CVPR 2021 Tutorial

Normalization Techniques in Deep Learning: Methods, Analyses and Applications

Lei Huang

Beihang University, Beijing, China

北京航空航天大学人工智能研究院
Institute of Artificial Intelligence, Beihang University

2021-06-19

软件开发环境国家重点实验室
State Key Laboratory of Software Development Environment

Outline

01. Motivations of Normalization Techniques

02. Introduction of Normalization Methods

03. Analyses of Normalization

04. Applications of Normalization

Outline

Introduction of Normalization Methods

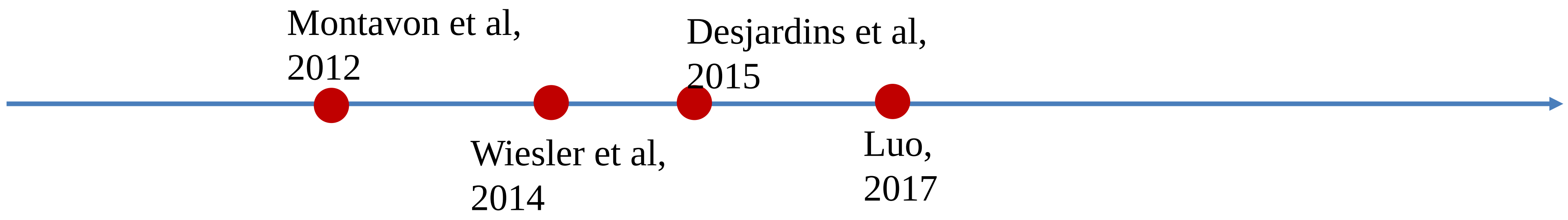
- Normalize Activation
  - By population statistics
  - As functions
- Normalize Weight
- Normalize Gradient

Normalizing activations

- Machine learning/Optimization community:

$$\Sigma_x = \mathbb{E}_{p(x)}\left(xx^{\boldsymbol{T}}\right)$$

Population statistics of a dataset



Normalization by Population Statistics

- Centering the activation
  - Montavon et al, 2014; Wiesler et al, 2014

$$\hat{\mathbf{x}} = \mathbf{x} - \hat{\boldsymbol{\mu}}$$

$\hat{\boldsymbol{\mu}}$ is the mean of activation over the training dataset;
Parameter to be estimated

- Standardizing the activation: centering + scaling
  - Wiesler et al, 2014

$$\hat{\mathbf{x}} = \frac{\mathbf{x} - \hat{\boldsymbol{\mu}}}{\hat{\sigma}}$$

$\hat{\sigma}$ is the standard deviation of activation over the training dataset;
Parameter to be estimated

- Whitening the activations
  - Desjardins et al 2015; Luo, 2017

$$\hat{x}_I = \hat{\Sigma}^{-\frac{1}{2}}(\mathbf{x} - \hat{\boldsymbol{\mu}})$$

$\hat{\Sigma}^{-\frac{1}{2}}$ is the whitening matrix of activation over the training dataset;
Parameter to be estimated

Normalization by Population Statistics

- Advantages
  - Well exploit the beneficial property of normalization in optimization
- Drawbacks
  - Training instability
    - The estimation is not accurate (sampled data)
    - Internal covariant shift (the distribution of activation varying with training progressing)
  - Can not be used to large networks
    - An inaccurate estimation of population statistics will be amplified as the layers increase

Outline

- Introduction of Normalization Methods
  - Normalize Activation
    - By population statistics
    - As functions
  - Normalize Weight
  - Normalize Gradient

Normalizing activations

- Machine learning/Optimization community:

  Population statistics of a dataset

$$\Sigma_x = \mathbb{E}_{p(x)}(xx^T)$$

Montavon et al, 2012

Wiesler et al, 2014

Desjardins et al, 2015

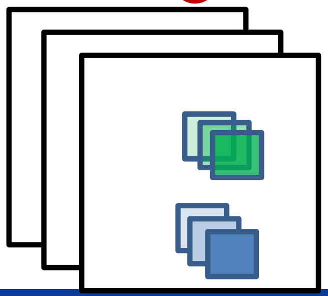
Luo, 2017

- Computer vision community:

  Local statistics in an sample

Jarrett et al, 2009

Krizhevsky et al, 2012



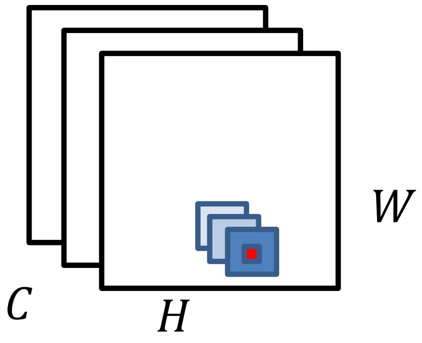
Ren et al, 2017

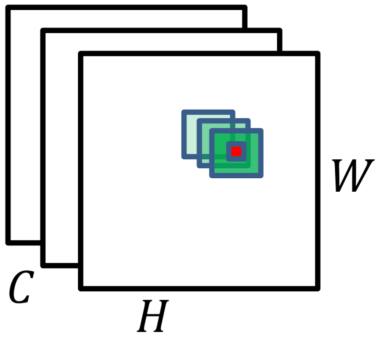
Ortiz et al, 2020

Local Normalization in a Sample

- Local normalization
  - Local contrast normalization [Jarrett et al, ICCV 2009]
  - Local response normalization [Krizhevsky et al, NeurIPS 2012]
  - Divisive normalization [Ren et al, ICLR 2017]
  - Local context normalization [Ortiz et al, CVPR 2020]

Given an example $X \in \mathbb{R}^{C \times H \times M}$





Local Normalization in a Sample

- Advantage
  - Training is somewhat stable due to back-propagating through normalization
  - The visual contrast invariant property may benefit generalization

- Limits
  - Specific to visual data (feature maps)
  - May change the representation ability and reduce the discriminative information
  - It is not clear whether benefits optimization

Normalizing activations

- Machine learning/Optimization community:

  Population statistics of a dataset

$$\Sigma_x = \mathbb{E}_{p(x)}\,(xx^T)$$

Montavon et al, 2012

Wiesler et al, 2014

Desjardins et al, 2015

Luo, 2017

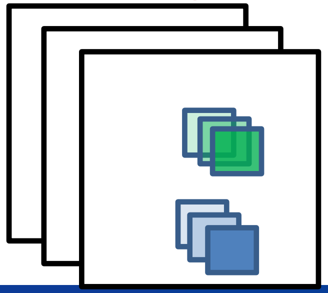
Batch Normalization, Ioffe and Szegedy, ICML 2015

- Computer Vision Community:

  Local statistics in an sample

Jarrett et al, 2009

Krizhevsky et al, 2012



Ren et al, 2017

Ortiz et al, 2020

Batch Normalization (BN)

Input: Values of x over a mini-batch: $\mathcal{B} = \{x_{1\ldots m}\}$;
Parameters to be learned: γ, β

Output: $\{y_i = \mathrm{BN}_{\gamma,\beta}(x_i)\}$

$$\mu_\mathcal{B} \leftarrow \frac{1}{m}\sum_{i=1}^{m} x_i \qquad \text{// mini-batch mean}$$

$$\sigma_\mathcal{B}^2 \leftarrow \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu_\mathcal{B})^2 \qquad \text{// mini-batch variance}$$

$$\widehat{x}_i \leftarrow \frac{x_i - \mu_\mathcal{B}}{\sqrt{\sigma_\mathcal{B}^2 + \epsilon}} \qquad \text{// normalize}$$

$$y_i \leftarrow \gamma\widehat{x}_i + \beta \equiv \mathrm{BN}_{\gamma,\beta}(x_i) \qquad \text{// scale and shift}$$

Normalize over mini-batch data

Extra learnable scale and bias

Back-propagate through the transformation

$$\frac{\partial \ell}{\partial \sigma_\mathcal{B}^2} = \sum_{i=1}^{m} \frac{\partial \ell}{\partial \widehat{x}_i} \cdot (x_i - \mu_\mathcal{B}) \cdot \frac{-1}{2}(\sigma_\mathcal{B}^2 + \epsilon)^{-3/2}$$

$$\frac{\partial \ell}{\partial \mu_\mathcal{B}} = \left(\sum_{i=1}^{m} \frac{\partial \ell}{\partial \widehat{x}_i} \cdot \frac{-1}{\sqrt{\sigma_\mathcal{B}^2+\epsilon}}\right) + \frac{\partial \ell}{\partial \sigma_\mathcal{B}^2} \cdot \frac{\sum_{i=1}^{m} -2(x_i-\mu_\mathcal{B})}{m}$$

$$\frac{\partial \ell}{\partial x_i} = \frac{\partial \ell}{\partial \widehat{x}_i} \cdot \frac{1}{\sqrt{\sigma_\mathcal{B}^2+\epsilon}} + \frac{\partial \ell}{\partial \sigma_\mathcal{B}^2} \cdot \frac{2(x_i-\mu_\mathcal{B})}{m} + \frac{\partial \ell}{\partial \mu_\mathcal{B}} \cdot \frac{1}{m}$$

Batch normalization accelerating deep network training by reducing internal covariate shift[Ioffe and Szegedy, ICML 2015]

Batch Normalization (BN)

Input: Values of x over a mini-batch: $\mathcal{B} = \{x_{1...m}\}$;
Parameters to be learned: γ, β

Output: $\{y_i = \text{BN}_{\gamma,\beta}(x_i)\}$

$$\mu_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m} x_i \qquad \text{// mini-batch mean}$$

$$\sigma_{\mathcal{B}}^2 \leftarrow \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu_{\mathcal{B}})^2 \qquad \text{// mini-batch variance}$$

$$\widehat{x}_i \leftarrow \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma_{\mathcal{B}}^2 + \epsilon}} \qquad \text{// normalize}$$

$$y_i \leftarrow \gamma\widehat{x}_i + \beta \equiv \text{BN}_{\gamma,\beta}(x_i) \qquad \text{// scale and shift}$$

Normalize over mini-batch data

mini-batch statistics for training and population statistics for inference

$$\begin{cases} \hat{u} = (1-\lambda)\hat{u} + \lambda u, \\ \hat{\sigma}^2 = (1-\lambda)\hat{\sigma}^2 + \lambda\sigma^2 \end{cases}$$

Back-propagate through the transformation

$$\frac{\partial \ell}{\partial \sigma_{\mathcal{B}}^2} = \sum_{i=1}^{m} \frac{\partial \ell}{\partial \widehat{x}_i} \cdot (x_i - \mu_{\mathcal{B}}) \cdot \frac{-1}{2}(\sigma_{\mathcal{B}}^2 + \epsilon)^{-3/2}$$

$$\frac{\partial \ell}{\partial \mu_{\mathcal{B}}} = \left(\sum_{i=1}^{m} \frac{\partial \ell}{\partial \widehat{x}_i} \cdot \frac{-1}{\sqrt{\sigma_{\mathcal{B}}^2+\epsilon}}\right) + \frac{\partial \ell}{\partial \sigma_{\mathcal{B}}^2} \cdot \frac{\sum_{i=1}^{m} -2(x_i - \mu_{\mathcal{B}})}{m}$$

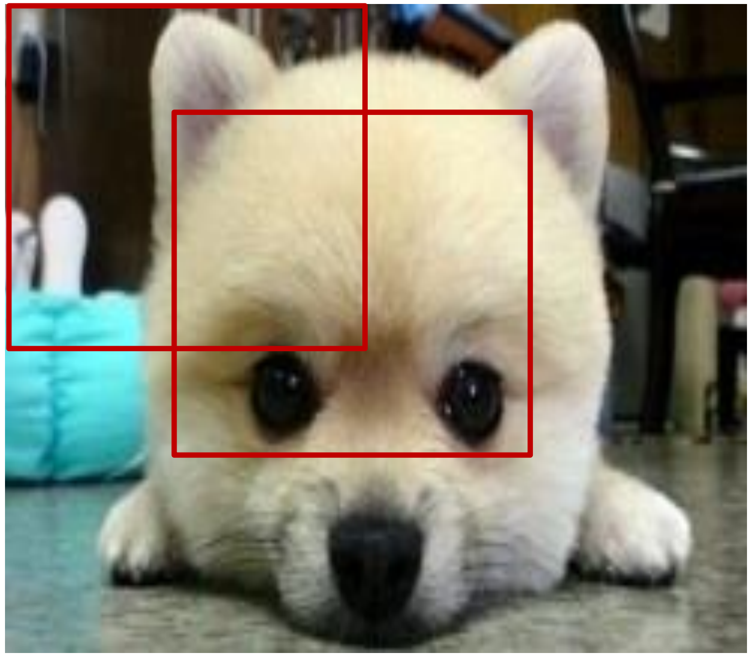
$$\frac{\partial \ell}{\partial x_i} = \frac{\partial \ell}{\partial \widehat{x}_i} \cdot \frac{1}{\sqrt{\sigma_{\mathcal{B}}^2+\epsilon}} + \frac{\partial \ell}{\partial \sigma_{\mathcal{B}}^2} \cdot \frac{2(x_i - \mu_{\mathcal{B}})}{m} + \frac{\partial \ell}{\partial \mu_{\mathcal{B}}} \cdot \frac{1}{m}$$

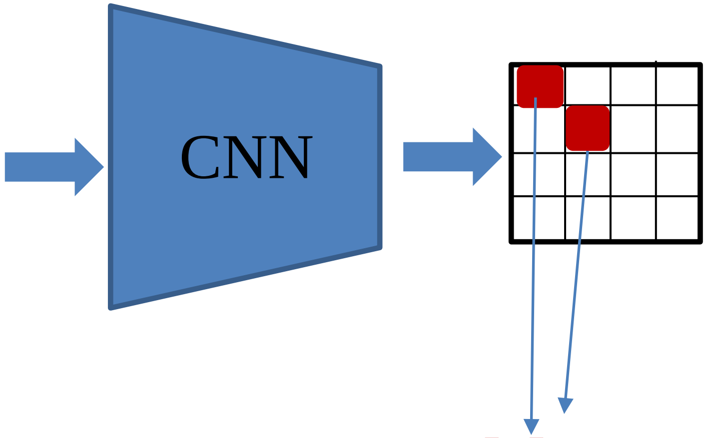
Extra learnable scale and bias

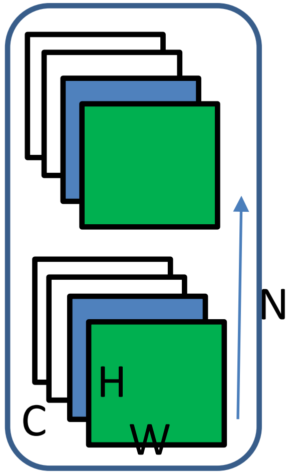
Batch normalization accelerating deep network training by reducing internal covariate shift[Ioffe and Szegedy, ICML 2015]

Batch Normalization (BN)

- Extension to CNN input
  - Jarrett et al, 2009; Gulcehre and Bengio, 2013





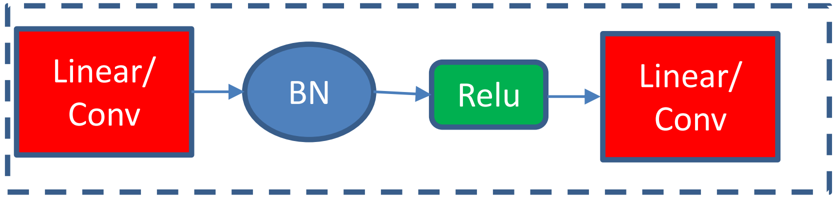


$\mathbf{X} \in \mathcal{R}^{N \times C \times H \times W}$

Spatial location as an example for normalization

Batch normalization accelerating deep network training by reducing internal covariate shift[Ioffe and Szegedy, ICML 2015]

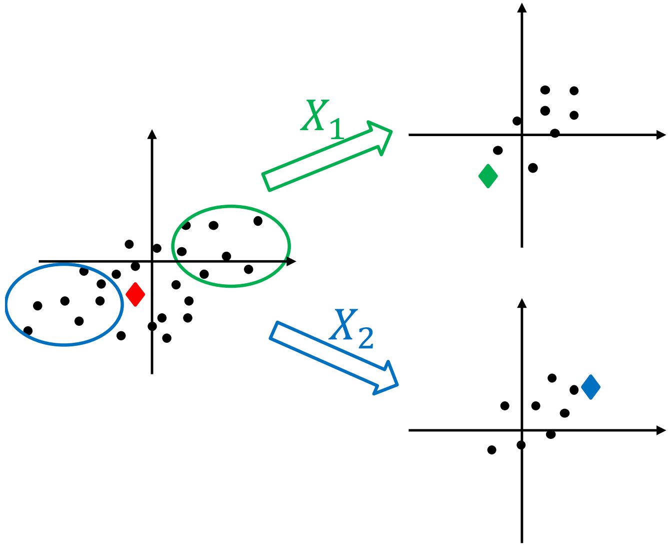
Batch Normalization: Property



- For accelerating training
  - Stable training
    - Weight scale invariant: not sensitive for weight initialization

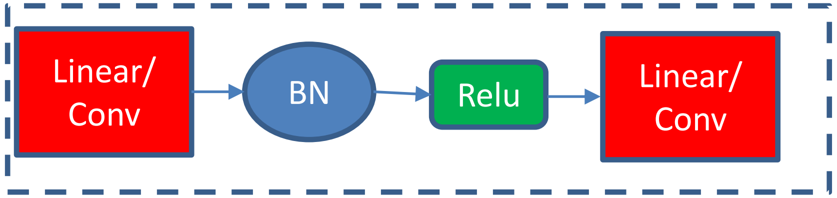
$$\mathrm{BN}(W\mathrm{u}) = \mathrm{BN}((aW)\mathrm{u})$$

  - Better conditioning
    - Can use large learning rate



- For generalization
  - Introduced stochasticity
    - Mini-batch dependence during training
    - Training-test discrepancy
  - Scale invariant representation

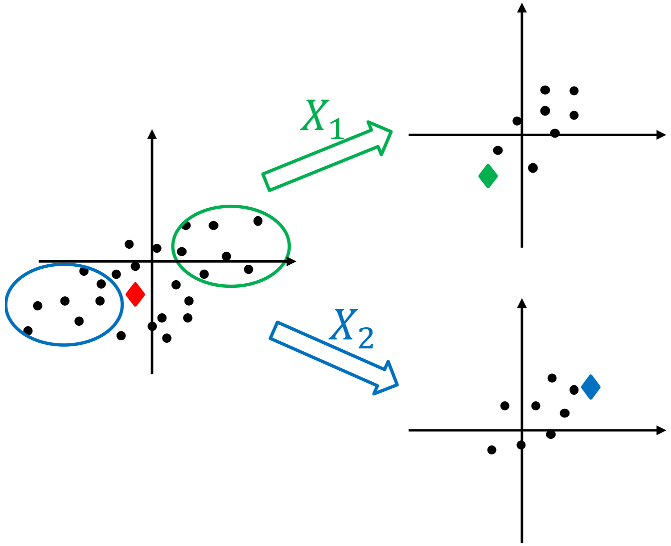
Batch Normalization: Property



- For accelerating training
  - Stable training
    - Weight scale invariant: not sensitive for weight initialization

$$\mathrm{BN}(W\mathrm{u}) = \mathrm{BN}((aW)\mathrm{u})$$

  - Better conditioning
    - Can use large learning rate
- For generalization
  - Introduced stochasticity
    - Mini-batch dependence during training
    - Training-test discrepancy
  - Scale invariant representation



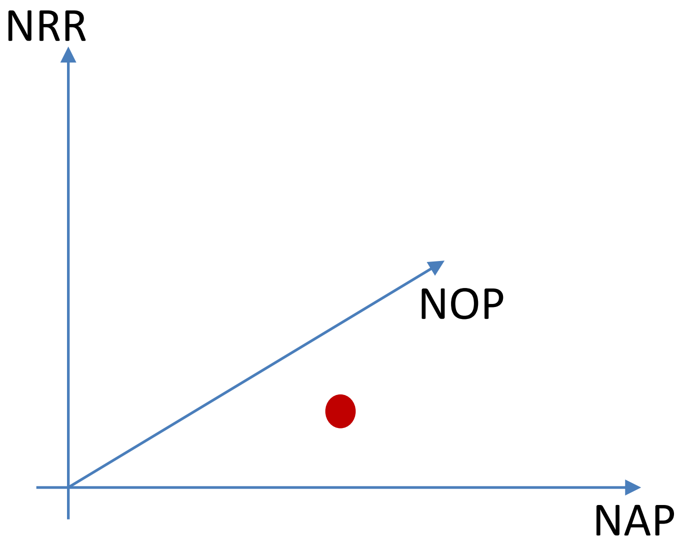
- Independent population statistics or sharing statistics in RNN?
- Data are from different domains
- Small batch size problem

Outline

- Normalizing activations as functions
  - A Framework for decomposing normalization
  - Multi-Mode and combinational normalization
  - BN for more robust estimation

A Framework for Decomposing Normalization

- The framework
  - Normalization Area Partitioning (NAP): which area to calculate the ‘statistics’
  - Normalization Operation (NOP): what kind of normalization operation?
  - Normalization Representation Recovery (NRR)



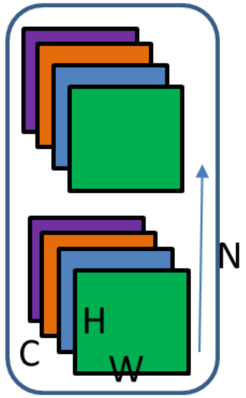
Algorithm 1 Framework of algorithms normalizing activations as functions.

1: **Input**: mini-batch inputs $\mathbf{X} \in \mathbb{R}^{d\times m\times h\times w}$.
2: **Output**: $\widetilde{\mathbf{X}} \in \mathbb{R}^{d\times m\times h\times w}$.
3: Normalization area partitioning: $\boldsymbol{X} = \Pi(\mathbf{X})$.
4: Normalization operation: $\widehat{\boldsymbol{X}} = \Phi(\boldsymbol{X})$.
5: Normalization representation recovery: $\widetilde{\boldsymbol{X}} = \Psi(\widehat{\boldsymbol{X}})$.
6: Reshape back: $\widetilde{\mathbf{X}} = \Pi^{-1}(\widetilde{\boldsymbol{X}})$.

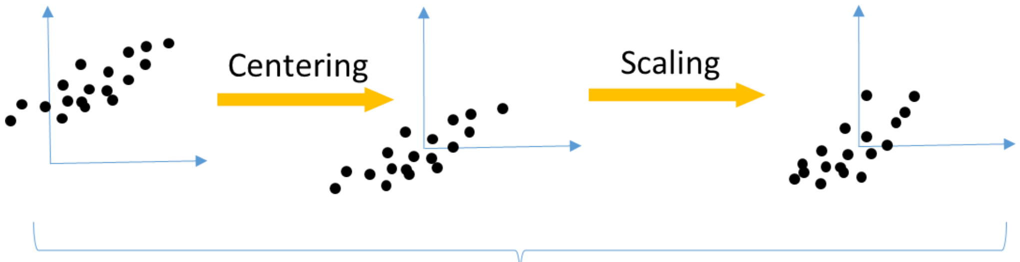
A Framework for Decomposing Normalization

- Batch Normalization
  - NAP: $\boldsymbol{X} = \Pi_{BN}(\mathbf{X}) \in \mathbb{R}^{d\times mhw}$

$\mathbf{X} \in \mathcal{R}^{N\times C\times H\times W}$



  - NOP: $\widehat{\boldsymbol{X}} = \Phi_{SD}(\boldsymbol{X}) = \Lambda^{-\frac{1}{2}}(\boldsymbol{X} - \mathbf{u}\mathbf{1}^T)$.



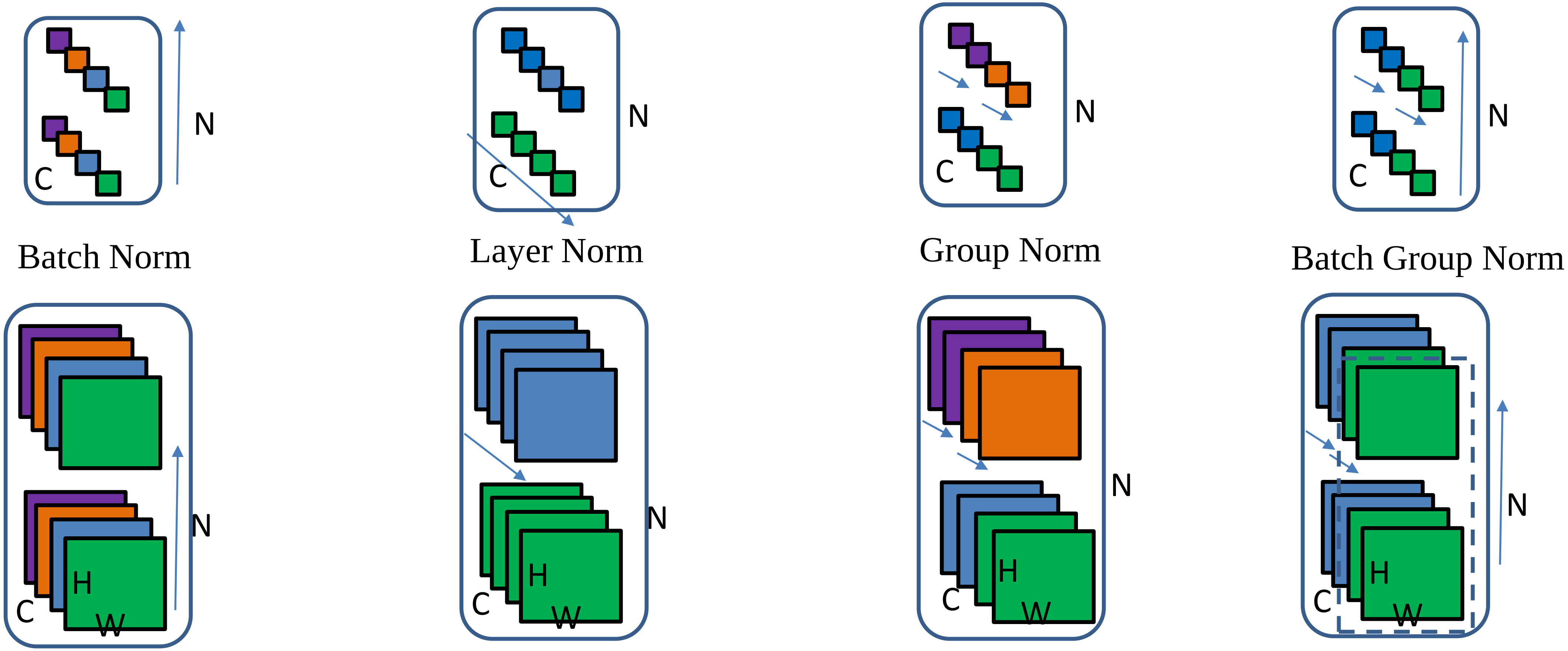
  - NRR: $\widetilde{\boldsymbol{X}} = \Psi_{AF}(\widehat{\boldsymbol{X}}) = \widehat{\boldsymbol{X}} \odot (\gamma\mathbf{1}^T) + (\beta\mathbf{1}^T)$

Outline

- Normalizing activations as functions
  - A Framework for decomposing normalization
    - Normalization Area Partitioning (NAP)
    - Normalization Operation (NOP)
    - Normalization Representation Recovery (NRR)
  - Multi-Mode and combinational normalization
  - BN for more robust estimation

Normalization Area Partitioning

- MLP input: $X \in \mathcal{R}^{N \times C}$

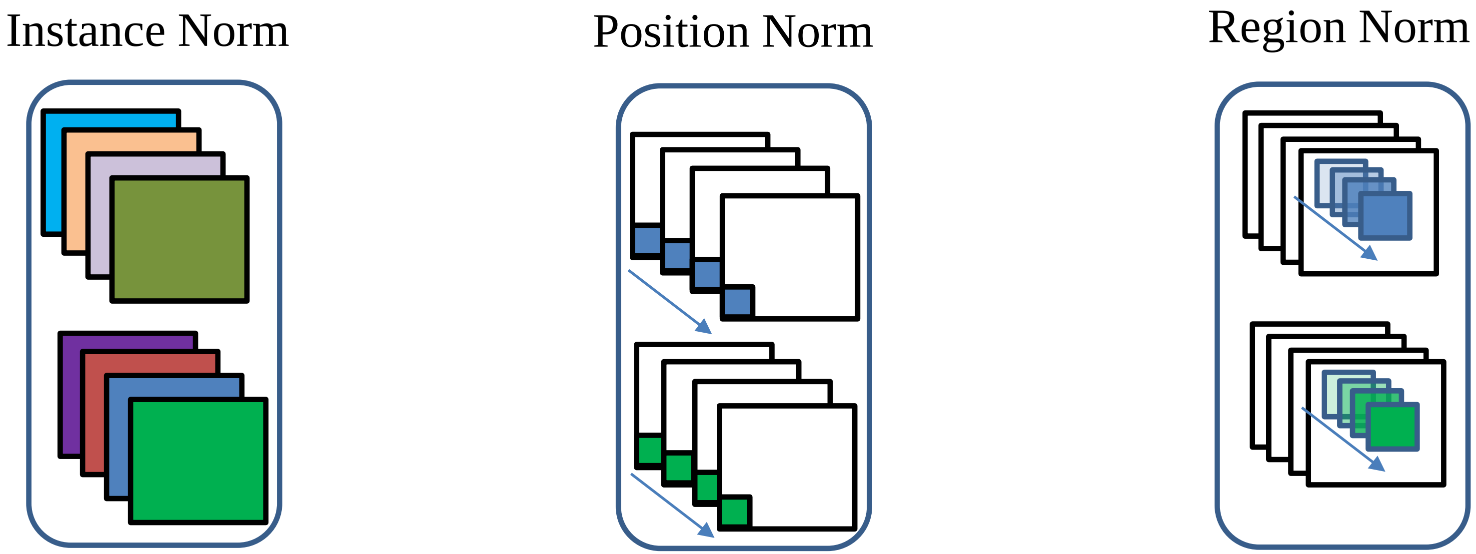


- CNN input: $\mathbf{X} \in \mathcal{R}^{N \times C \times H \times W}$

[Ioffe and Szegedy, ICML 2015; Ba et al, 2016; Wu and He, ECCV 2018; Summers and Dinneen, ICLR 2020]

Normalization Area Partitioning

- CNN Input: $X \in \mathcal{R}^{N\times C\times H\times W}$



[Ulyanov et al, 2016; Ba et al, 2016; Li et al, NeurIPS 2019; Yu et al, AAAI 2020]

Outline

- Normalizing activations as functions
  - A Framework for decomposing normalization
    - Normalization Area Partitioning (NAP)
    - Normalization Operation (NOP)
    - Normalization Representation Recovery (NRR)
  - Multi-Mode and combinational normalization
  - BN for more robust estimation

Normalization Operation

- Batch Whitening (BW)

Activation distribution

Standardization:

$$\hat{X} = \varphi(X) = \left(diag(\Sigma)\right)^{-\frac{1}{2}}(X - \mu 1^T)$$

$$diag(\hat{X}\hat{X}^T) = I$$

Covariance $matrix$

Whitening:

$$\hat{X} = \varphi(X) = \Sigma^{-\frac{1}{2}}(X - \mu 1^T)$$

$$\hat{X}\hat{X}^T = I$$

Standardization is a special case of whitening

Whitening further improver conditioning over standardization

Decorrelated Batch Normalization [Huang et al, CVPR 2018]

Normalization Operation

- Batch Whitening (BW)

➢ Forward

$$\mu = \frac{1}{m}\sum_{j=1}^{m}\mathbf{x}_j$$

$$\Sigma = \frac{1}{m}\sum_{j=1}^{m}(\mathbf{x}_j-\mu)(\mathbf{x}_j-\mu)^T$$

$$\Sigma = \mathbf{D}\Lambda\mathbf{D}^T$$

$$\mathbf{U} = \Lambda^{-1/2}\mathbf{D}^T$$

$$\tilde{\mathbf{x}}_i = \mathbf{U}(\mathbf{x}_i-\mu)$$

$$\hat{\mathbf{x}}_i = \mathbf{D}\tilde{\mathbf{x}}_i$$

➢ Backward

$$\frac{\partial L}{\partial \tilde{\mathbf{x}}_i} = \frac{\partial L}{\partial \hat{\mathbf{x}}_i}\mathbf{D}$$

$$\frac{\partial L}{\partial \mathbf{U}} = \sum_{i=1}^{m}\frac{\partial L}{\partial \tilde{\mathbf{x}}_i}^T(\mathbf{x}_i-\mu)^T$$

$$\frac{\partial L}{\partial \Lambda} = (\frac{\partial L}{\partial \mathbf{U}})\mathbf{D}(-\frac{1}{2}\Lambda^{-3/2})$$

$$\frac{\partial L}{\partial \mathbf{D}} = \frac{\partial L}{\partial \mathbf{U}}^T\Lambda^{-1/2} + \sum_{i=1}^{m}\frac{\partial L}{\partial \hat{\mathbf{x}}_i}^T\tilde{\mathbf{x}}_i^T$$

[Ionescu et al, ICCV 2015]

$$\boxed{\frac{\partial L}{\partial \Sigma} = \mathbf{D}\{(\mathbf{K}^T\odot(\mathbf{D}^T\frac{\partial L}{\partial \mathbf{D}})) + (\frac{\partial L}{\partial \Lambda})_{diag}\}\mathbf{D}^T}$$

$$\frac{\partial L}{\partial \mu} = \sum_{i=1}^{m}\frac{\partial L}{\partial \tilde{\mathbf{x}}_i}(-\mathbf{U}) + \sum_{i=1}^{m}\frac{-2(\mathbf{x}_i-\mu)^T}{m}(\frac{\partial L}{\partial \Sigma})_{sym}$$

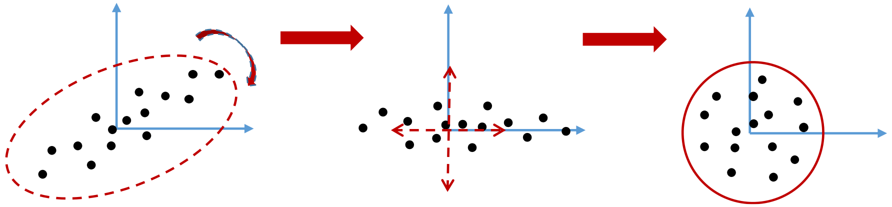
$$\frac{\partial L}{\partial \mathbf{x}_i} = \frac{\partial L}{\partial \tilde{\mathbf{x}}_i}\mathbf{U} + \frac{2(\mathbf{x}_i-\mu)^T}{m}(\frac{\partial L}{\partial \Sigma})_{sym} + \frac{1}{m}\frac{\partial L}{\partial \mu}$$

Decorrelated Batch Normalization [Huang et al, CVPR 2018]

Batch Whitening

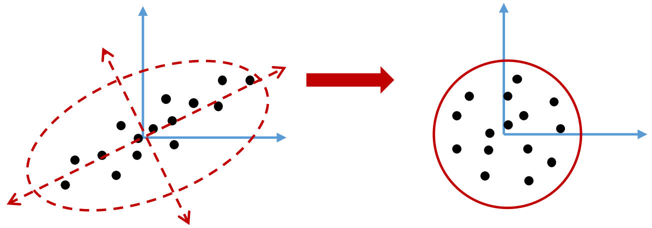
- Whitening
  - PCA whitening not work

$$G_{PCA} = \Lambda^{-\frac{1}{2}} D^T, \qquad D\Lambda D^T = \Sigma$$

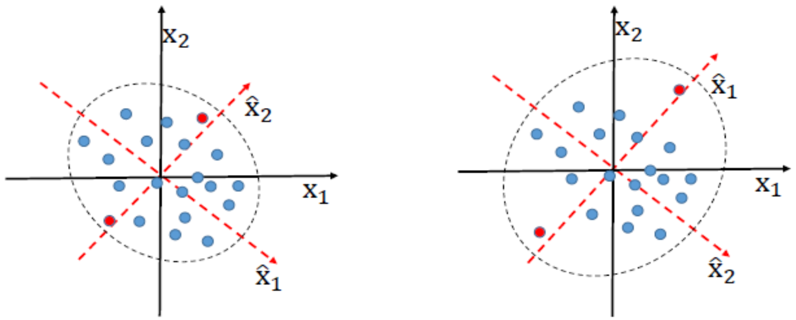


  - ZCA whitening work

$$G_{ZCA} = D\Lambda^{-\frac{1}{2}} D^T$$

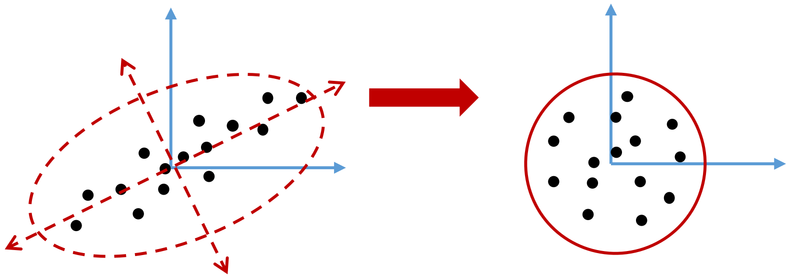


- PCA cause stochastic axis swapping

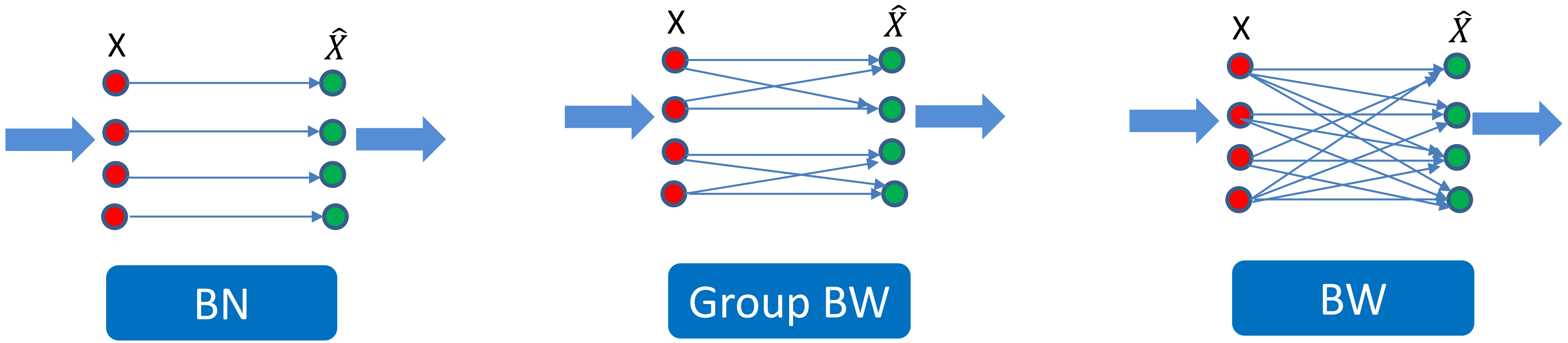


Control the Extent of Batch Whitening

- What's the problem of full whitening
  - Computational cost
  - Overmuch constraints on $\hat{X}$ ($\hat{X}\hat{X}^T = I$)
  - Difficulty in estimating the population statistics



- Group based batch whitening



Control the Extent of Batch Whitening

- Newton's iteration to calculate whitening matrix

$$\hat{X} = \varphi(X) = \Sigma^{-\frac{1}{2}}(X - \mu 1^T)$$

Normalize eigenvalues: $\Sigma_N = \Sigma / tr(\Sigma)$

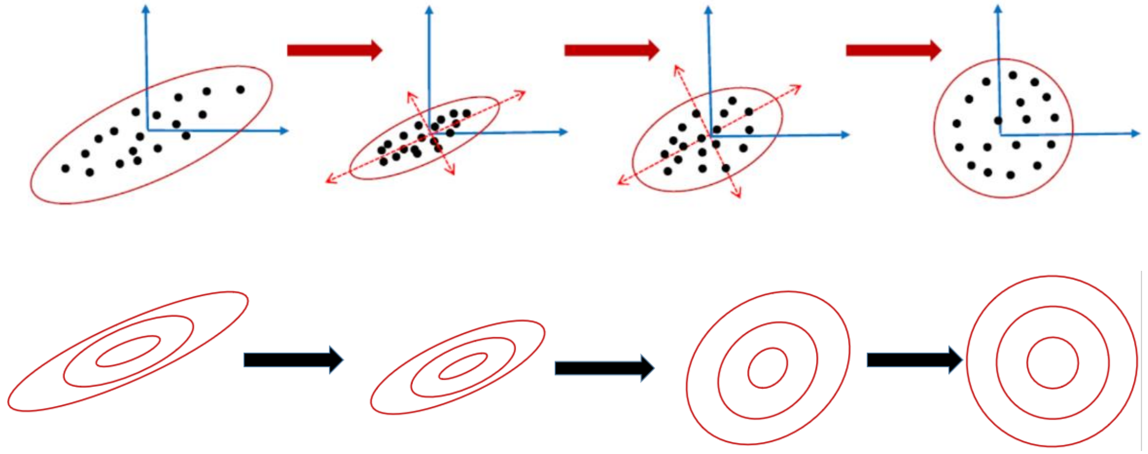
Iteration:

$\mathbf{P}_0 = \mathbf{I}.$
for $k = 1$ to T **do**
$\mathbf{P}_k = \frac{1}{2}(3\mathbf{P}_{k-1} - \mathbf{P}_{k-1}^3 \Sigma_N)$
end for

Whitening matrix: $\Sigma^{-\frac{1}{2}} = \mathbf{P}_T / \sqrt{tr(\Sigma)}$

Activation distribution

Optimization landscape



Iterative Normalization: Beyond Standardization towards Efficient Whitening [Huang et al, CVPR 2019]

Batch Whitening

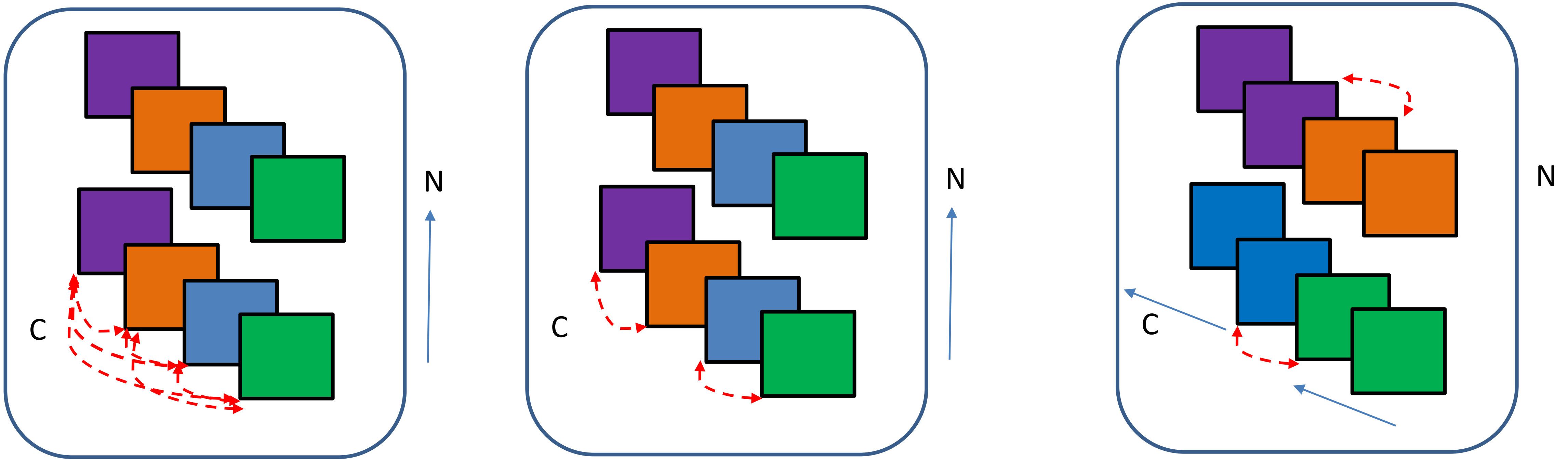
- Advantages over standardization
  - Better conditioning theoretically
  - Probably better generalization (the amplified stochasticity) by controlling the extent of whitening

- Disadvantages
  - Computational costs ⟹ Group based, Newtown's iteration
  - Numerical instability ⟹ Cholesky decomposition, Newtown's iteration
  - More difficulty in ensuring the training and inference consistency

[Huang et al CVPR 2018; Siarohin et al ICLR 2019; Huang et al CVPR 2019; Ye et al ICLR 2020; Huang et al, CVPR 2020]

Group whitening

- Exploiting the advantages of whitening and avoid the disadvantages of normalization over batch



BW | Group based BW | Group Whitening

Group Whitening: Balancing Learning Efficiency and Representational Capacity [Huang et al, CVPR 2021]

Normalization Operation

- Variations of standardization

$$\hat{x}^{(i)} = \frac{x^{(i)} - u}{\sqrt{\sigma^2 + \epsilon}}$$

  – L^2 Norm (BN): $\sigma^2 = \frac{1}{m}\sum_{i=1}^{m}(x^{(i)} - u)^2$

  – L^1 Norm : $\sigma = \frac{1}{m}\sum_{i=1}^{m}|x^{(i)} - u|$

  – L^∞ Norm : $\sigma = \max_i |x^{(i)}|$

  – More general L^p: $\sigma = \frac{1}{m}\sqrt[p]{\sum_{i=1}^{m}(x^{(i)})^p}$

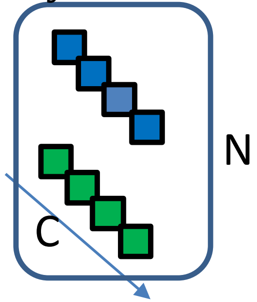
$\left.\begin{matrix} L^1 \\ L^\infty \end{matrix}\right\}$ more efficient and numerical stability in a low precision implementation

[Liao et al, 2016; Wu et al, 2018; Hoffer et al, 2018]

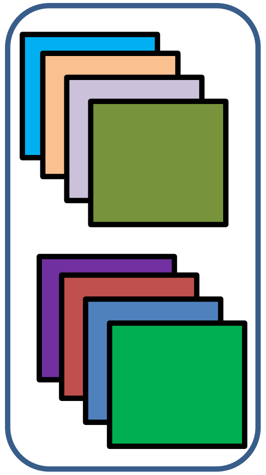
Normalization Operation

- Reduced standardization
  - Centering only (Mean only BN)
  - Scaling only

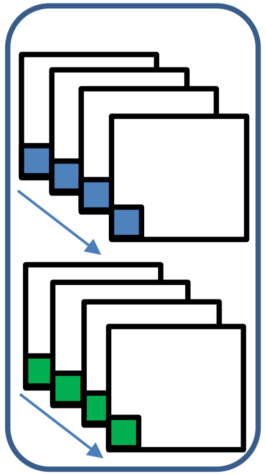
Root Mean Square Layer Norm



Filter Response Normalization



Pixel Normalization



Standardization: Layer Normalization | Instance Normalization | Position Normalization

[Karras, et al, ICLR 2018; Zhang and Sennrich, NeurIPS 2019; Singh and Krishnan, CVPR2020et al, 2018]

Outline

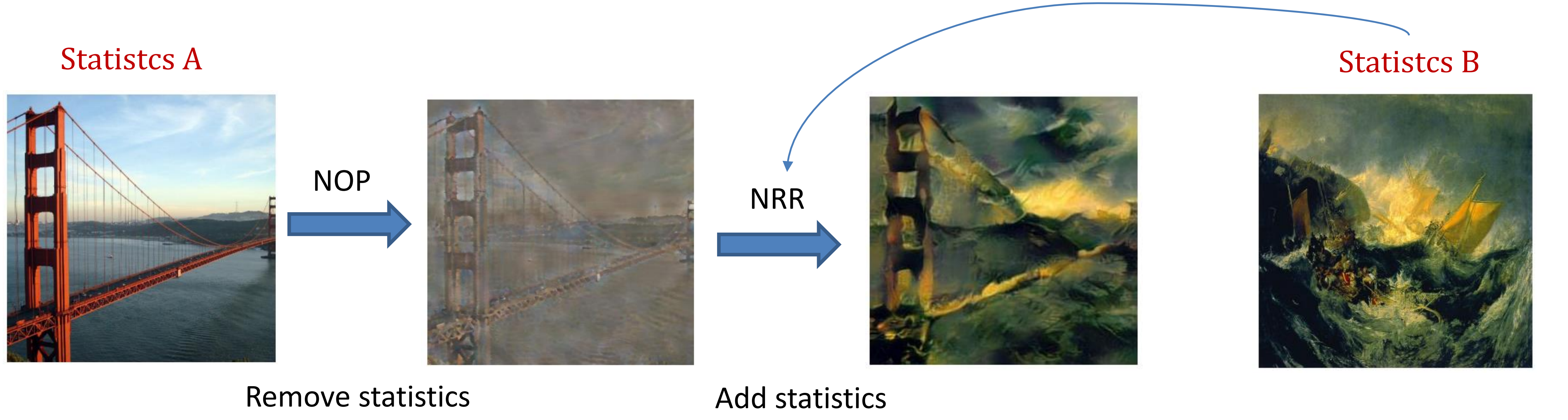
- Normalizing activations as functions
  - A Framework for decomposing normalization
    - Normalization Area Partitioning (NAP)
    - Normalization Operation (NOP)
    - Normalization Representation Recovery (NRR)
  - Multi-Mode and combinational normalization
  - BN for more robust estimation

Normalization Representation Recovery

- Why NRR
  - Recover the representation
  - Edit the statistical distribution

NOP: $\widehat{\boldsymbol{X}} = \Phi_{SD}(\boldsymbol{X}) = \Lambda^{-\frac{1}{2}}(\boldsymbol{X} - \mathbf{u}\mathbf{1}^T)$

NRR: $\widetilde{\boldsymbol{X}} = \Psi_{AF}(\widehat{\boldsymbol{X}}) = \widehat{\boldsymbol{X}} \odot (\gamma\mathbf{1}^T) + (\beta\mathbf{1}^T)$

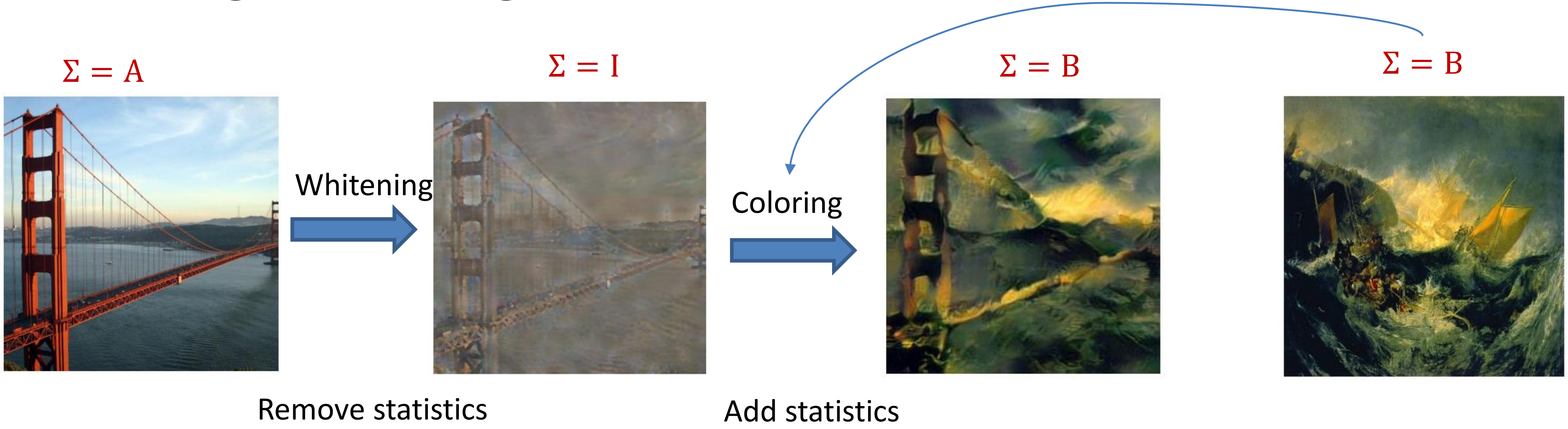


Normalization Representation Recovery

- More general coloring transformation

$$\widetilde{\boldsymbol{X}} = \Psi_{LR}(\widehat{\boldsymbol{X}}) = \widehat{\boldsymbol{X}}\mathbf{W} + (\beta\mathbf{1}^T)$$

- Whitening + Coloring



Dynamic Generate NRR

- Dynamic generate the affine parameters

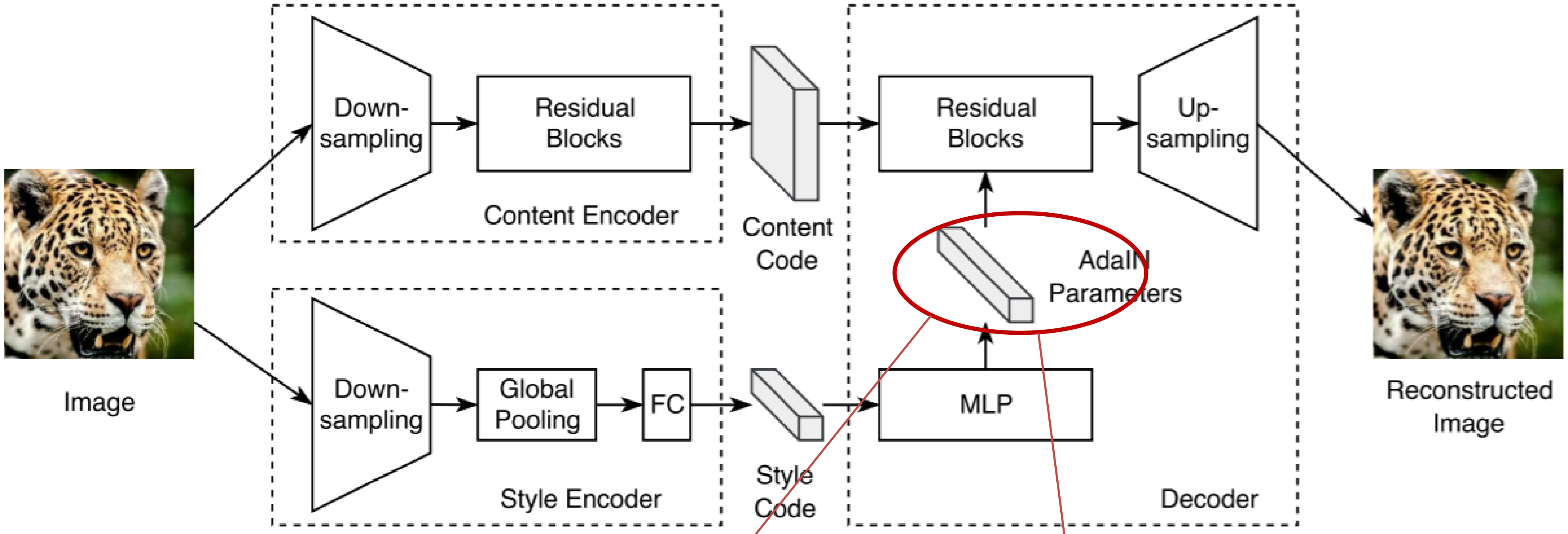
$$\widetilde{\boldsymbol{X}} = \Psi_{DC}(\widehat{\boldsymbol{X}}) = \widehat{\boldsymbol{X}} \odot \Gamma_{\phi^\gamma} + B_{\phi^\beta}$$

where $\Gamma_{\phi^\gamma} \in \mathbb{R}^{d\times m}$ and $B_{\phi^\beta} \in \mathbb{R}^{d\times m}$ are generated by the subnetworks $\phi^\gamma_{\theta_\gamma}(\cdot)$ and $\phi^\beta_{\theta_\beta}(\cdot)$, respectively.

  – Dynamic layer normalization [Kim et al, 2017]

Dynamic Generate NRR

- Adaptive instance normalization

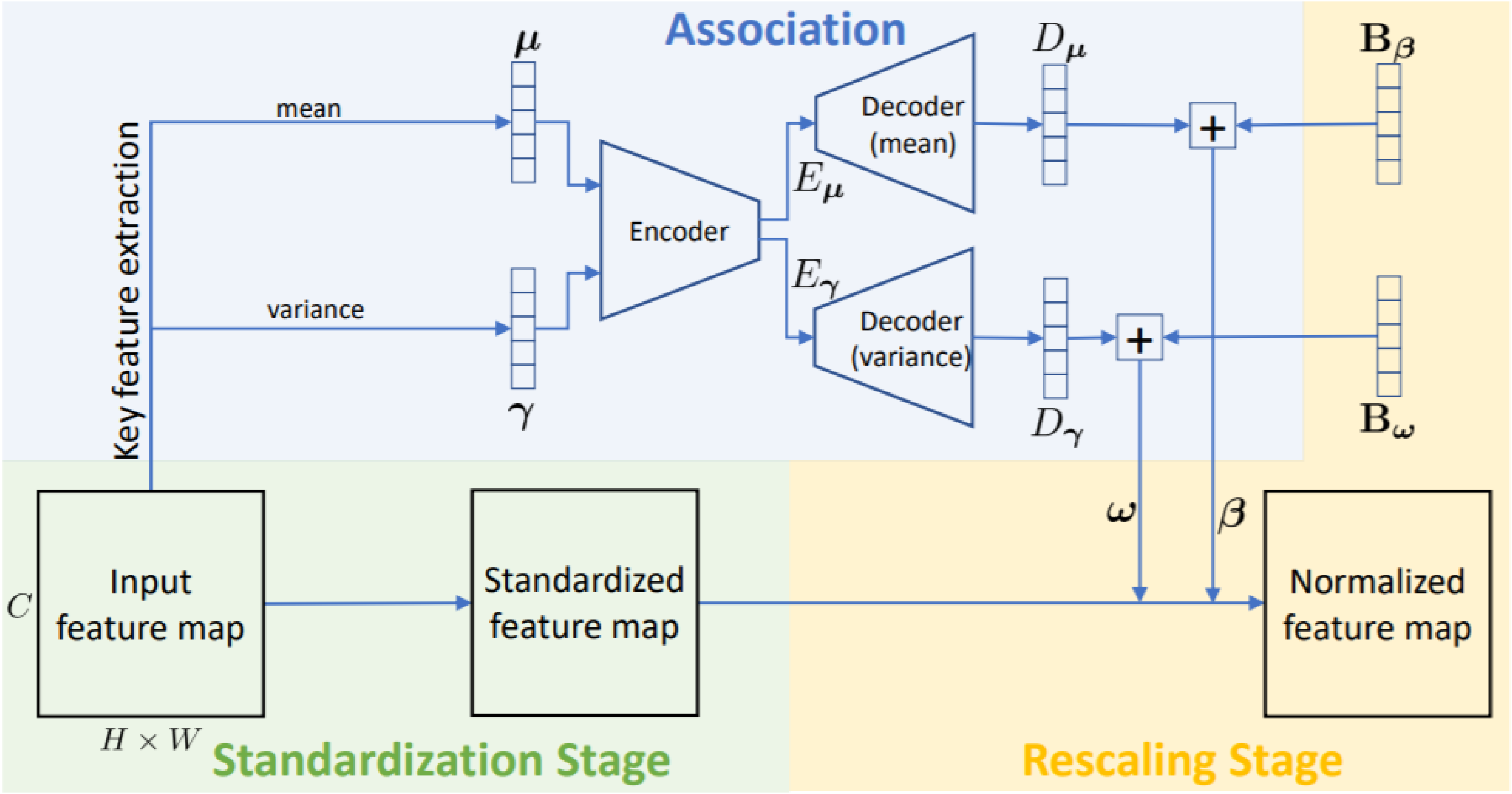


$$\mathrm{AdaIN}(z,\gamma,\beta)=\gamma\left(\frac{z-\mu(z)}{\sigma(z)}\right)+\beta$$

Multimodal Unsupervised Image-to-Image Translation [Huang et al, ECCV 2018]

Dynamic Generate NRR

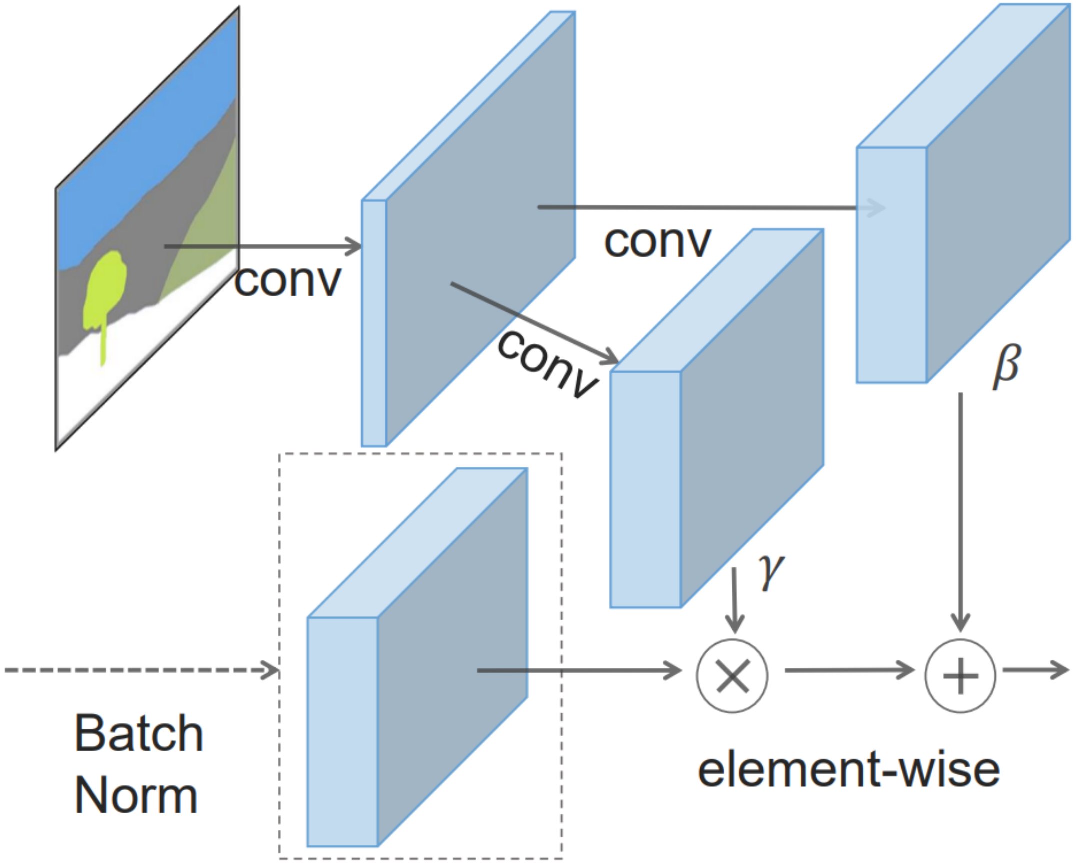
- Instance level-meta norm



Instance-Level Meta Normalization [Jia et al, CVPR 2019]

Dynamic Generate NRR

- Spatially Adaptive Denormalization (SPADE)

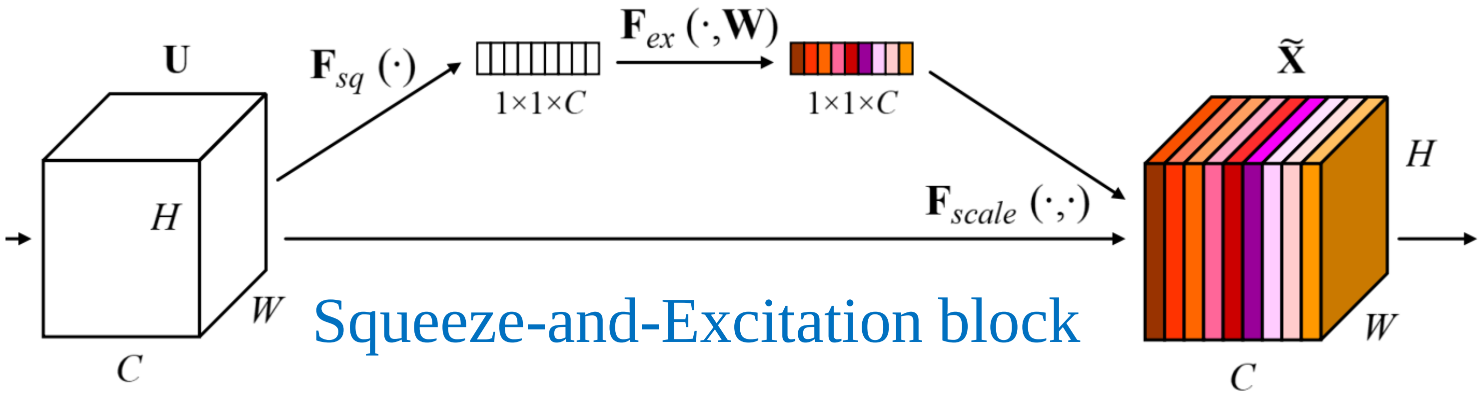


$$\beta, \gamma \in \mathbb{R}^{d \times h \times w}$$

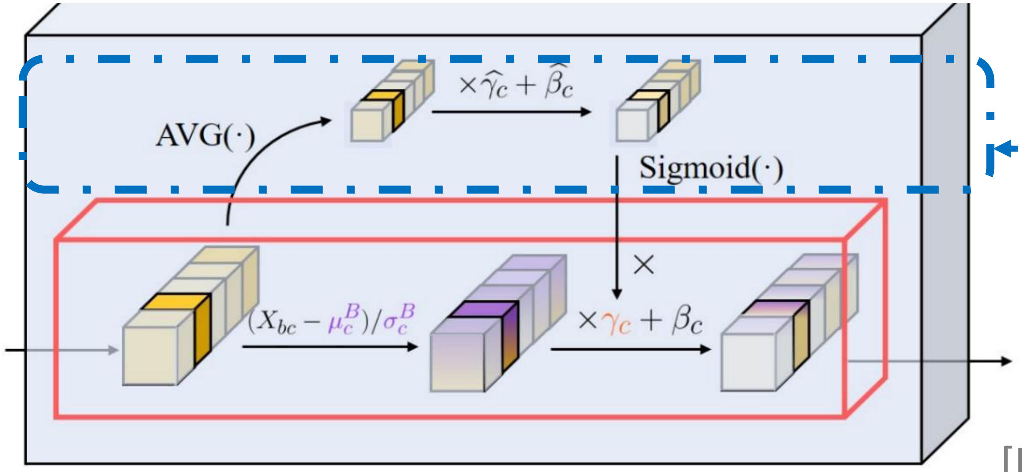
Semantic Image Synthesis with Spatially-Adaptive Normalization [Park et al, CVPR 2019]

Dynamic Generate NRR

- The explanation of channel attention
  - Instance Enhancement Batch Normalization (IEBN)



Squeeze-and-Excitation block

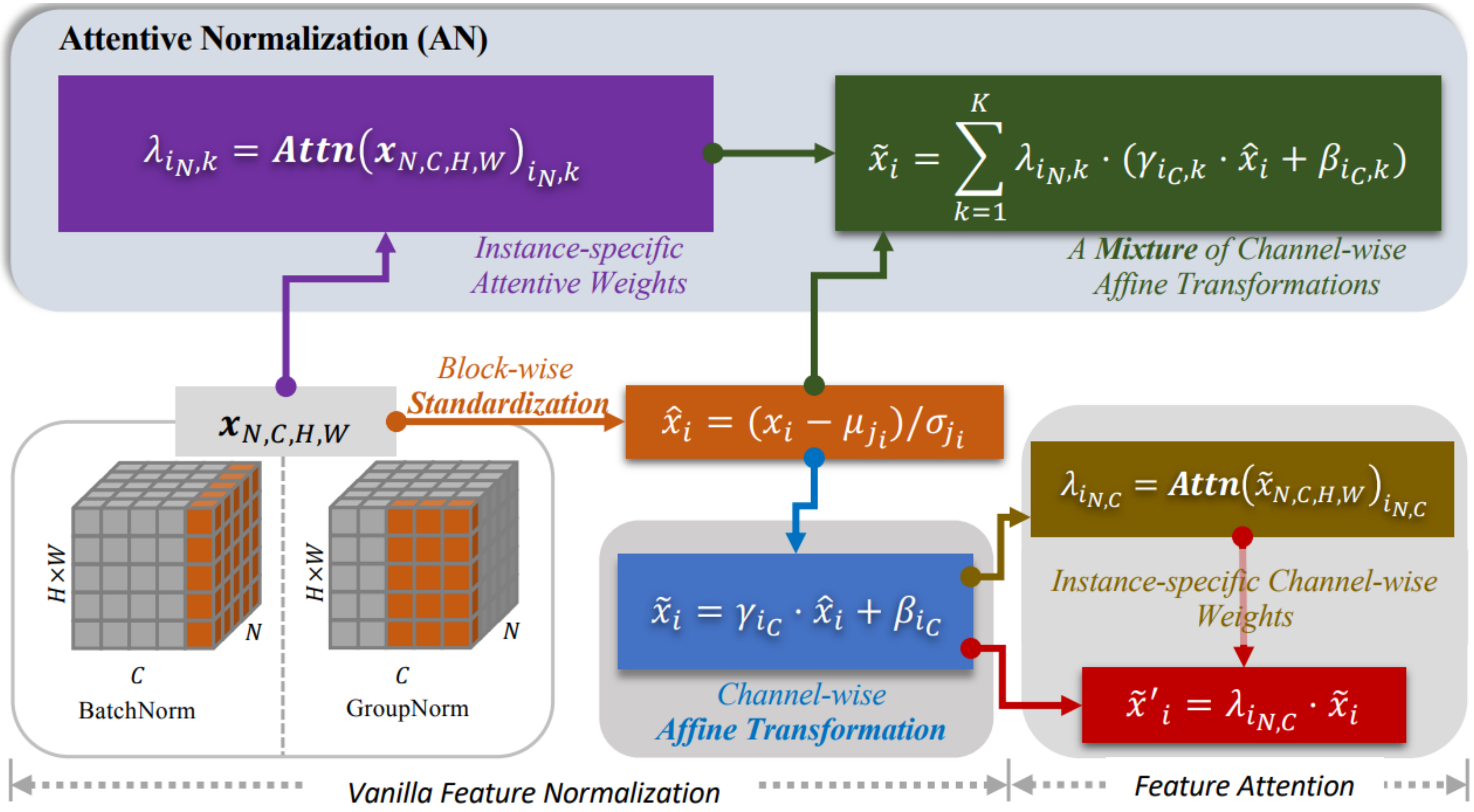


Channel attention

[Hu et al, CVPR 2018; Liang et al, AAAI 2020]

Dynamic Generate NRR

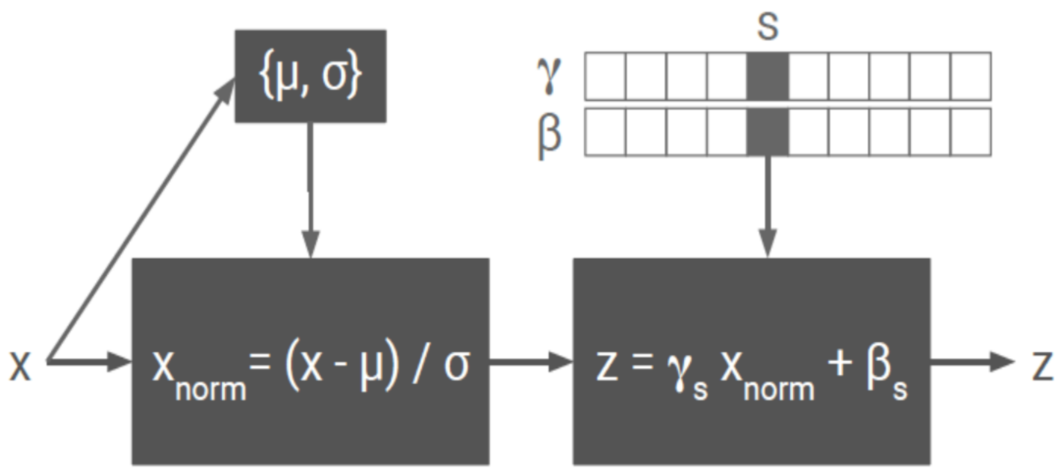
- Attentive Normalization (AN)

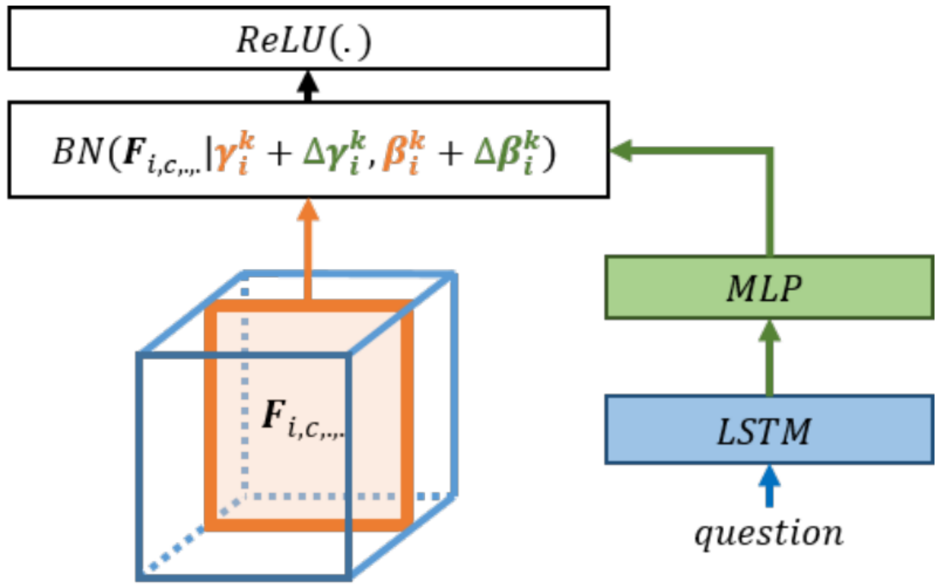


Attentive Normalization [Li et al, ECCV 2020]

Side Information NRR

- Conditional instance normalization
- Conditional batch normalization





A Learned Representation For Artistic Style [Dumoulin et al, ICLR 2017)]

Modulating early visual processing by language [Vries et al, NeurIPS 2017]

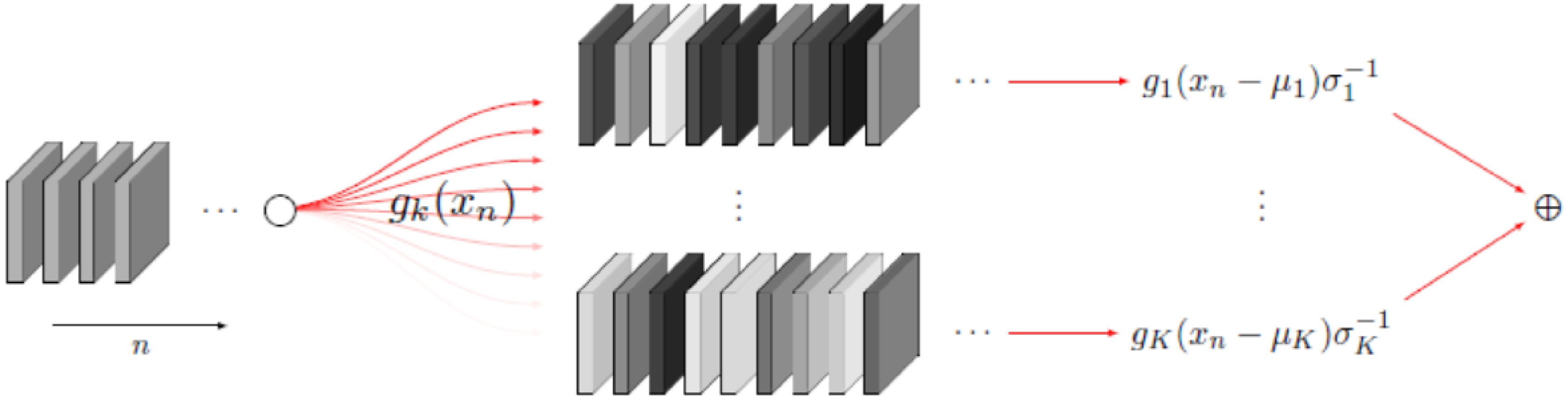
Outline

- Normalizing activations as functions
  - A Framework for decomposing normalization
    - Normalization Area Partitioning (NAP)
    - Normalization Operation (NOP)
    - Normalization Representation Recovery (NRR)
  - Multi-Mode and combinational normalization
  - BN for more robust estimation

Multi-mode Normalization

- Mode normalization

$$\mathrm{MN}(x_n) \triangleq \alpha\left(\sum_{k=1}^{K} g_k(x_n)\frac{x_n-\mu_k}{\sigma_k}\right)+\beta$$



Mode Normalization [Deecke et al, ICLR 2019]

Combinational Normalization

- Switchable Normalization (SN)
  - Combing BN, LN and IN

$$\hat{x}_{nchw} = \gamma \frac{x_{nchw} - (w_{IN}\mu_{IN} + w_{BN}\mu_{BN} + w_{LN}\mu_{LN})}{\sqrt{w'_{IN}\sigma^2_{IN} + w'_{BN}\sigma^2_{BN} + w'_{LN}\sigma^2_{LN}}} + \beta$$

$$w_k = \frac{e^{\lambda_k}}{\sum_{z\in\{IN,LN,BN\}} e^{\lambda_z}}, k \in \{IN, LN, BN\}$$

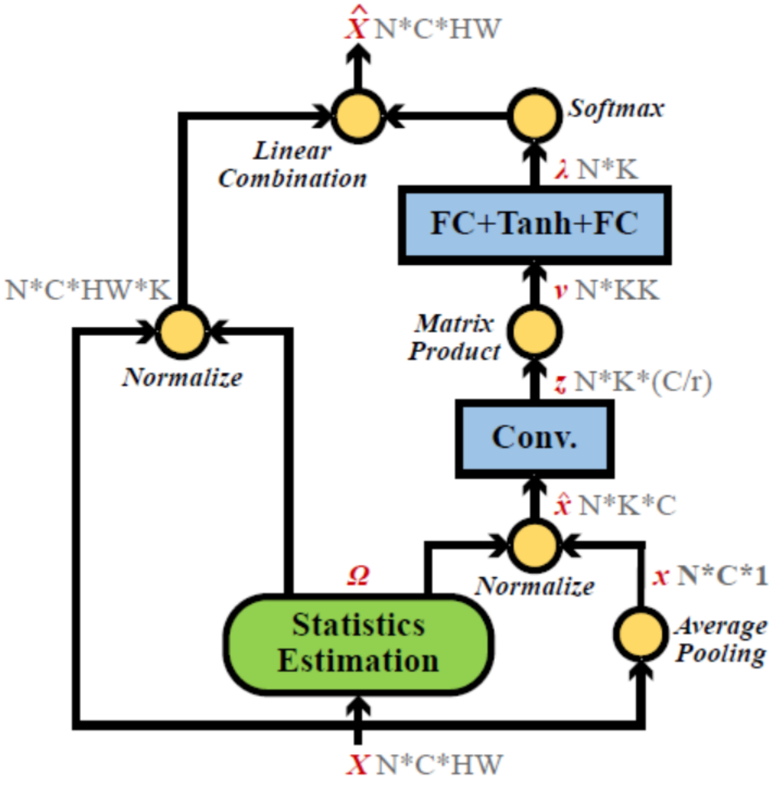
- Switchable Whitening (SW)
  - $k \in \{BW, IW\}$ or $k \in \{BW, IW, IN, LN, BN\}$

[Luo et al, ICLR 2018; Pan et al, ICCV 2019]

Combinational Normalization

- Exemplar Normalization

$$\widehat{X}_n = \sum_k \gamma^k (\lambda_n^k \frac{X_n - \mu^k}{\sqrt{(\delta^k)^2 + \epsilon}}) + \beta^k$$



- Representative Batch Normalization
  - During training: Mini-batch statistics+ instance statistics
  - During inference: Population statistics + instance statistics

[Zhang et al, CVPR 2020; Gao et al, CVPR 2021]

Combinational Normalization

- Batch-Instance Normalization

$$\mathbf{y} = \left(\rho \cdot \hat{\mathbf{x}}^{(B)} + (1-\rho) \cdot \hat{\mathbf{x}}^{(I)}\right) \cdot \gamma + \beta,$$

$$\rho \leftarrow \mathrm{clip}_{[0,1]}\left(\rho - \eta \Delta \rho\right)$$

- Adaptive Layer-Instance Normalization (AdaLIN)

$$AdaLIN(a,\gamma,\beta) = \gamma \cdot (\rho \cdot \hat{a_I} + (1-\rho) \cdot \hat{a_L}) + \beta,$$

$$\hat{a_I} = \frac{a-\mu_I}{\sqrt{\sigma_I^2+\epsilon}}, \hat{a_L} = \frac{a-\mu_L}{\sqrt{\sigma_L^2+\epsilon}},$$

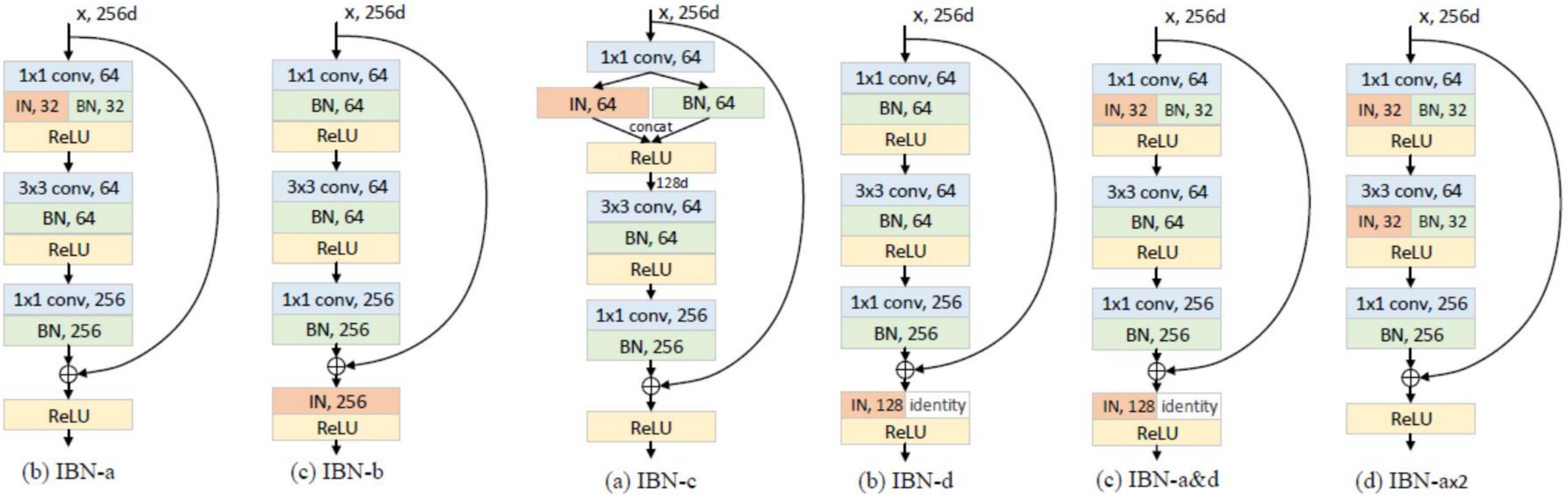
$$\rho \leftarrow clip_{[0,1]}(\rho - \tau\Delta\rho)$$

Generated by
a network

Attentive Normalization [Nam and Kim, NeurIPS 2018; Kim et al, ICLR 2020]

Combinational Normalization

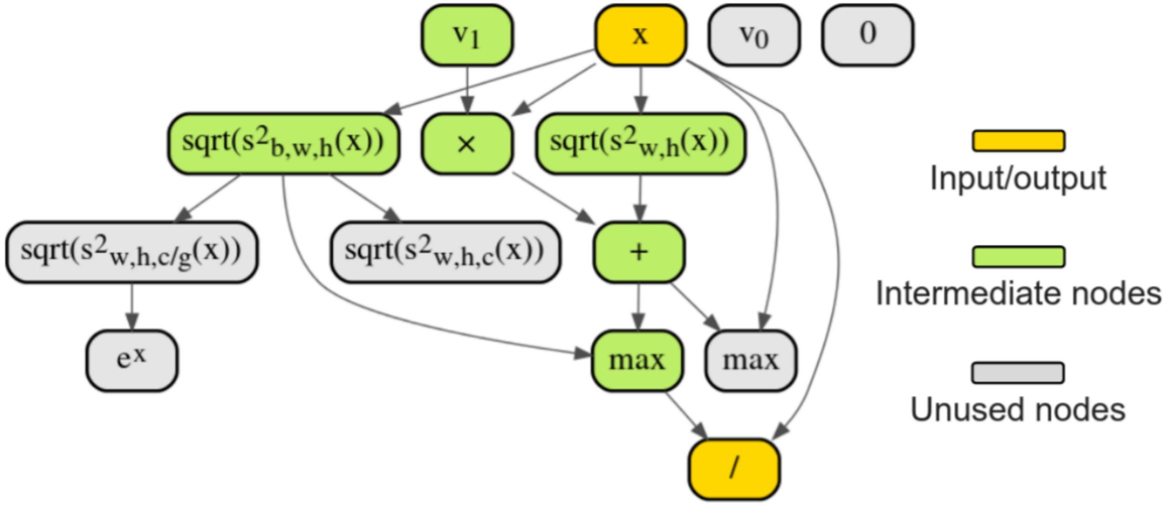
- Combination by design: IBN-Net



Two at Once: Enhancing Learning and Generalization Capacities via IBN-Net [Pan et al, ECCV 2018]

Normalization by Learning

- EvoNorm



BN-ReLU	$\max\left(\frac{x-\mu_{b,w,h}(x)}{\sqrt{s^2_{b,w,h}(x)}}\gamma+\beta, 0\right)$
EvoNorm-B0	$\frac{x}{\max\left(\sqrt{s^2_{b,w,h}(x)}, v_1 x+\sqrt{s^2_{w,h}(x)}\right)}\gamma+\beta$

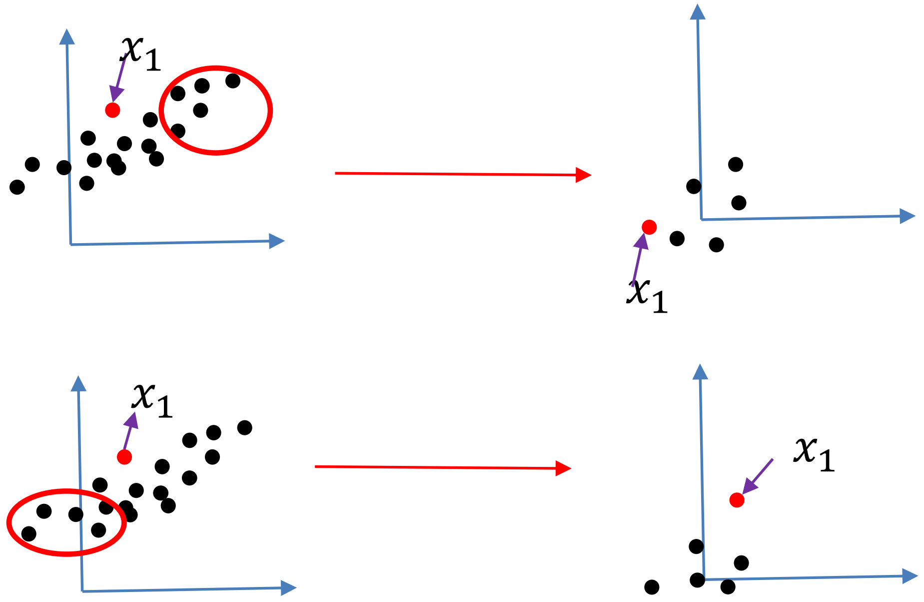
Evolving Normalization-Activation Layers[Liu et al, NeurIPS 2020]

Outline

- Normalizing activations as functions
  - A Framework for decomposing normalization
    - Normalization Area Partitioning (NAP)
    - Normalization Operation (NOP)
    - Normalization Representation Recovery (NRR)
  - Multi-Mode and combinational normalization
  - BN for more robust estimation

BN for More Robust Estimation

- Small batch size problem of BN
  - Large Training-test discrepancy
  - Large stochasticity during training



- Combining the population and mini-batch during training
  - [Dinh et al, ICLR 2016]

$$\begin{cases} \hat{u} = (1-\lambda)\hat{u} + \lambda u, \\ \hat{\sigma}^2 = (1-\lambda)\hat{\sigma}^2 + \lambda\sigma^2 \end{cases}$$

BN for More Robust Estimation

- Combining the population and mini-batch during training
  - Batch Re-normalization

$$\mu_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m} x_i$$

$$\sigma_{\mathcal{B}} \leftarrow \sqrt{\epsilon + \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu_{\mathcal{B}})^2}$$

$$r \leftarrow \texttt{stop\_gradient}\left(\text{clip}_{[1/r_{\max}, r_{\max}]}\left(\frac{\sigma_{\mathcal{B}}}{\sigma}\right)\right)$$

$$d \leftarrow \texttt{stop\_gradient}\left(\text{clip}_{[-d_{\max}, d_{\max}]}\left(\frac{\mu_{\mathcal{B}} - \mu}{\sigma}\right)\right)$$

$$\widehat{x}_i \leftarrow \frac{x_i - \mu_{\mathcal{B}}}{\sigma_{\mathcal{B}}} \cdot r + d$$

$$y_i \leftarrow \gamma\, \widehat{x}_i + \beta$$

Update moving averages

$$\mu := \mu + \alpha(\mu_{\mathcal{B}} - \mu)$$
$$\sigma := \sigma + \alpha(\sigma_{\mathcal{B}} - \sigma)$$

Batch Renormalization: Towards Reducing Minibatch Dependence in Batch-Normalized Models[Ioffe, NeurIPS 2017]

BN for More Robust Estimation

- Normalization as Functions Combining Population Statistics
  - Online normalization [Chiley et al, NeurIPS 2019]
  - Towards stabilizing batch statistics in backward propagation of batch normalization [Yan et al, ICLR 2020]
  - PowerNorm: rethinking batch normalization in transformers [Shen et al, ICML 2020]
  - Momentum batch normalization for deep learning with small batch size [Yong et al, ECCV 2020]
  - Double forward propagation for memorized batch normalization [Guo et al, AAAI 2018]
  - Cross-iteration batch normalization [Yao et al, CVPR 2021]

BN for More Robust Estimation

- Robust inference methods for BN
  - Estimating population statistics after training
    - Ioffe and Szegedy, 2015
    - Luo et al, 2018
    - Wu et al, 2021
  - Estimating batch normalization statistics for evaluation
    - EvalNorm [Singh et al, ICCV 2019]
    - [Summers and Dinneen, ICLR 2020]

Outline

Introduction of Normalization Methods

- Normalize Activation
  - By population statistics
  - As functions
- Normalize Weight
- Normalize Gradient

Normalizing Weights

- The general idea: normalize the activation implicitly during training

- Normalization Propagation [Arpit et al 2016; Shekhovtsov et al, 2018]
  - Normalize input: 0-mean and unit variance
  - Assuming **W** is orthogonal
  - Derivate the nonlinear dynamic, e.g. Relu:

Remark 1. *(Post-ReLU distribution) Let* $X \sim \mathcal{N}(0,1)$ *and* $Y = \max(0, X)$. *Then* $\mathbb{E}[Y] = \frac{1}{\sqrt{2\pi}}$ *and* $\mathrm{var}(Y) = \frac{1}{2}\left(1 - \frac{1}{\pi}\right)$

- Deriving the dynamics of activation by designing non-linearity [Shang et al, 2017; Klambauer et al, 2017]

Weight Normalization

- Target BN's drawback:
  - Unstable for small mini batch size
  - RNN

- Express weight as new parameters

$$\mathbf{w} = \frac{g}{||\mathbf{v}||}\mathbf{v} \qquad\qquad y = \phi(\mathbf{w} \cdot \mathbf{x} + b)$$

- Decouple direction and length of weight vectors

Weight Normalization: A Simple Reparameterization to Accelerate Training of Deep Neural Networks [Salimans et al, NeurIPS 2016]

Centered Weight Normalization

- Motivated by initialization methods: zero-mean, stable variance
  [Glorot et al, AISTATS 2010; He et al, ICCV 2015]

- Constrained optimization problem:

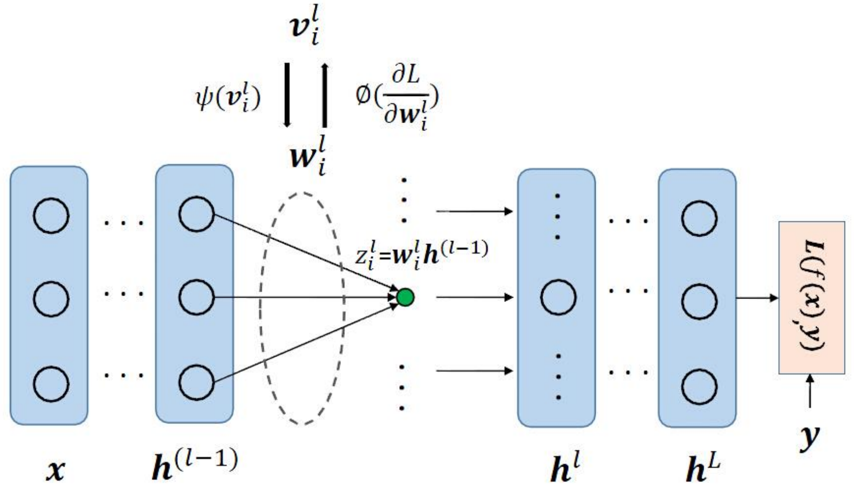
$$\theta^* = \arg\min_\theta \mathbb{E}_{(\mathbf{x},\mathbf{y})\in D}[\mathcal{L}(\mathbf{y}, f(\mathbf{x};\theta))]$$
$$s.t. \quad \mathbf{w}^T\mathbf{1} = 0 \; and \; \|\mathbf{w}\| = 1$$

- Solution by re-parameterization
  - Using proxy parameter v

$$\mathbf{w} = \frac{\mathbf{v} - \frac{1}{d}\mathbf{1}(\mathbf{1}^T\mathbf{v})}{\|\mathbf{v} - \frac{1}{d}\mathbf{1}(\mathbf{1}^T\mathbf{v})\|}$$

  - Gradient information:

$$\frac{\partial\mathcal{L}}{\partial\mathbf{v}} = \frac{1}{\|\hat{\mathbf{v}}\|}[\frac{\partial\mathcal{L}}{\partial\mathbf{w}} - (\frac{\partial\mathcal{L}}{\partial\mathbf{w}}\mathbf{w})\mathbf{w}^T - \frac{1}{d}(\frac{\partial\mathcal{L}}{\partial\mathbf{w}}\mathbf{1})\mathbf{1}^T]$$



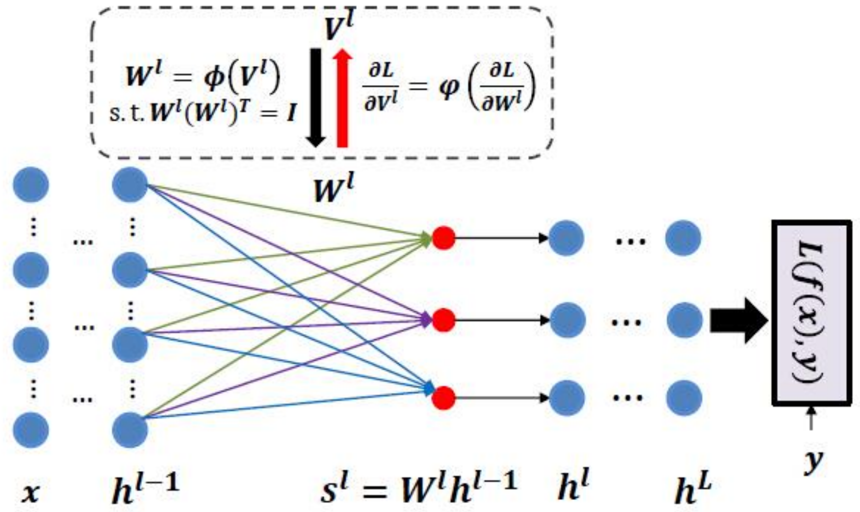
Centered Weight Normalization in Accelerating Training of Deep Neural Networks [Huang et al, ICCV 2017]

Orthogonal Weight Normalization

- Motivation: orthogonal initialization [Saxe et al, ICLR 2014; Mishkin et al, ICLR 2016]
  - Activation: $\| h \| = \| x \|$
  - Gradient: $\| \frac{\partial L}{\partial x} \| = \| \frac{\partial L}{\partial h} \|$
- Constrained optimization problem over multiple Stiefel manifold:

$$\theta^* = \arg\min_\theta \mathbb{E}_{(\mathbf{x},\mathbf{y})\in D}\left[\mathcal{L}\left(\mathbf{y}, f\left(\mathbf{x};\theta\right)\right)\right]$$
$$s.t. \quad \mathbf{W}^l \in \mathcal{O}_l^{n_l \times d_l}, l = 1, 2, ..., L$$

- Solution by re-parameterization



Orthogonal Weight Normalization: Solution to Optimization over Multiple Dependent Stiefel Manifolds in Deep Neural Networks [Huang et al, AAAI 2018]

Normalizing Weights

- Constraints with optimization:

$$\theta^* = \arg\min_\theta \mathbb{E}_{(\mathbf{x},\mathbf{y})\in D}[\mathcal{L}(\mathbf{y}, f(\mathbf{x};\theta))] \quad s.t. \quad \Upsilon(\boldsymbol{W}),$$

- Weight normalization:

$$\Upsilon(\boldsymbol{W}) = \{\|\boldsymbol{W}_i\| = 1, i = 1, ..., d_{out}\}$$

- Centered weight normalization/Scaled weight standardization:

$$\Upsilon(\boldsymbol{W}) = \{\boldsymbol{W}_i^T\mathbf{1} = 0 \,\&\, \|\boldsymbol{W}_i\| = 1, i = 1, ..., d_{out}\}$$

- Weight standardization:

$$\Upsilon(\boldsymbol{W}) = \{\boldsymbol{W}_i^T\mathbf{1} = 0 \,\&\, \|\boldsymbol{W}_i\| = \sqrt{d_{out}}, i = 1, ..., d_{out}\}$$

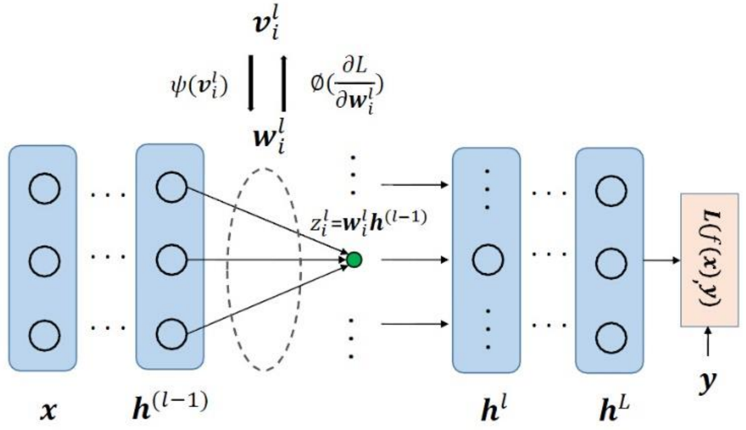
- Orthogonal weight normalization:

$$\Upsilon(\boldsymbol{W}) = \{\boldsymbol{W}\boldsymbol{W}^T = \boldsymbol{I}\}$$

[Salimans et al, NeurIPS 2016; Huang et al, ICCV 2017;Huang et al, AAAI 2018; Qiao et al, 2019; Brock et al, ICLR 2021]

Training with Constraints

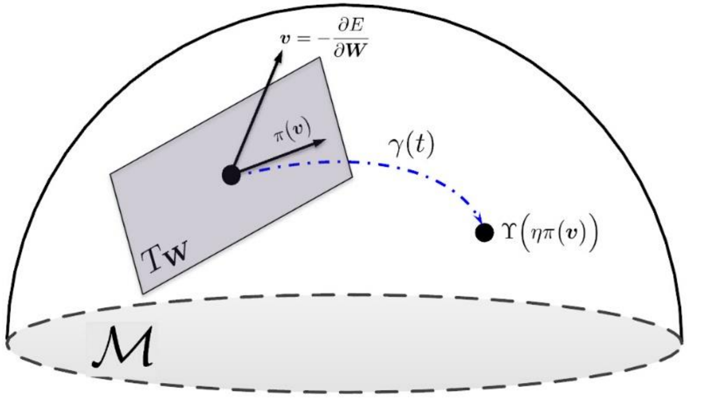
- Re-Parameterization



- Regularization with an extra penalty

$$\theta^* = \arg\min_\theta \mathbb{E}_{(\mathbf{x},\mathbf{y})\in D}\left[\mathcal{L}\left(\mathbf{y}, f\left(\mathbf{x};\theta\right)\right)\right] + \frac{\lambda}{2}\sum_{i=1}^{D} \|\mathbf{W}_l^T \mathbf{W}_l - \mathbf{I}\|_F^2$$

- Riemannian optimization



[Huang et al, ICCV 2017; Cisse et al, ICML 2017; Bansal et al; NeurIPS 2018; Ozay and Okatani, 2016]

Discussions with Normalizing Weights

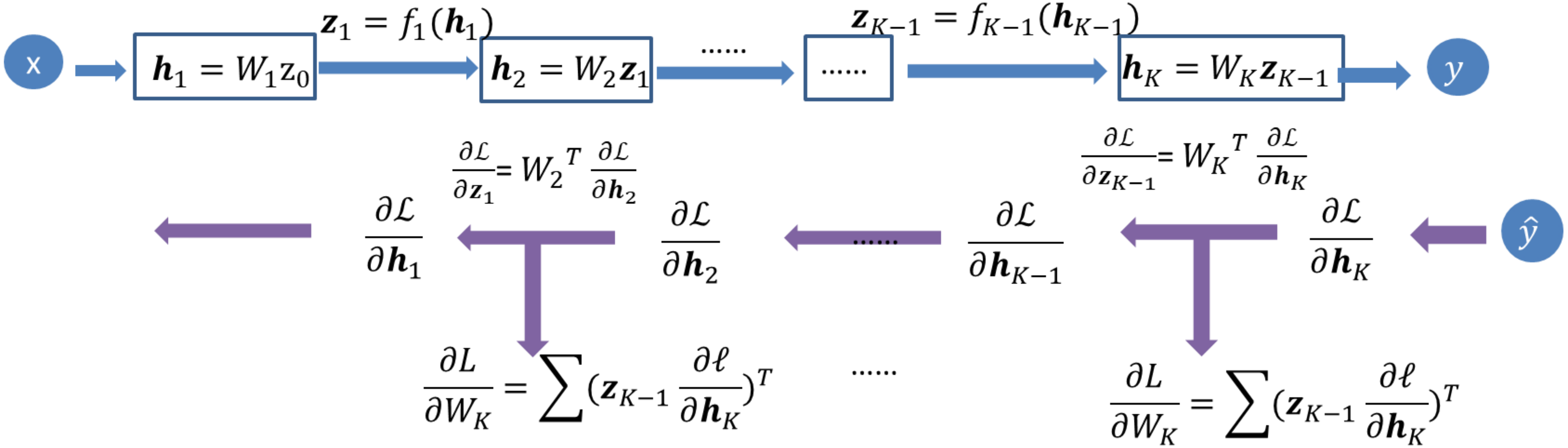
- Advantages over activation normalizations
  – No extra cost during inference
  – Not sensitive to the batch size, compared to BN

- Disadvantages over activation normalizations
  – It is not stable, compared to activation normalizations
  – It needs to well design the gain parameters for satisfying criteria 1 (equivalent variance/distribution among layers).
  – The gain parameters depend on the network architectures [Huang et al 2017, Brock et al, 2021], thus it is more difficult to use in practice

Outline

- Introduction of Normalization Methods
  - Normalize Activation
    - By population statistics
    - As functions
  - Normalize Weight
  - Normalize Gradient

Normalizing Gradients



- Block-wise (layer-wise) gradient normalization [Yu et al, 2017]
  - $\widehat{\frac{\partial L}{\partial W_K}} = \frac{\frac{\partial L}{\partial W_K}}{\|\frac{\partial L}{\partial W_K}\|_2}$
  - $\widehat{\frac{\partial L}{\partial W_K}} = \alpha \frac{\frac{\partial L}{\partial W_K}\|W_K\|_2}{\|\frac{\partial L}{\partial W_K}\|_2}$ (adaptive for scale-invariant network)
- Layer-wise Adaptive Rate Scaling (LARS) [You et al 2017], for large batch training

Normalizing Gradients

- LAMB [You et al, 2020]
  - LARS + Adam, for large-batch BERT training

- LANS [Zheng et al, 2020]
  - Incorporate Nesterov’s Momentum into LAMB, for large-batch BERT training

- Gradient centralization [Yong et al, 2020]
  - $\widehat{\frac{\partial L}{\partial W_K}} = (\mathrm{I} - \boldsymbol{e}\boldsymbol{e}^T)\frac{\partial L}{\partial W_K}$

Q&A