated as follows:

$$\begin{aligned}\sigma_{nc}^2 &= \frac{1}{HW}\sum_{i,j}^{H,W}\left[(\gamma_c\frac{x_{ncij}-\mu_{\text{BN}}^c}{\sigma_{\text{BN}}^c}+\beta_c)-(\gamma_c\frac{\mu_{\text{IN}}^{nc}-\mu_{\text{BN}}^c}{\sigma_{\text{BN}}^c}+\beta_c)\right]^2 \\ &= \frac{\gamma_c^2}{(\sigma_{\text{BN}}^2)^c}\frac{1}{HW}\sum_{i,j}^{H,W}(x_{ncij}-\mu_{\text{IN}}^{nc})^2 \\ &= \frac{\gamma_c^2(\sigma_{\text{IN}}^2)^{nc}}{(\sigma_{\text{BN}}^2)^c}\end{aligned} \tag{12}$$

At last, the output of BN is $\tilde{x}_{ncij} = \gamma_c\bar{x}_{ncij}+\beta_c$, then the entry in c-th row and d-th column of covariance matrix Σ of $\tilde{x}$ is calculated as follows:

$$\Sigma_{cd} = \frac{1}{NHW}\sum_{n,i,j}^{N,H,W}(\gamma_c\bar{x}_{ncij})(\gamma_d\bar{x}_{ndij}) = \gamma_c\gamma_d\rho_{cd} \tag{13}$$

where ρ_{cd} is the element in c-th row and j-th column of correlation matrix of $\bar{x}$. Thus, we can write Σ into vector form: $\Sigma = \gamma\gamma^{\mathsf{T}} \odot \frac{1}{M}\bar{x}\bar{x}^{\mathsf{T}}$ if we reshape $\tilde{x}$ to $\tilde{x} \in \mathbb{R}^{C\times M}$ and $M = N\cdot H\cdot W$.

C Proof of proposition 1

Proposition 1. *Let Σ be covariance matrix of feature maps after batch normalization. Assume that $\Sigma_k = \Sigma^{-\frac{1}{2}}$, $\forall k = 2,3,\cdots,T$, then $\|\hat{\gamma}\|_1 > \|\gamma\|_1$. Especially, we have $|\hat{\gamma_i}| > |\gamma_i|$*

Proof. Since $\Sigma_k = \Sigma^{-\frac{1}{2}}$, $\forall k = 2,3,\cdots,T$, we have $\Sigma_k\gamma = \frac{1}{2}\Sigma_{k-1}(3I-\Sigma_{k-1}^2\Sigma)\gamma = \Sigma_{k-1}\gamma$. Therefore, we only need to show $\|\hat{\gamma}\|_1 = \|\Sigma_T\gamma\|_1 = \cdots = \|\Sigma_2\gamma\|_1 \geq \|\gamma\|_1$. Now, we show that for $k=2$ we have $\left\|\frac{1}{2}(3I-\Sigma)\gamma\right\|_1 \geq \|\gamma\|_1$. From Eqn.(5), we know that $\Sigma = \frac{\gamma\gamma^{\mathsf{T}}}{\|\gamma\|_2^2}\odot\rho$ where ρ is the correlation matrix of $\tilde{x}$ and $-1 \leq \rho_{ij} \leq 1$, $\forall i,j \in [C]$. Then, we have

$$\begin{aligned}\frac{1}{2}(3I-\Sigma)\gamma &= \frac{1}{2}(3I-\frac{\gamma\gamma^{\mathsf{T}}}{\|\gamma\|_2^2}\odot\rho)\gamma \\ &= \frac{1}{2}(3\gamma-(\frac{\gamma\gamma^{\mathsf{T}}}{\|\gamma\|_2^2}\odot\rho)\gamma) \\ &= \frac{1}{2}(3\gamma-\frac{1}{\|\gamma\|_2^2}\left[\sum_j^C\gamma_1\gamma_j\rho_{1j}\gamma_j,\sum_j^C\gamma_2\gamma_j\rho_{2j}\gamma_j,\cdots,\sum_j^C\gamma_C\gamma_j\rho_{Cj}\gamma_j\right]^{\mathrm{T}}) \\ &= \frac{1}{2}(3\gamma-\frac{1}{\|\gamma\|_2^2}\left[\sum_j^C\gamma_1\gamma_j\rho_{1j}\gamma_j,\sum_j^C\gamma_2\gamma_j\rho_{2j}\gamma_j,\cdots,\sum_j^C\gamma_C\gamma_j\rho_{Cj}\gamma_j\right]^{\mathrm{T}}) \\ &= \frac{1}{2}\left[(3-\sum_j^C\frac{\gamma_j^2\rho_{1j}}{\|\gamma\|_2^2})\gamma_1,(3-\sum_j^C\frac{\gamma_j^2\rho_{2j}}{\|\gamma\|_2^2})\gamma_2,\cdots,(3-\sum_j^C\frac{\gamma_j^2\rho_{Cj}}{\|\gamma\|_2^2})\gamma_C\right]^{\mathrm{T}}\end{aligned} \tag{14}$$

Note that $|3-\sum_j^C\frac{\gamma_j^2\rho_{ij}}{\|\gamma\|_2^2}| \geq 3-|\sum_j^C\frac{\gamma_j^2\rho_{ij}}{\|\gamma\|_2^2}| \geq 3-\sum_j^C\frac{\gamma_j^2}{\|\gamma\|_2^2} = 2$, where the last equality holds iff $\rho_{ij}=1$, $\forall i,j\in[C]$. However, this is not the case in practice. Hence we have

$$\left|\left[\frac{1}{2}(3I-\Sigma)\gamma\right]_i\right| = \left|\frac{1}{2}(3-\sum_j^C\frac{\gamma_j^2\rho_{ij}}{\|\gamma\|_2^2})\gamma_i\right| > |\gamma_i| \tag{15}$$

Therefore, we have $\|\hat{\gamma}\|_1 > \|\gamma\|_1$. Here completes the proof.

D CONNECTION BETWEEN CE BLOCK AND NASH EQUILIBRIUM

We first introduce the definition of Gaussian interference game in context of CNN and then build the connection between a CE block and Nash Equilibrium. For clarity of notation, we omit the subscript 'n' for a concrete sample.

Let the channels $1, 2, \cdots, C$ operate over $H \times W$ pixels. Assume that the C channels have dependencies $G = \{g_{cd}(i,j)\}_{c,d=1}^{C,C}$. Each pixel is characterized by a power gain $h_{cij} \geq 0$ and channel noise strength $\sigma_c > 0$. In context of normalization, we suppose $h_{cij} = \bar{x}_{cij} + \delta$ where $\bar{x}_{cij}$ is standardized pixel in Eqn.(1) and δ is sufficiently large to guarantee a non-negative power gain. Assume that c-th channel is allowed to transmit a total power of P_c and we have $\sum_{i,j=1}^{H,W} p_{cij} = P_c$. Besides, each channel can transmit a power vector $p_c = (p_{c11}, \cdots, p_{cHW})$. Since normalization layer is often followed by a ReLU activation, we restrict $p_{cij} \geq 0$. What we want to maximize the capacity transmitted over the c-th channel, $\forall c \in [C]$, then the maximization problem is given by:

$$\begin{aligned} &\max \; C_c(p_1, p_2, \cdots, p_C) = \sum_{i,j=1}^{h,W} \ln\left(1 + \frac{g_{cc} p_{cij}}{\sum_{d \neq c} g_{cd} p_{dij} + \sigma_c / h_{cij}}\right) \\ &s.t. \; \begin{cases} \sum_{i,j=1}^{H,W} p_{cij} = P_c, & \\ p_{cij} \geq 0, & \forall i \in [H], j \in [W] \end{cases} \end{aligned} \tag{16}$$

where C_c is the capacity available to the c-th channel given power distributions $p_1, p_2, \cdots, p_C$. In game theory, C channels and solution space of $\{p_{cij}\}_{c,i,j=1}^{C,H,W}$ together with pay-off vector $\mathbf{C} = (C_1, C_2, \cdots, C_C)$ form a Gaussian interference game $\mathbb{G}$. Different from basic settings in $\mathbb{G}$, here we do not restrict dependencies g_{cd} to $(0, 1)$. It is known that $\mathbb{G}$ has a unique Nash Equilibrium point whose definition is given as below,

Definition 1. *An C-tuple of strategies $(p_1, p_2, \cdots, p_C)$ for channels $1, 2, \cdots, C$ respectively is called a Nash equilibrium iff for all c and for all p (p a strategy for channel c)*

$$C_c(p_1, \cdots, p_{c-1}, p, p_{c+1}, \cdots, p_C) \leq C_c(p_1, p_2, \cdots, p_C) \tag{17}$$

i.e., given that all other channels $d \neq c$ use strategies p_d, channel c best response is p_c. Since $C_1, C_2, \cdots, C_C$ are concave in $p_1, p_2, \cdots, p_C$ respectively, KKT conditions imply the following theorem.

Theorem 1. *Given pay-off in Eqn.(16), $(p_1^*, \cdots, p_C^*)$ is a Nash equilibrium point if and only if there exist $v_0 = (v_0^1, \cdots, v_0^C)$ (Lagrange multiplier) such that for all $i \in [H]$ and $j \in [W]$,*

$$\frac{g_{cc}}{\sum_d g_{cd} p_{dij}^* + \sigma_c / h_{cij}} \begin{cases} = v_0^c \text{ for } p_{cij}^* > 0 \\ \leq v_0^c \text{ for } p_{cij}^* = 0 \end{cases} \tag{18}$$

Proof. The Lagrangian corresponding to minimization of $-C_c$ subject to the equality constraint and non-negative constraints on p_{cij} is given by

$$L_c = -\sum_{i,j=1}^{h,W} \ln\left(1 + \frac{g_{cc} p_{cij}}{\sum_{d \neq c} g_{cd} p_{dij} + \sigma_c / h_{cij}}\right) + v_0^c\left(\sum_{i,j=1}^{H,W} p_{cij} - P_c\right) + \sum_{i,j=1}^{H,W} v_1^{cij}(-p_{cij}). \tag{19}$$

Differentiating the Lagrangian with respect to p_{cij} and equating the derivative to zero, we obtain

$$\frac{g_c C}{\sum_d g_{cd} p_{cij} + \sigma_c / h_{cij}} + v_1^{cij} = v_0^c \tag{20}$$

Now, using the complementary slackness condition $v_1^{cij} p_{cij} = 0$ and $v_1^{cij} \geq 0$, we obtain condition (18). This completes the proof.

By Theorem 1, the unique Nash Equilibrium point can be explicitly written as follows when $p_{cij}^* > 0$,

$$p_{ij}^* = G^{-1}\left(\mathrm{Diag}(v_0)^{-1}\mathrm{diag}(G) - \mathrm{Diag}(h_{ij})^{-1}\sigma\right) \tag{21}$$

where $p^*_{ij}, h_{ij}, \sigma \in \mathbb{R}^C$ and $v_0 \in \mathbb{R}^C$ are Lagrangian multipliers corresponding to equality constraints. Note that a approximation can be made using Taylor expansion as follow: $-\frac{\sigma_c}{h_{cij}} = \sigma_c(2 + h_{cij} + \mathcal{O}((1 + h_{cij})^2))$. Thus, a linear proxy to Eqn.(21) can be written as

$$p^*_{ij} = G^{-1}\left(\mathrm{Diag}(\sigma)\bar{x}_{ij} + \mathrm{Diag}(v_0)^{-1}\mathrm{diag}(G) + (2+\delta)\sigma\right) \tag{22}$$

Let $G = [D_n]^{\frac{1}{2}}, \gamma = \sigma$ and $\beta = \mathrm{Diag}(v_0)^{-1}\mathrm{diag}(G) + (2+\delta)\sigma$, Eqn.(22) can surprisingly match CE unit in Eqn.(4), implying that the proposed CE block indeed performs a mechanism on channel equalization. In Gaussian interference game, σ is known and v_0 can be determined when budget P_c's are given. However, γ and β are learned by SGD in deep neural networks.

E Experiments

Training and validation loss. We plot training and validation loss during the training process for ResNet50, SE-ResNet50 and CE-ResNet50 in Fig.6. We can observe that CE-ResNet50 consistently have lower training and validation errors over the whole training period.

Grad-cam visualization. We claim that AII learns adaptive inverse of variance for each channel in a self-attention manner. Fed into more informative input, AII is expected to make the network respond to different inputs in a highly class-specific manner. In this way, it helps CE learn preciser feature representation. To verify this, we employ an off-the-shelf tool to visualize the class activation map (CAM) Selvaraju et al. (2017). We use ResNet50, BD-ResNet50, and CE-ResNet50 trained on ImageNet for comparison. As shown in Fig.5, the heat maps extracted from CAM for CE-ResNet50 have more coverage on the object region and less coverage on the background region. It shows that the AII branch helps CE learn preciser information from the images.

F Training and Inference

Moving average in inference. Unlike previous methods in manual architecture design that do not depend on batch estimated statistics, the proposed CE block requires computing the inverse square root of a batch covariance matrix Σ and a global variance scale s in each training step. To make the output depend only on the input, deterministically in inference, we use the moving average to calculate the population estimate of $\hat{\Sigma}^{-\frac{1}{2}}$ and $\hat{s}^{-\frac{1}{2}}$ by following the below updating rules:

$$\hat{\Sigma}^{-\frac{1}{2}} = (1-m)\hat{\Sigma}^{-\frac{1}{2}} + m\Sigma^{-\frac{1}{2}}, \;\; \hat{s}^{-\frac{1}{2}} = (1-m)\hat{s}^{-\frac{1}{2}} + m \cdot s^{-\frac{1}{2}} \tag{23}$$

where s and Σ are the variance scale and covariance calculated within each mini-batch during training, and m denotes the momentum of moving average. It is worth noting that since $\hat{\Sigma}^{-\frac{1}{2}}$ is fixed during inference, the BD branch does not introduce extra costs in memory or computation except for a simple linear transformation ($\hat{\Sigma}^{-\frac{1}{2}}\tilde{x}$).

Model and computational complexity. The main computation of our CE includes calculating the covariance and inverse square root of it in the BD branch and computing two FC layers in the AII branch. We see that there is a lot of space to reduce computational cost of CE. For BD branch, given an internal feature $x \in \mathbb{R}^{N\times C\times H\times W}$, the cost of calculating a covariance matrix is $2NHWC^2$, which is comparable to the cost of convolution operation. A pooling operation can be employed to downsample featuremap for too large H and W. In this way, the complexity can be reduced to $2NHWC^2/k^2 + CHW$ where k is kernel size of the window for pooling. Further, we can use group-wise whitening to improve efficiency, reducing the cost of computing $\Sigma^{-\frac{1}{2}}$ from TC^3 to TCg^2 (g is group size). For AII branch, we focus on the additional parameters introduced by two FC layers. In fact, the reduction ratio r can be appropriately chosen to balance model complexity and representational power. Besides, the majority of these parameters come from the final block of the network. For example, a single AII in the final block of ResNet-50 has $2 * 2048^2/r$ parameters. In practice, the CE blocks in the final stages of networks are removed to reduce additional parameters. We provide the measurement of computational burden and Flops in Table.1.

ResNet Training Setting. All networks are trained using 8 GPUs with a mini-batch of 32 per GPU. We train all the architectures from scratch for 100 epochs using stochastic gradient descent (SGD) with momentum 0.9 and weight decay 1e-4. The base learning rate is set to 0.1 and is multiplied

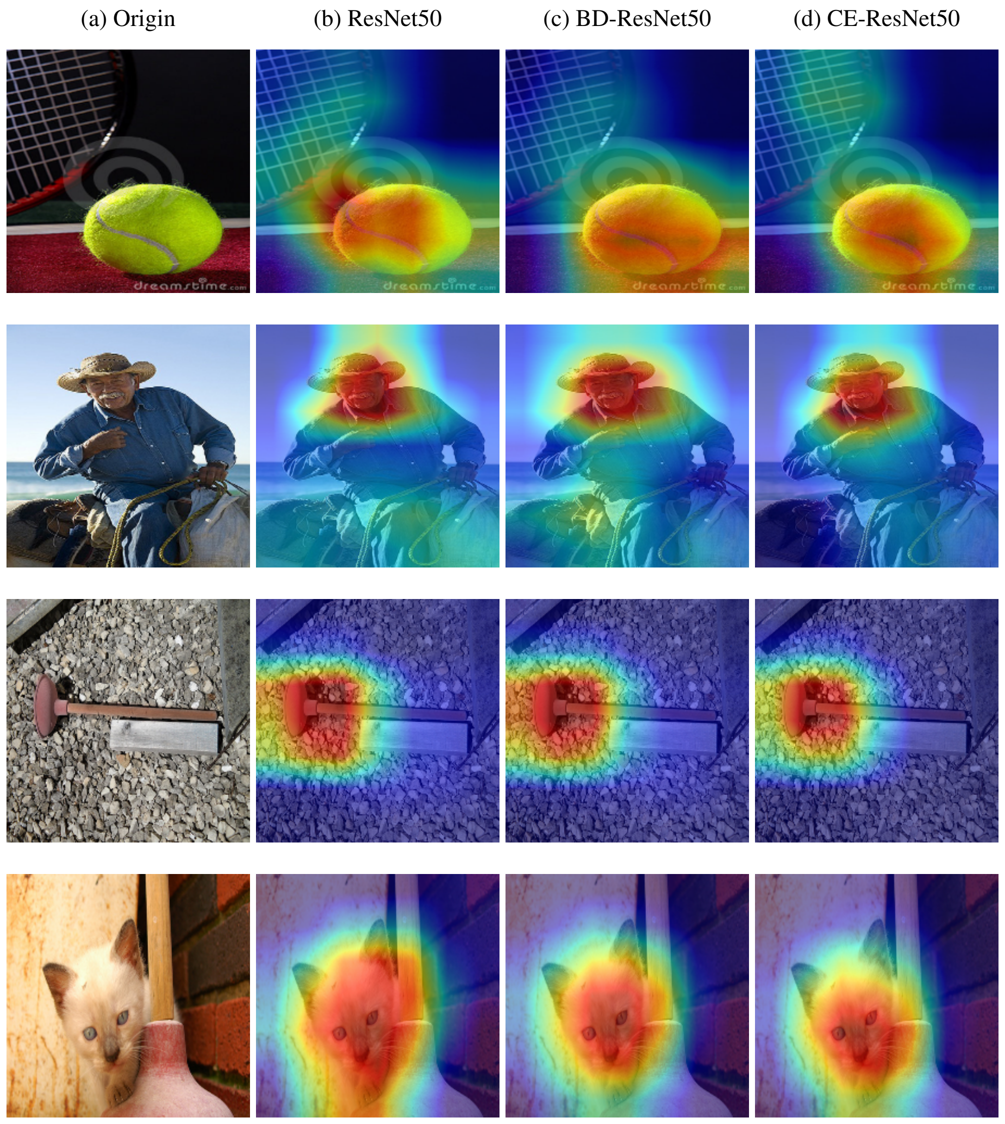

Figure 5: Grad-cam visualization results from the final convolutional layer for plain ResNet50, SE-ResNet50, and CE-ResNet50.

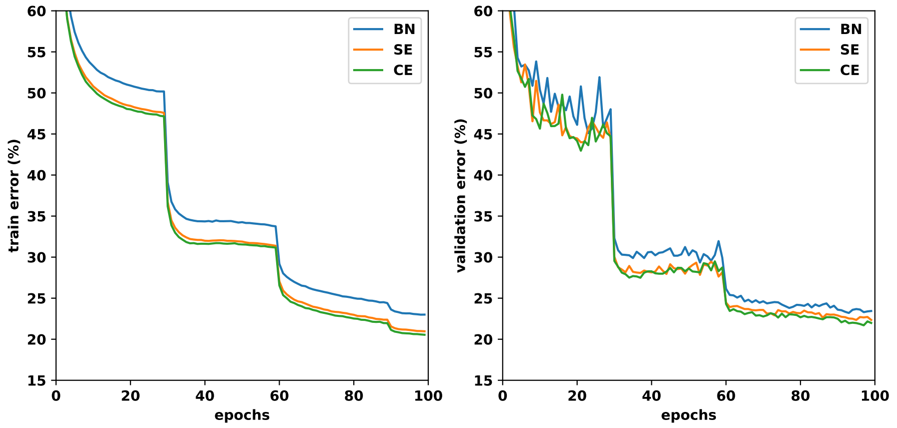

Figure 6: Training and validation error curves on ImageNet for different methods with ResNet50 as backbone.

by 0.1 after $30, 60$ and 90 epochs. Besides, the covariance matrix in BD branch is calculated within each GPU. Since the computation of covariance matrix involves heavy computation when the size of feature map is large, a 2×2 maximum pooling is adopted to down-sample the feature map after the first batch normalization layer. Like (Huang et al., 2019), we also use group-wise decorrelation with group size 16 across the network to improve the efficiency in the BD branch. By default, the reduction ratio r in AII branch is set to 4.

MobileNet V2 training Setting. All networks are trained using 8 GPUs with a mini-batch of 32 per GPU for 150 epochs with cosine learning rate. The base learning rate is set to 0.05 and the weight decay is 4e-5.

ShuffleNet V2 training Setting. All networks are trained using 8 GPUs with a mini-batch of 128 per GPU for 240 epochs with poly learning rate. The base learning rate is set to 0.5 and weight decay is 4e-5. We also adopt warmup and label smoothing tricks.

VGG networks on CIFAR10 training setting. For CIFAR10, we train VGG networks with a batch size of 256 on a single GPU for 160 epochs. The initial learning rate is 0.1 and is decreased by 10 times every 60 epochs.

Mask-RCNN training setting in COCO. We fine-tune the ImageNet pretrained model in COCO for 24 epoch with base learning rate 0.02 and multiply it by 0.1 after 16 and 22 epochs. All the models are trained using 8 GPUs with a mini-batch of 2 images. The basic backbone structure is adopted from the ResNet50/ResNet101 trained on ImageNet.