Understanding Normalization in Deep Learning

Speaker: Wenqi Shao

Email: Weqish@link.cuhk.edu.hk

Outline

- Introduction
- Various Normalizers: IN, BN, LN, SN, SSN
- An Unified Representation: Meta Norm (MN)

  Back-propagation & Geometric Interpretation
- Why Batch Normalization?

  Optimization & Generalization
- Normalization in Various Computer Vision Tasks

Introduction

- Normalization is a well-known technique in deep learning.

- The first normalization method----Batch Normalization (BN).
  BN achieves the same accuracy with 14 times fewer training steps

- Normalization improves both optimization and generalization of a DNN.
- Various normalizers in terms of tasks and network architecture

  — **Batch Normalization (BN),** Image classification [1]

  — **Instance Normalization (IN),** Image style transfer [2]

  — **Layer normalization (LN),** Recurrent Neural Network (RNN) [3]

  — **Group normalization (GN),** robust to batch size, image classification, object detection [4]

Normalization methods have been a foundation of various **state-of-the-art** computer vision tasks

Introduction

- Object of normalization method

  — a **4-D tensor** $\boldsymbol{h} \in \boldsymbol{R}^{N\times C\times H\times W}$

  N- minibatch size (the number of samples)

  C- number of channels

  H- height of a channel

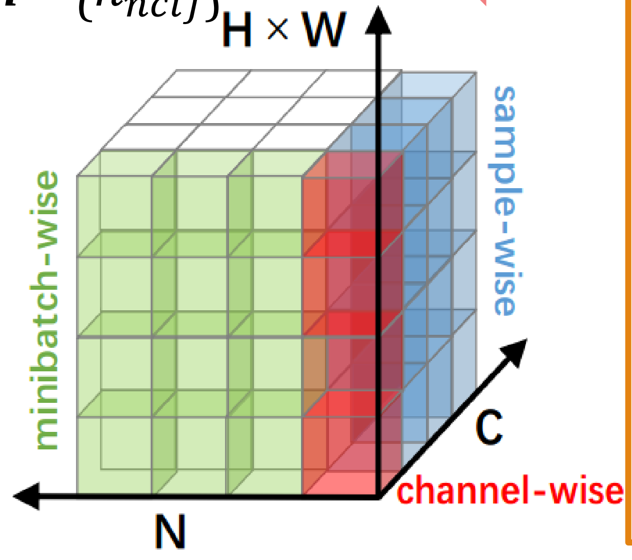
  W- width of a channel

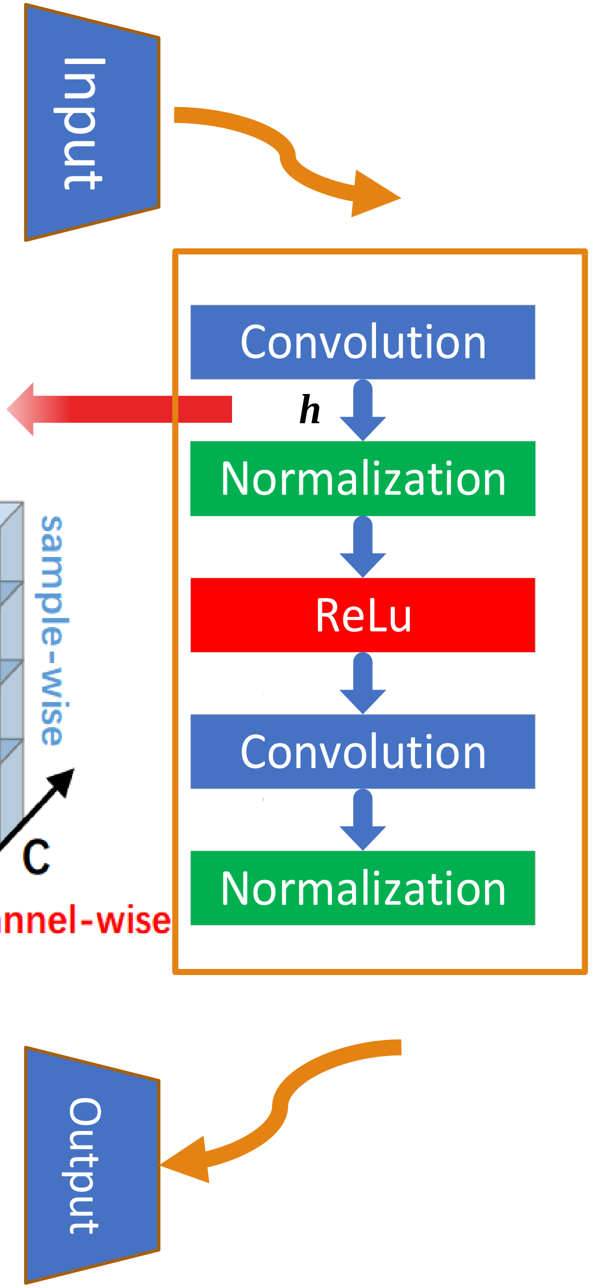
- A very common building block

  — **Conv+Norm+ReLU**

- They work by standardizing the activations within specific scope.

$$\boldsymbol{h} = \{h_{ncij}\}^{NCHW}$$





$$\boxed{\hat{h}_{ncij} = \gamma \frac{h_{ncij} - \mu}{\sqrt{\sigma^2 + \epsilon}} + \beta,}$$

- **Two statistics**: mean μ and variance σ^2
- **Two learnable parameters:**

  scale parameter γ and shift parameter β

Various Normalizers-IN, BN, LN and GN

Calculating mean μ and variance σ^2 in different scope produces different normalizers.

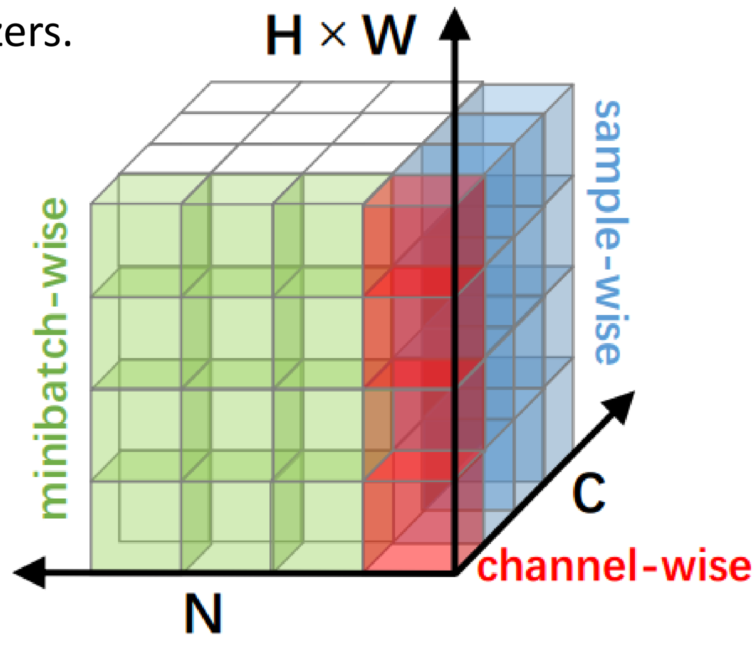
Given a feature map in DNN $\boldsymbol{h_{ncij}} \in \boldsymbol{R}^{\boldsymbol{N \times C \times H \times W}}$,

IN $$\boldsymbol{\mu_{IN}} = \frac{1}{HW}\sum_{i,j=1}^{H,W} h_{ncij}\,, \boldsymbol{\sigma_{IN}^2} = \frac{1}{HW}\sum_{i,j=1}^{H,W}\left(h_{ncij} - \mu_{IN}\right)^2$$

BN $$\boldsymbol{\mu_{BN}} = \frac{1}{NHW}\sum_{n,i,j=1}^{N,H,W} h_{ncij}\,, \boldsymbol{\sigma_{BN}^2} = \frac{1}{NHW}\sum_{n,i,j=1}^{N,H,W}\left(h_{ncij} - \mu_{BN}\right)^2$$

LN $$\boldsymbol{\mu_{LN}} = \frac{1}{CHW}\sum_{c,i,j=1}^{C,H,W} h_{ncij}\,, \boldsymbol{\sigma_{LN}^2} = \frac{1}{CHW}\sum_{c,i,j=1}^{C,H,W}\left(h_{ncij} - \mu_{LN}\right)^2$$

GN $$\boldsymbol{\mu_{GN}^g} = \frac{1}{C_g HW}\sum_{c,i,j=1}^{C_g,H,W} h_{ncij}\,, \boldsymbol{\sigma_{GN}^{g2}} = \frac{1}{C_g HW}\sum_{c,i,j=1}^{C_g,H,W}\left(h_{ncij} - \mu_{GN}^g\right)^2$$

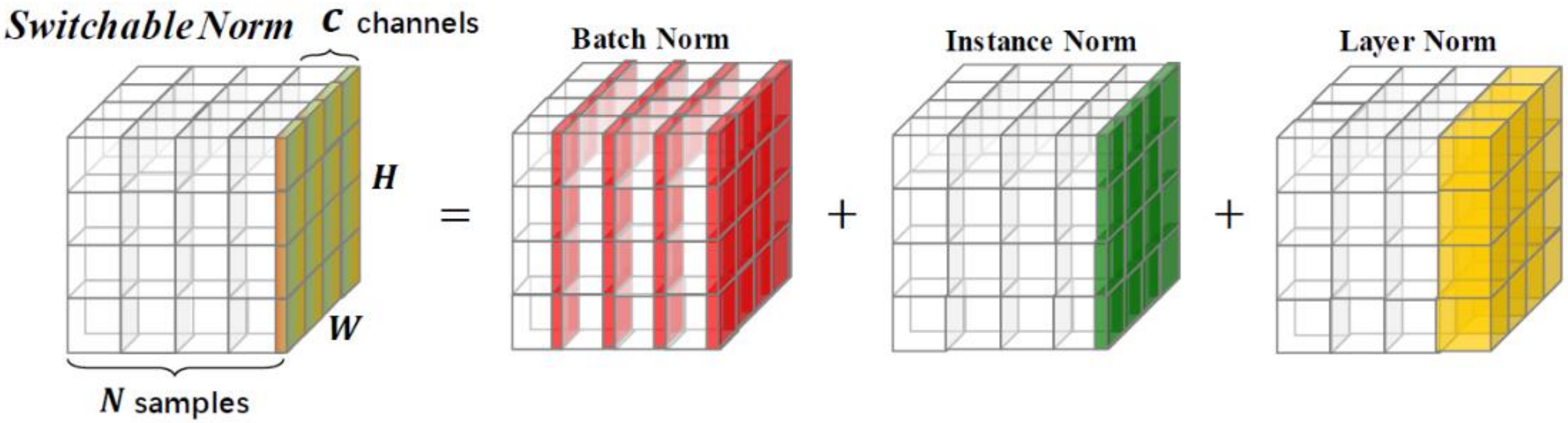


GN divides the channels into groups and computes within each group the mean and variance for normalization.

Various Normalizers-SN and SSN

The above-mentioned methods of normalization use the same normalizer in different normalization layer.

Swithchable Normalization (SN) is able to learn different normalizer for each normalization layer [5].



$$\mu_{SN} = p_1\mu_{IN} + p_2\mu_{BN} + p_3\mu_{LN}, \sigma_{SN}^2 = p_1\sigma_{SN}^2 + p_2\sigma_{SN}^2 + p_3\sigma_{SN}^2$$

Where $(p_1, p_2, p_3) = \mathrm{softmax}(z_1, z_2, z_3)$ and z_1, z_2, z_3 are learnable parameters

z_1, z_2, z_3 learned by SGD in different layers could be different

Various Normalizers-SN and SSN

However, SN suffers from overfitting and redundant computation.

— **overfitting**, z_1, z_2, z_3 are optimized without any constraint.

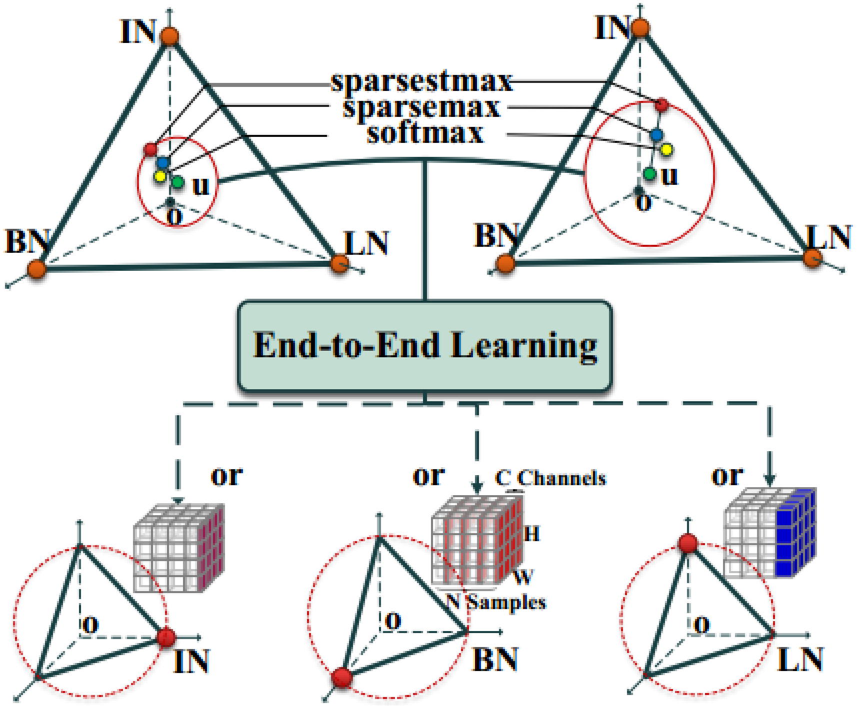
— **redundant computation**, compute all statistics in IN, BN and LN in the inference stage

Sparse Switchable Normalization (SSN) is able to learn only one normalizer for each normalization layer [6].

Statistics in SSN:

$\mu_{SN} = p_1\mu_{IN} + p_2\mu_{BN} + p_3\mu_{LN}, \sigma_{SN}^2 = p_1\sigma_{SN}^2 + p_2\sigma_{SN}^2 + p_3\sigma_{SN}^2$

Such that $p_1 + p_2 + p_3 = 1 \; and \; p_i \in \{0,1\}$



SSN is achieved by a novel transformation **'SparsestMax',** which is used to substituted **softmax in SN**

$$\mathrm{SparsestMax}(\mathbf{z}; r) := \underset{\mathbf{p} \in \triangle_r^{K-1}}{\mathrm{argmin}} \|\mathbf{p} - \mathbf{z}\|_2^2,$$

An Unified Representation: Meta Normalization [7]

Question. Is there an universal normalization that could include IN, BN, LN, etc. ?

To answer this question, let's consider the relation between μ_{IN} and μ_{BN}, μ_{LN}

$$\begin{bmatrix} \mu_{11} & \cdots & \mu_{1C} \\ \vdots & \ddots & \vdots \\ \mu_{N1} & \cdots & \mu_{NC} \end{bmatrix}$$

$$\mu_{IN} \in R^{N\times C}$$

Taking sum over each column

$$\mu_{BN} \in R^{C} \xrightarrow{\text{duplicate N rows}} \mu_{BN} = \frac{1}{N}\begin{bmatrix} 1 & \cdots & 1 \\ \vdots & \ddots & \vdots \\ 1 & \cdots & 1 \end{bmatrix}\mu_{IN}$$

Taking sum over each row

$$\mu_{LN} \in R^{N} \xrightarrow{\text{duplicate C columns}} \mu_{LN} = \frac{1}{C}\mu_{IN}\begin{bmatrix} 1 & \cdots & 1 \\ \vdots & \ddots & \vdots \\ 1 & \cdots & 1 \end{bmatrix}$$

An Unified Representation: Meta Normalization

MN. We can design an universal normalization by constructing binary matrix U and V as follows:

$$\mu_{MN} = \left(\frac{1}{Z_U} U\right) \mu_{IN} \left(\frac{1}{Z_V} V\right)$$

$$\sigma_{MN} = \left(\frac{1}{Z_U} U\right) \sigma_{IN} \left(\frac{1}{Z_V} V\right)$$

Z_U and Z_V are normalizing factor. $U \in R^{N \times N}$ and $V \in R^{C \times C}$ are two binary matrix whose elements are either 0 or 1

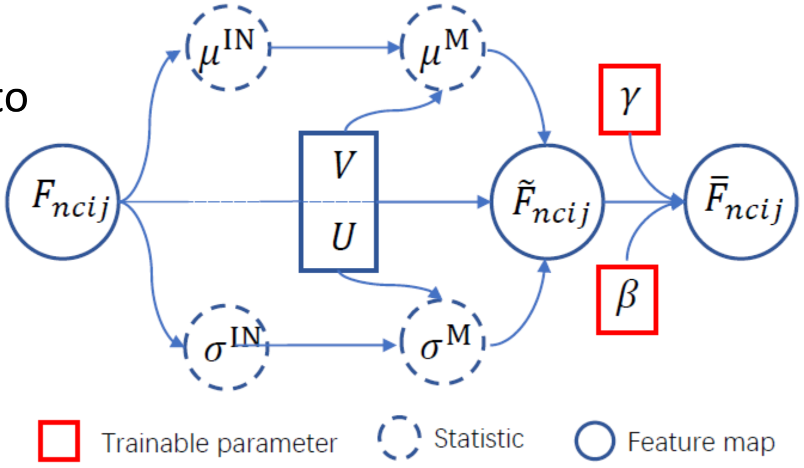
Representation Capacity. In MN, *V* aggregates the statistics from the channels, while *U* aggregates those in a batch of samples. Therefore, different *V* and *U* represent different normalization approaches.

- Let $U = I$ and $V = I$, then MN represents IN.
- Let $U = \frac{1}{N}\mathbf{1}$ and $V = I$, then MN turns into BN.
- Let $U = I$ and $V = \frac{1}{C}\mathbf{1}$, then MN represents LN.
- Let $U = I$ and $V = \frac{2}{C}\begin{bmatrix}\mathbf{1} & \mathbf{0} \\ \mathbf{0} & \mathbf{1}\end{bmatrix}$, then MN represents GN with a group number of 2.

Back-propagation of MN

MN. Let $\tilde{F}_{ncij}$ be the neuron after normalization, and then it is transformed to $\bar{F}_{ncij}$.

$$\tilde{F}_{ncij} = \frac{F_{ncij} - \mu^{\mathrm{MN}}_{nc}}{\sigma^{\mathrm{MN}}_{nc}}, \quad \bar{F}_{ncij} = \gamma_c \tilde{F}_{ncij} + \beta_c,$$



Back-propagation. What we most care about is to back-propagate the gradient of output $\frac{\partial L}{\partial \bar{F}_{ncij}}$ to the gradient of input $\frac{\partial L}{\partial F_{ncij}}$.

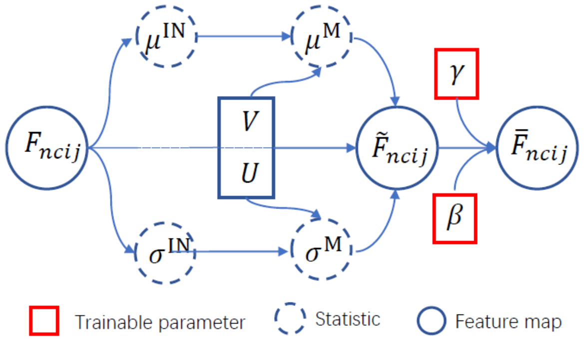
$$\begin{aligned}\frac{\partial \mathcal{L}}{\partial F_{ncij}} &= \frac{\partial \mathcal{L}}{\partial \tilde{F}_{ncij}} \frac{\partial \tilde{F}_{ncij}}{\partial F_{ncij}} + \frac{\partial \mathcal{L}}{\partial \mu^{\mathrm{MN}}} \bullet \frac{\partial \mu^{\mathrm{MN}}}{\partial F_{ncij}} \\ &+ \frac{\partial \mathcal{L}}{\partial \sigma^{\mathrm{MN}}} \bullet \frac{\partial \sigma^{\mathrm{MN}}}{\partial F_{ncij}} \\ &\triangleq \mathrm{term1} + \mathrm{term2} + \mathrm{term3}.\end{aligned}$$

$$\mathrm{term1} = \frac{1}{\sigma^{\mathrm{MN}}_{nc}} \frac{\partial \mathcal{L}}{\partial \tilde{F}_{ncij}}.$$

$$\begin{aligned}\mathrm{term2} &= \left(\frac{\partial \mathcal{L}}{\partial \mu^{\mathrm{MN}}} \bullet \frac{\partial \mu^{\mathrm{MN}}}{\partial \mu^{\mathrm{IN}}_{nc}}\right) \frac{\partial \mu^{\mathrm{IN}}_{nc}}{\partial F_{ncij}} \\ &= \left(\frac{\partial \mathcal{L}}{\partial \mu^{\mathrm{MN}}} \bullet (u_n v_c)\right) \frac{1}{HW} \\ &= \frac{1}{HW} \mathrm{tr}\left[\left(\frac{\partial \mathcal{L}}{\partial \mu^{\mathrm{MN}}}\right)^{\mathsf{T}} u_n v_c\right] \\ &= \frac{1}{HW} v_c \left(\frac{\partial \mathcal{L}}{\partial \mu^{\mathrm{MN}}}\right)^{\mathsf{T}} u_n,\end{aligned}$$

$$\begin{aligned}\mathrm{term3} &= \left(\frac{\partial \mathcal{L}}{\partial \sigma^{\mathrm{MN}}} \bullet \frac{\partial \sigma^{\mathrm{MN}}}{\partial \sigma^{\mathrm{IN}}_{nc}}\right) \frac{\partial \sigma^{\mathrm{IN}}_{nc}}{\partial F_{ncij}} \\ &= \left(\frac{\partial \mathcal{L}}{\partial \sigma^{\mathrm{MN}}} \bullet (u_n v_c)\right) \frac{\tilde{F}_{ncij}}{HW} \\ &= \frac{\tilde{F}_{ncij}}{HW} \mathrm{tr}\left[\left(\frac{\partial \mathcal{L}}{\partial \sigma^{\mathrm{MN}}}\right)^{\mathsf{T}} u_n v_c\right] \\ &= \frac{\tilde{F}_{ncij}}{HW} v_c \left(\frac{\partial \mathcal{L}}{\partial \sigma^{\mathrm{MN}}}\right)^{\mathsf{T}} u_n.\end{aligned}$$

Back-propagation of MN



Back-propagation. $\frac{\partial L}{\partial \tilde{F}_{ncij}} \triangleq \tilde{d}_{ncij}$ $\Rightarrow$ $\frac{\partial L}{\partial F_{ncij}} \triangleq d_{ncij}$

$$\frac{\partial \mathcal{L}}{\partial F_{ncij}} = \frac{1}{\sigma_{nc}^{\mathrm{MN}}} \frac{\partial \mathcal{L}}{\partial \tilde{F}_{ncij}} + \frac{1}{HW} v_c \left(\frac{\partial \mathcal{L}}{\partial \mu^{\mathrm{MN}}} \right)^{\mathsf{T}} u_n + \frac{\tilde{F}_{ncij}}{HW} v_c \left(\frac{\partial \mathcal{L}}{\partial \sigma^{\mathrm{MN}}} \right)^{\mathsf{T}} u_n, \quad (*)$$

Geometric View of BN. Let $U = \frac{1}{N}\mathbf{1}$ and $V = I$.

$$d_c = \frac{1}{\sigma_c^{\mathrm{MN}}} \left(I - \frac{\mathbf{1}\mathbf{1}^{\mathsf{T}} + \tilde{F}_c \tilde{F}_c^{\mathsf{T}}}{NHW} \right) \tilde{d}_c,$$

Geometric View of LN. Let $U = I$ and $V = \frac{1}{C}\mathbf{1}$.

$$d_n = \frac{1}{\sigma_n^{\mathrm{MN}}} \left(I - \frac{\mathbf{1}\mathbf{1}^{\mathsf{T}} + \tilde{F}_n \tilde{F}_n^{\mathsf{T}}}{CHW} \right) \tilde{d}_n,$$

Geometric View of N with group number G.

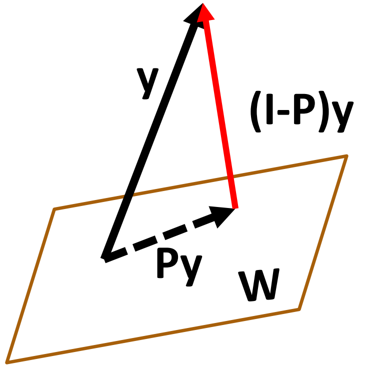
Let $U = I$ and $V = \frac{g}{C}\underbrace{\begin{bmatrix} \mathbf{1} & & \\ & \mathbf{1} & \\ & & \mathbf{1} \end{bmatrix}}_{\text{G diagonal sub-matrixes}}.$

$$d_{n(\frac{c}{g})} = \frac{1}{\sigma^{MN}_{n(\frac{c}{g})}} \left(I - \frac{11^{\mathsf{T}} + \tilde{F}_{n(\frac{c}{g})ij} \tilde{F}_{n(\frac{c}{g})ij}^{\mathsf{T}}}{C_g HW} \right) \tilde{d}_{n(\frac{c}{g})}$$

Geometric Interpretation

Projection Matrix. Given a matrix A, we have projection matrix $\boldsymbol{P} = \boldsymbol{A}(\boldsymbol{A}^{\boldsymbol{T}}\boldsymbol{A})^{-1}\boldsymbol{A}^{\boldsymbol{T}}$.
The columns of A, we're given, form a basis for some subspace W, matrix $(\boldsymbol{I} - \boldsymbol{P})$ is the projection matrix for the **orthogonal complement of W.**

Given a vector y, $\boldsymbol{Py}$ lies in subspace $\boldsymbol{W}$ and $(\mathrm{I} - \boldsymbol{P})\boldsymbol{y}$ is in the **orthogonal complement of W.**



Take **BN** as an example.

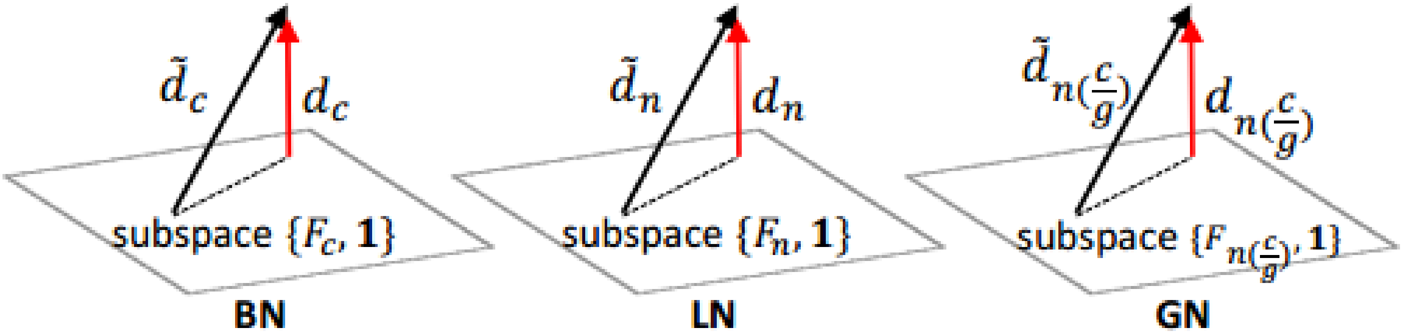
$$d_c = \frac{1}{\sigma_c^{\mathrm{MN}}}\left(I - \frac{\mathbf{1}\mathbf{1}^{\top} + \tilde{F}_c\tilde{F}_c^{\top}}{NHW}\right)\tilde{d}_c,$$

Let $A = [\mathbf{1}, \tilde{F}_c]$, then

$$A^T A = \begin{bmatrix} \mathbf{1}^{\boldsymbol{T}}\mathbf{1} & 0 \\ 0 & \tilde{F}_c^T\tilde{F}_c \end{bmatrix} = NHW \begin{bmatrix} \mathbf{1} & \\ & \mathbf{1} \end{bmatrix}.$$

Therefore, the projection matrix corresponding to A is exactly

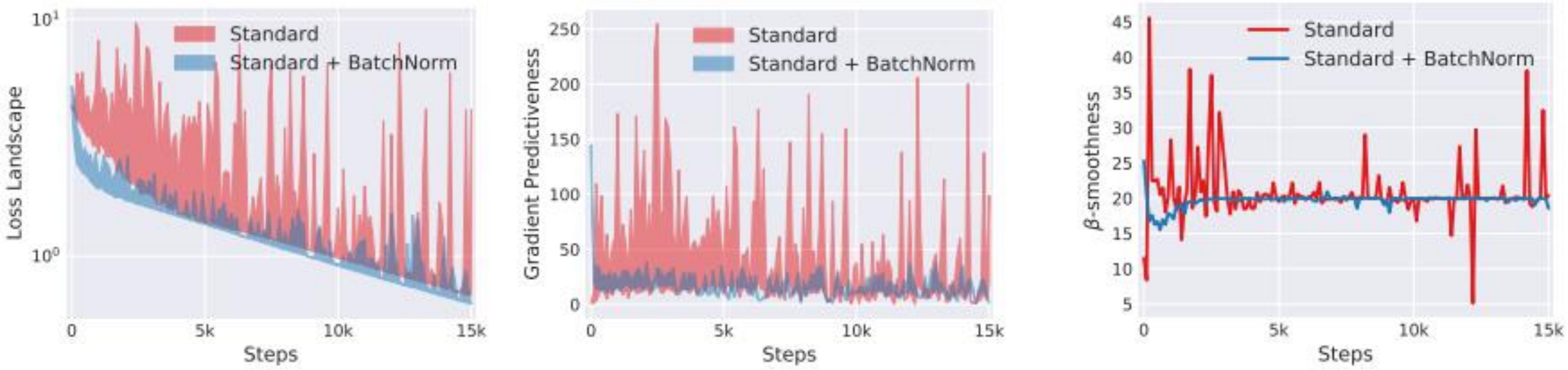
$$\frac{\mathbf{1}\mathbf{1}^{\top} + \tilde{F}_c\tilde{F}_c^{\top}}{NHW}$$



Why Batch Normalization?

BN has been an indispensable component in various networks architectures. The effectiveness of BN has been uncovered form two aspects: **optimization and generalization.**

A more fundamental impact of BatchNorm on the training process: **it makes the optimization landscape significantly smoother** [8]**.**



(a) loss landscape (b) gradient predictiveness (c) "effective" β-smoothness

the variation (shaded region) in loss

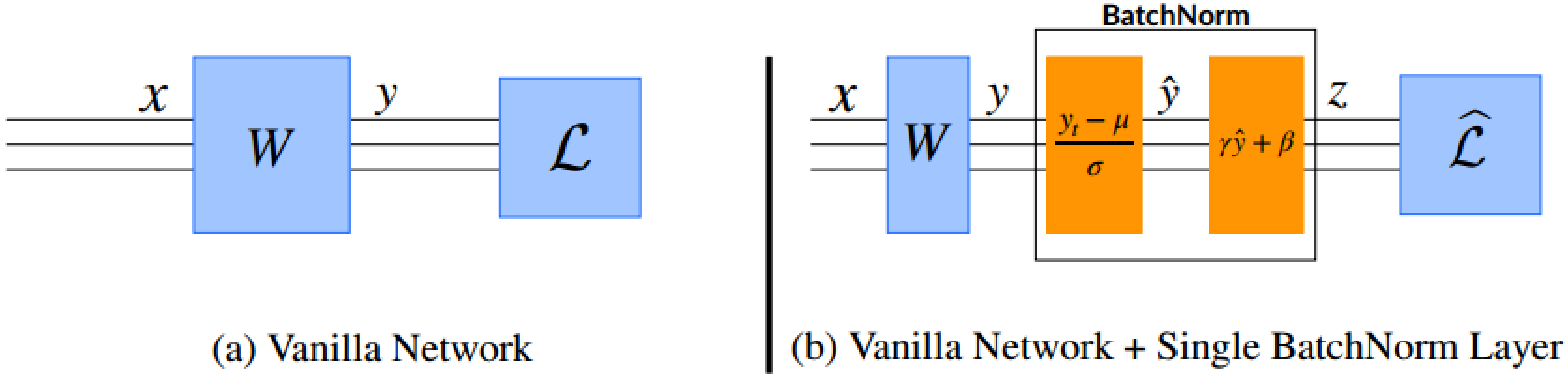
ℓ_2 changes in the gradient as we move in the gradient direction

maximum difference (ℓ_2 nrom) in gradient over distance moved in that direction.

Lipschitzness of the Loss

BN causes the landscape to be more well-behaved, inducing favorable properties in Lipschitz-continuity。

Let’s first consider the optimization landscape **wrt. activation.**



(a) Vanilla Network

(b) Vanilla Network + Single BatchNorm Layer

Theorem 4.1 (The effect of BatchNorm on the Lipschitzness of the loss). *For a BatchNorm network with loss $\widehat{\mathcal{L}}$ and an identical non-BN network with (identical) loss $\mathcal{L}$,*

$$\underbrace{\left|\left|\nabla_{\boldsymbol{y_j}}\widehat{\mathcal{L}}\right|\right|^2}_{} \leq \underbrace{\frac{\gamma^2}{\sigma_j^2}}_{}\left(\left|\left|\nabla_{\boldsymbol{y_j}}\mathcal{L}\right|\right|^2 - \underbrace{\frac{1}{m}\left\langle \mathbf{1}, \nabla_{\boldsymbol{y_j}}\mathcal{L}\right\rangle^2}_{} - \underbrace{\frac{1}{\sqrt{m}}\left\langle \nabla_{\boldsymbol{y_j}}\mathcal{L}, \hat{\boldsymbol{y}}_j\right\rangle^2}_{}\right).$$

gradient magnitude, captures **the Lipschitzness** of the loss

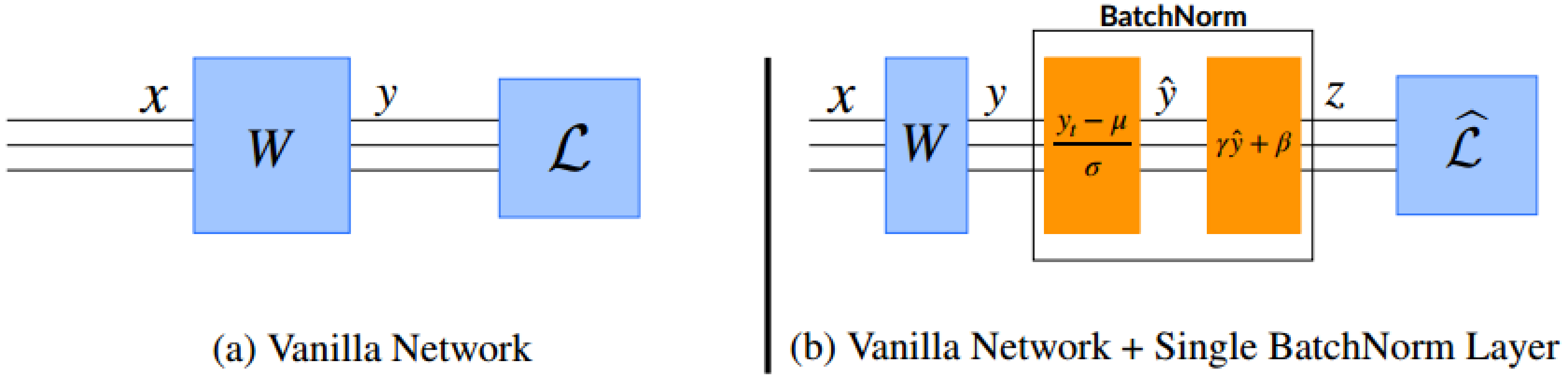
empically **less than 1**

grows quadratically in the dimension

bounded away from zero

Lipschitzness of the Loss

Let's now turn to consider the optimization landscape **wrt. weight.**



Theorem 4.4 (Minimax bound on weight-space Lipschitzness). *For a BatchNorm network with loss $\widehat{\mathcal{L}}$ and an identical non-BN network (with identical loss $\mathcal{L}$), if*

$$g_j = \max_{||X|| \le \lambda} ||\nabla_W \mathcal{L}||^2, \qquad \hat{g}_j = \max_{||X|| \le \lambda} \left|\left|\nabla_W \widehat{\mathcal{L}}\right|\right|^2 \implies \hat{g}_j \le \frac{\gamma^2}{\sigma_j^2}\left(g_j^2 - m\mu_{g_j}^2 - \lambda^2 \left\langle \nabla_{\boldsymbol{y}_j}\mathcal{L}, \hat{\boldsymbol{y}}_j \right\rangle^2\right).$$

Regularization in BN

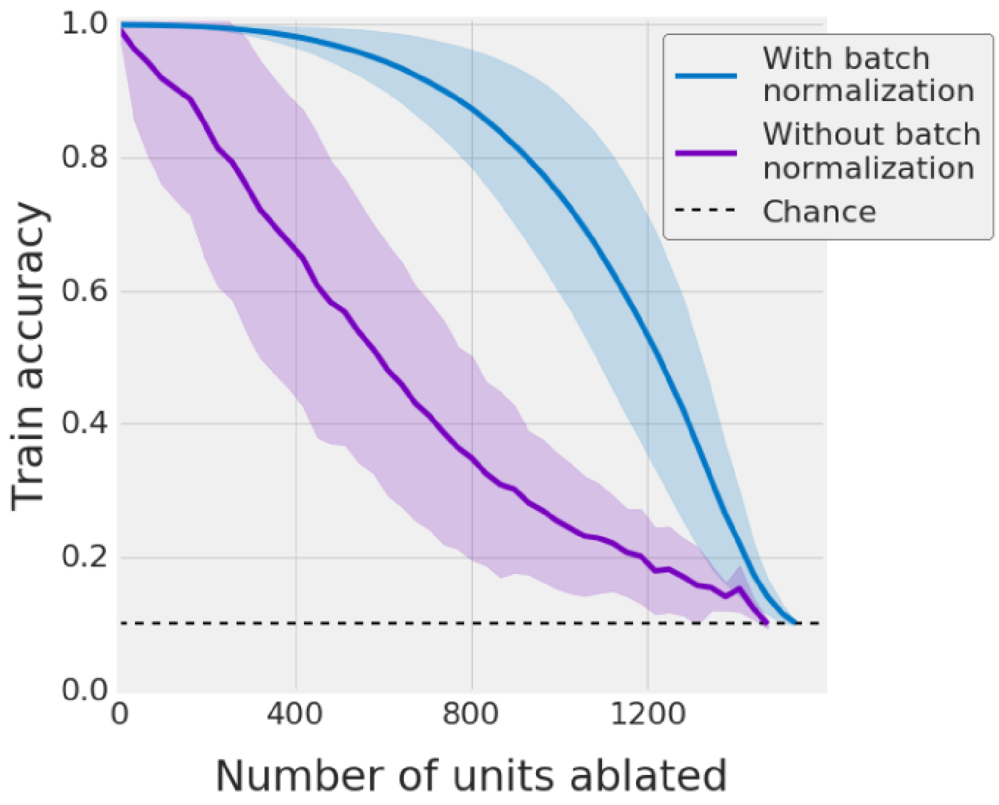
Batch normalization implicitly **discourages single channel reliance**, suggesting an alternative **regularization mechanism** by which batch normalization may **encourage good generalization performance.**

BN makes channel equal such that they play homogeneous role in representing a prediction function.

How to empirically verify this conclusion? [9]

measure their robustness to cumulative ablation of channels

Networks trained with batch normalization are more robust to these ablations than those trained without batch normalization

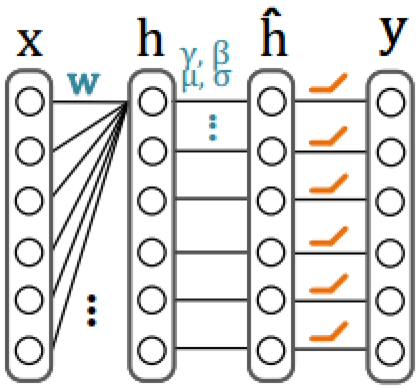


Regularization in BN

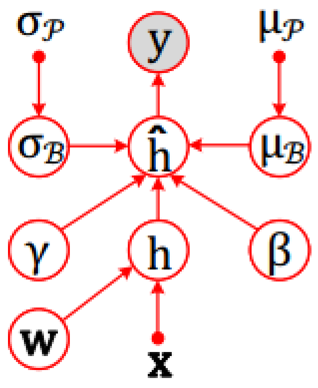
We explore **explicit regularization expression in BN** by analyzing a building block in a deep network.

BN also induces Gaussian priors for
batch mean $\boldsymbol{\mu_B}$ and batch standard deviation $\boldsymbol{\sigma_B}$. [10]

$$\mu_{\mathcal{B}} \sim \mathcal{N}(\mu_{\mathcal{P}}, \frac{\sigma_{\mathcal{P}}^2}{M}) \text{ and } \sigma_{\mathcal{B}} \sim \mathcal{N}(\sigma_{\mathcal{P}}, \frac{\rho+2}{4M}),$$



(a) a building block



(b) a graphical model

These priors tell us that $\boldsymbol{\mu_B}$ **and** $\boldsymbol{\sigma_B}$ would produce Gaussian noise.

Taking expectation over such noise may give us **explicit regularization expression in BN.** [11]

Theorem 1 (Regularization of $\mu_{\mathcal{B}}, \sigma_{\mathcal{B}}$). *Let ζ be the strength (coefficient) of the regularization and the activation function be ReLU. Then*

$$\frac{1}{P}\sum_{j=1}^{P} \mathbb{E}_{\mu_{\mathcal{B}},\sigma_{\mathcal{B}}} \ell(\hat{h}^j) \simeq \frac{1}{P}\sum_{j=1}^{P} \ell(\bar{h}^j) + \zeta\gamma^2, \quad (3)$$

$$\text{and } \zeta = \underbrace{\frac{\rho+2}{8M} F_\gamma}_{\text{from } \sigma_{\mathcal{B}}} + \underbrace{\frac{1}{2M}\frac{1}{P}\sum_{j=1}^{P} \sigma(\bar{h}^j)}_{\text{from } \mu_{\mathcal{B}}}, \quad (4)$$

- regularization strength ζ **is *inversely proportional* to** the batch size *M*.
- $\boldsymbol{\mu_B}$ **and** $\boldsymbol{\sigma_B}$ produce two different regularization strengths.
- $\boldsymbol{\mu_B}$ penalizes the expectation of activation, implying that *the* ***neuron with larger output may exposure to larger regularization***.

expectation of activation $\uparrow$ ζ $\uparrow$ γ $\downarrow$ expectation of activation $\downarrow$

Normalization in Various Computer Vision Tasks

Image Classification

	IN	LN	BN	GN	SN	SSN
top-1	71.6	74.7	76.4	75.9	76.9	**77.2**
Δ *v.s.* BN	-4.8	-1.7	-	-0.5	0.5	**0.8**

Table 1 Comparisons of top-1 accuracy(%) of ResNet-50 in ImageNet validation set. All models are trained with a batch size of 32 images/GPU. The second row shows the accuracy differences between BN and other normalization methods.

Object Detection

backbone	head	AP	$AP_{.5}$	$AP_{.75}$	AP_l	AP_m	AP_s
BN†	-	37.9	59.3	41.1	49.9	41.1	21.5
GN	GN	38.3	60.4	41.4	49.3	41.3	22.9
SN	SN	**39.1**	**61.5**	42.4	**50.0**	42.2	**23.4**
SSN	SSN	**39.1**	61.2	**42.7**	**50.0**	**42.6**	22.9

Table 4 Faster R-CNN+FPN using ResNet50 and FPN with 2x LR schedule. BN† represents BN is frozen. The best results are bold. SSN is finetuned from ResNet-50 SSN ImageNet pretrained model.

Semantic Segmentation

	ADE20K		Cityscapes	
	$mIoU_{ss}$	$mIoU_{ms}$	$mIoU_{ss}$	$mIoU_{ms}$
SyncBN	36.4	37.7	69.7	73.0
GN	35.7	36.3	68.4	73.1
SN (8,2)	38.7	39.2	71.6	75.4
SN (8,4)	38.6	39.0	72.1	75.8
SSN (8,2)	36.5	37.1	71.1	75.0
SSN (8,4)	38.5	39.3	71.7	75.7
SyncSSN (8,2)	39.3	39.8	75.1	76.2
SyncSSN (8,4)	**40.1**	**40.3**	**75.7**	**76.3**

Table 6 Results in ADE20K validation set and Cityscapes test set by using ResNet50 with dilated convolutions. 'ss' and 'ms' indicate single-scale and multi-scale inference. SyncBN represents multi-GPU synchronization of BN. SyncSSN indicates the BN in SSN is synchronized across mutli-GPU.

Normalization in Various Computer Vision Tasks

Image Classification

	IN	LN	BN	GN	SN	SSN
top-1	71.6	74.7	76.4	75.9	76.9	**77.2**
Δ *v.s.* BN	-4.8	-1.7	-	-0.5	0.5	**0.8**

Table 1 Comparisons of top-1 accuracy(%) of ResNet-50 in ImageNet validation set. All models are trained with a batch size of 32 images/GPU. The second row shows the accuracy differences between BN and other normalization methods.

Object Detection

backbone	head	AP	$AP_{.5}$	$AP_{.75}$	AP_l	AP_m	AP_s
BN†	-	37.9	59.3	41.1	49.9	41.1	21.5
GN	GN	38.3	60.4	41.4	49.3	41.3	22.9
SN	SN	**39.1**	**61.5**	42.4	**50.0**	42.2	**23.4**
SSN	SSN	**39.1**	61.2	**42.7**	**50.0**	**42.6**	22.9

Table 4 Faster R-CNN+FPN using ResNet50 and FPN with 2x LR schedule. BN† represents BN is frozen. The best results are bold. SSN is finetuned from ResNet-50 SSN ImageNet pretrained model.

Semantic Segmentation

	ADE20K		Cityscapes	
	$mIoU_{ss}$	$mIoU_{ms}$	$mIoU_{ss}$	$mIoU_{ms}$
SyncBN	36.4	37.7	69.7	73.0
GN	35.7	36.3	68.4	73.1
SN (8,2)	38.7	39.2	71.6	75.4
SN (8,4)	38.6	39.0	72.1	75.8
SSN (8,2)	36.5	37.1	71.1	75.0
SSN (8,4)	38.5	39.3	71.7	75.7
SyncSSN (8,2)	39.3	39.8	75.1	76.2
SyncSSN (8,4)	**40.1**	**40.3**	**75.7**	**76.3**

Table 6 Results in ADE20K validation set and Cityscapes test set by using ResNet50 with dilated convolutions. 'ss' and 'ms' indicate single-scale and multi-scale inference. SyncBN represents multi-GPU synchronization of BN. SyncSSN indicates the BN in SSN is synchronized across mutli-GPU.

References

1. Ioffe S, Szegedy C (2015) Batch normalization: Accelerating deep network training by reducing internal covariate shift.
2. Ulyanov D, Vedaldi A, Lempitsky V (2017) Instance normalization: the missing ingredient for fast stylization.
3. Ba JL, Kiros JR, Hinton GE (2016) Layer normalization.
4. Wu Y, He K (2018) Group normalization.
5. Luo P, Ren J, Peng Z (2018) Differentiable learning-tonormalize via switchable normalization.
6. Shao W, Meng T, Li J (2019) Learning sparse switchable normalization via SparsestMax.
7. Luo P (2019) Differentiable Learning to Learn to Normalize. (To be appeared)
8. Santurkar S, Tsipras D, Ilyas A, Madry A (2018) How does batch normalization help optimization?
9. Ari S Morcos, David GT Barrett, Neil C Rabinowitz, and Matthew Botvinick (2019). On the importance of single directions for generalization.
10. Teye M, Azizpour H, Smith K (2018) Bayesian uncertainty estimation for batch normalized deep networks.
11. Luo P, Wang X, Shao W, Peng Z (2018) Understanding regularization in batch normalization.

Thanks!