Towards Understanding Normalization in Deep Learning and its Applications

Ping Luo
The Chinese University of Hong Kong (CUHK)

VALSE Webinar, 2018-10-24

Outline

1. Whitened Neural Network (WNN) *(Desjardins etal. NIPS15)*
   - Relations between network design and optimization
   - Post-GWNN *(Luo ICML17)*
2. Batch Normalization (BN)
   - Regularization and Generalization *(Luo etal. arXiv:1809.00846)*
3. Switchable Normalization (SN) *(Luo etal. arXiv:1806.10779)*
4. More Techniques
   - Kalman Normalization (KN) *(Wang etal. NIPS18)*
   - Instance-Batch Normalization Network (IBN-Net) *(Pan etal. ECCV18)*
5. Discussions and Future Work

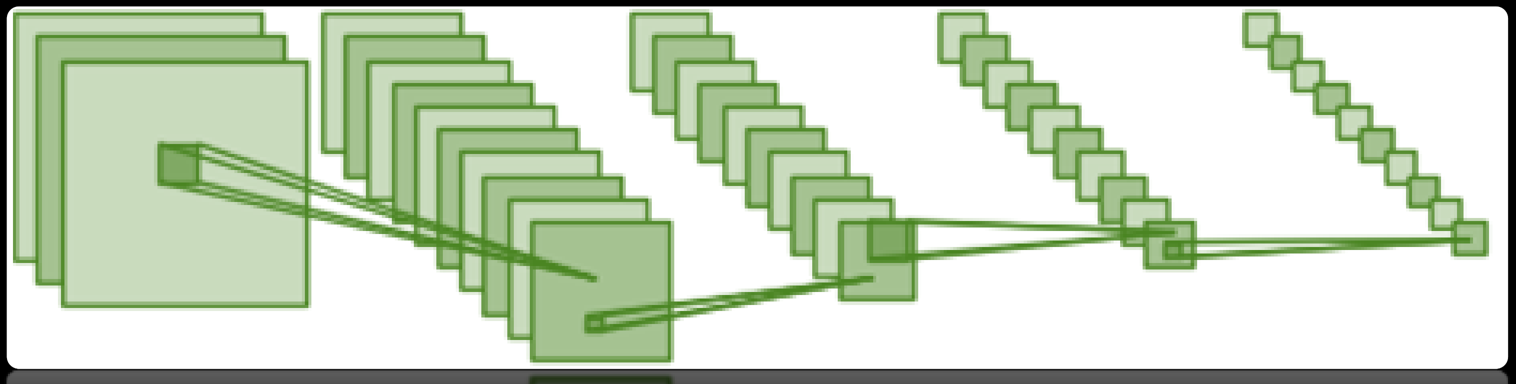
Whitened Neural Network (WNN)

PART 1

Relations between Network Design and Optimization

$$y = f(x;\theta)$$

Input: x



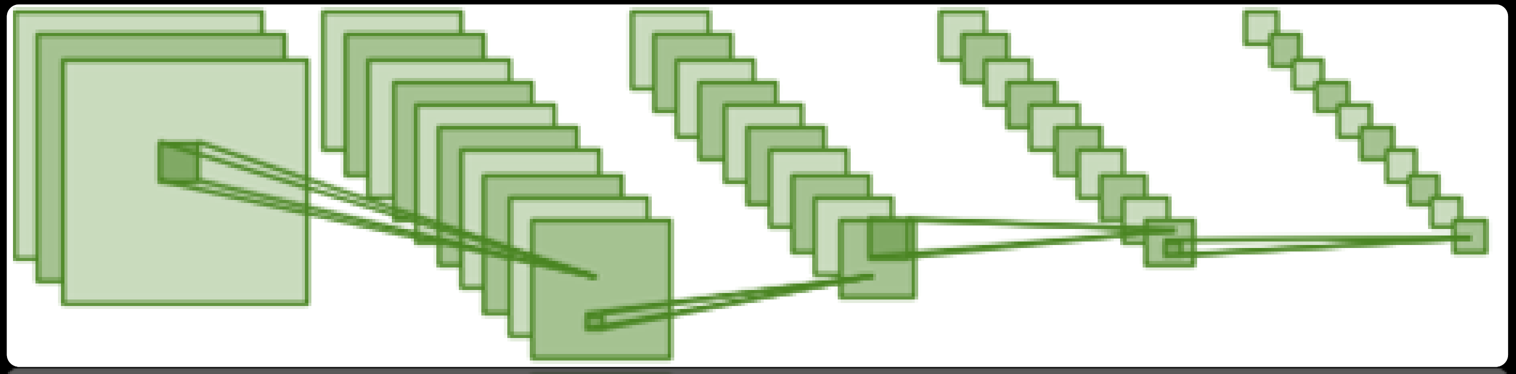
Label: $\hat{y}$

$$\theta^{t+1} = \theta^{t} - F^{-1}\nabla_{\theta}\mathcal{L}(\hat{y}, f(x;\theta))$$

Relations between Network Design and Optimization

$$\boxed{y = f(x;\theta)}$$

Input: x



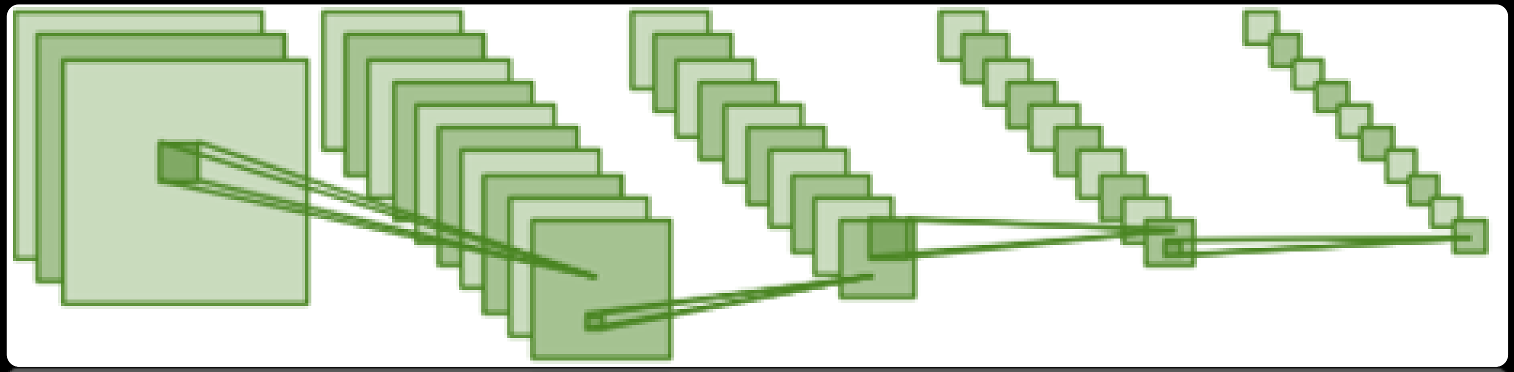
Label: $\hat{y}$

$$\theta^{t+1} = \theta^{t} - F^{-1}\nabla_{\theta}\mathcal{L}(\hat{y}, f(x;\theta))$$

Relations between Network Design and Optimization

$$y = f(x;\theta)$$

Input: x



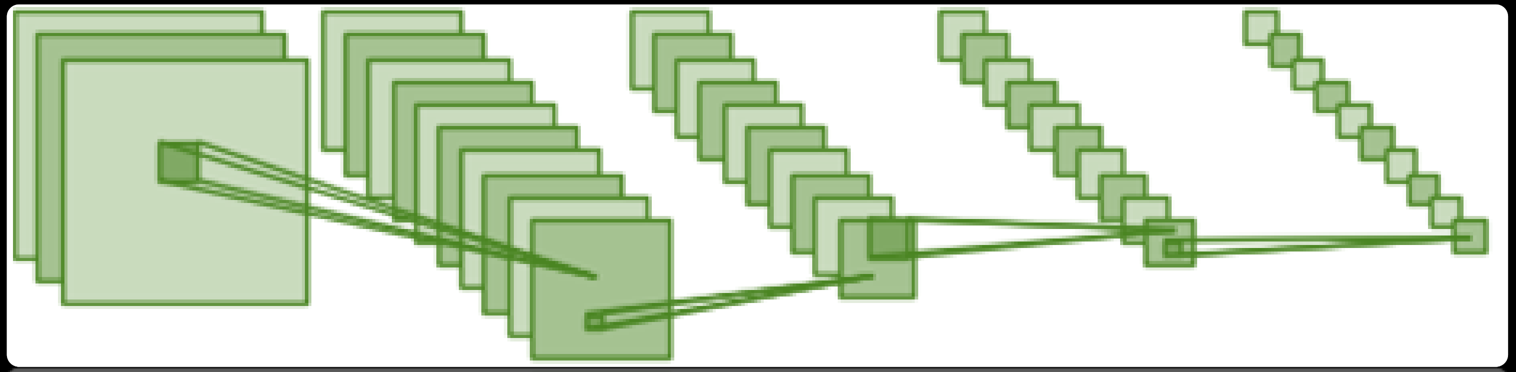
Label: $\hat{y}$

$$\theta^{t+1} = \theta^{t} - F^{-1}\nabla_{\theta}\mathcal{L}(\hat{y}, f(x;\theta))$$

Relations between Network Design and Optimization

$$y = f(x;\theta)$$

Input: x



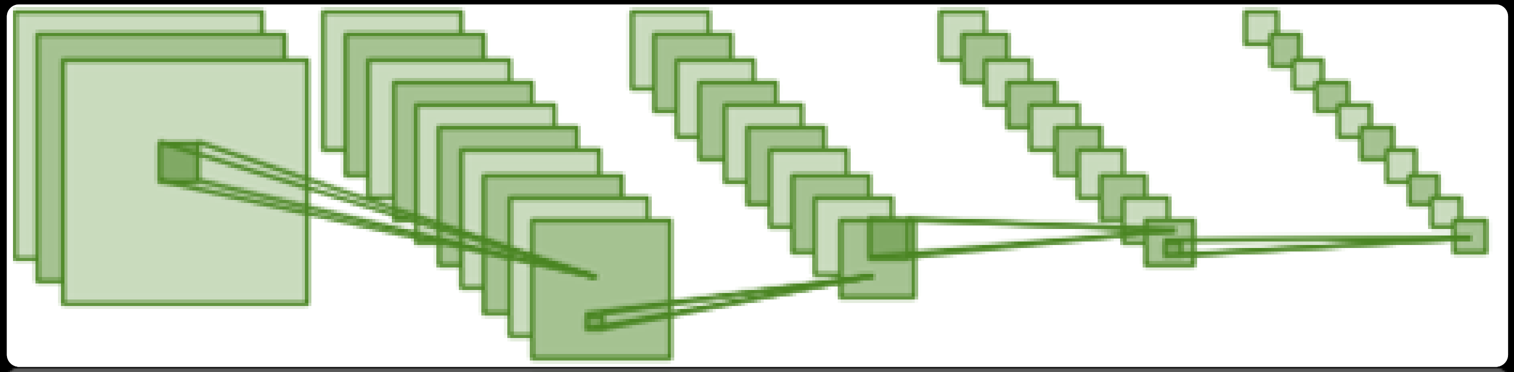
Label: $\hat{y}$

Natural Gradient Descent (NGD): $\theta^{t+1} = \theta^{t} - F^{-1}\boxed{\nabla_\theta \mathcal{L}(\hat{y}, f(x;\theta))}$

Relations between Network Design and Optimization

$$y = f(x;\theta)$$

Input: x



Label: $\hat{y}$

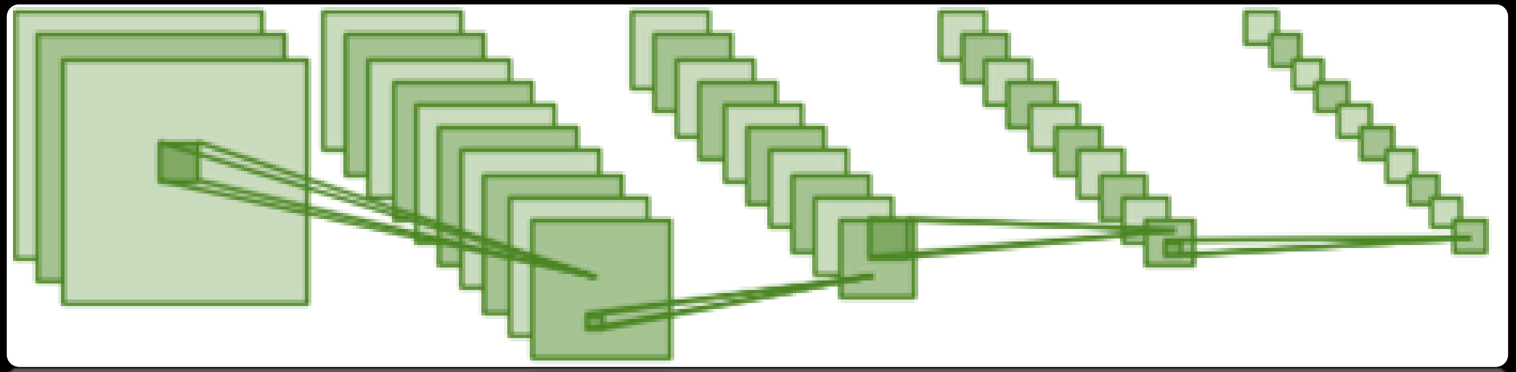
Natural Gradient Descent (NGD): $\theta^{t+1} = \theta^{t} - \boxed{F^{-1}} \nabla_\theta \mathcal{L}(\hat{y}, f(x;\theta))$

F: Fisher Information Matrix (FIM)

Relations between Network Design and Optimization

$$y = f(x;\theta)$$

Input: x



Label: $\hat{y}$

Natural Gradient Descent (NGD): $\theta^{t+1} = \theta^{t} - F^{-1}\nabla_{\theta}\mathcal{L}(\hat{y}, f(x;\theta))$

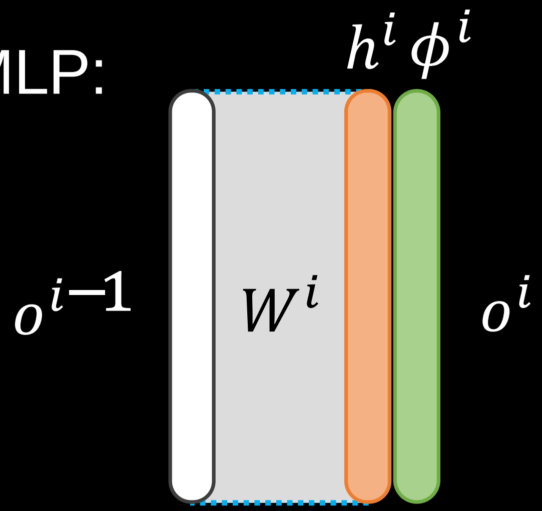
$\boxed{\text{SGD: } F = I}$ works well.

Relations between Network Design and Optimization

Deep Learning turns Optimization Problems into Feed-Forward Network Design.

Multilayer Perceptron (MLP)

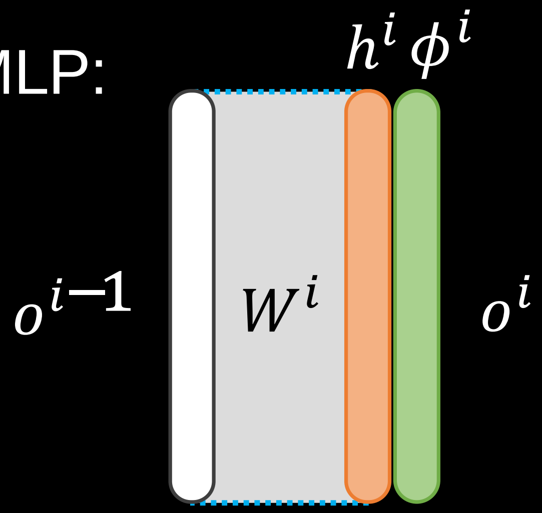
MLP:



Forward：

Multilayer Perceptron (MLP)

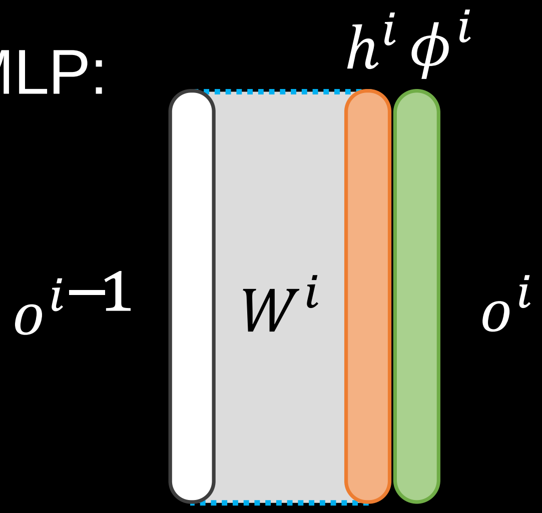
MLP:



Forward: $h^i = W^i o^{i-1}$ (linear transformation)

Multilayer Perceptron (MLP)

MLP:



layer input

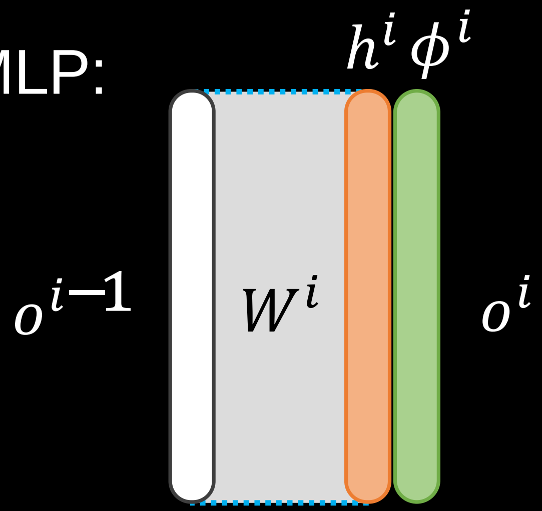
Forward: $h^i = W^i o^{i-1}$ (linear transformation)

hidden feature

network parameter

Multilayer Perceptron (MLP)

MLP:

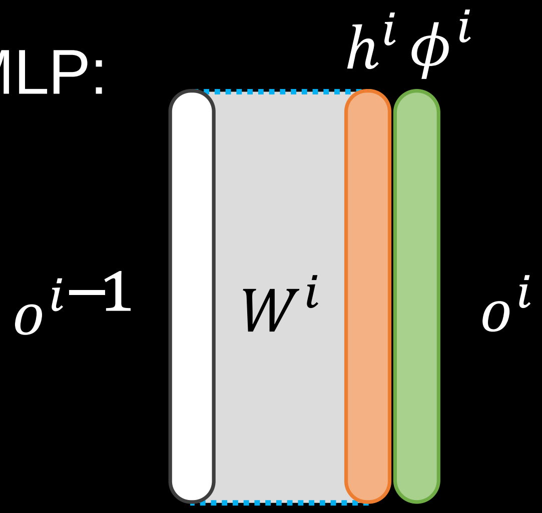


Forward: $h^i = W^i o^{i-1}$ (linear transformation)

$$\phi^i = \frac{h^i - \mathrm{E}[h^i]}{\sqrt{\mathrm{Var}(h^i)}}$$ (batch normalization)

Multilayer Perceptron (MLP)

MLP:



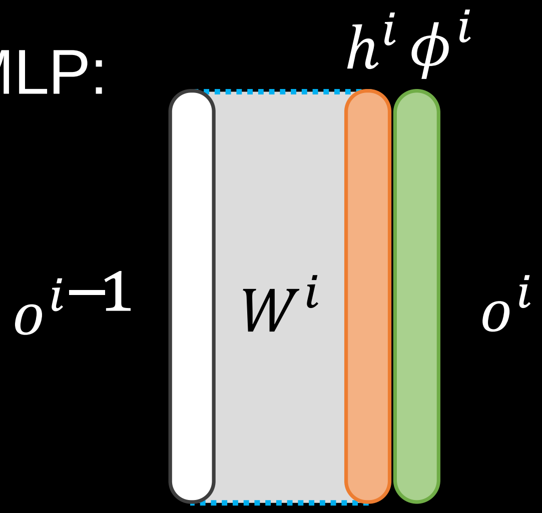
Forward： $h^i = W^i o^{i-1}$ (linear transformation)

$$\phi^i = \frac{h^i - \mathrm{E}[h^i]}{\sqrt{\mathrm{Var}(h^i)}}$$ (batch normalization)

$$o^i = \max(0, \alpha\phi^i + \beta)$$ (nonlinear activation)

Multilayer Perceptron (MLP)

MLP:



Forward： $h^i = W^i o^{i-1}$ (linear transformation)

$$\phi^i = \frac{h^i - \mathrm{E}[h^i]}{\sqrt{\mathrm{Var}(h^i)}}$$ (batch normalization)

$$o^i = \max(0, \alpha\phi^i + \beta)$$ (nonlinear activation)

- Natural gradient descent (NGD) with Fisher information matrix (FIM) F, $W_{t+1}^i = W_t^i - \lambda_t F_t^{-1} \nabla W_t^i$.

WNN is NGD

A DNN with l layers.

$$F = \left[\begin{array}{c:c:c:c} F_{11} & & & F_{ij} \\ \hdashline & F_{22} & & \\ \hdashline & & \ddots & \\ \hdashline F_{ij} & & & F_{ll} \end{array}\right]$$

- Natural gradient with Fisher information matrix (FIM) F is defined as $W_{t+1}^{i} = W_{t}^{i} - \lambda_t F_t^{-1} \nabla W_t^i$.

WNN is NGD

A DNN with l layers.

$$F = \left[\begin{array}{c:c:c:c} F_{11} & & & F_{ij} \\ \hdashline & F_{22} & & \\ \hdashline & & \ddots & \\ \hdashline F_{ij} & & & F_{ll} \end{array}\right] \Longleftrightarrow \text{Design network whose } F \approx I$$

- Natural gradient with Fisher information matrix (FIM) F is defined as $W_{t+1}^{i} = W_t^i - \lambda_t F_t^{-1} \nabla W_t^i$.

WNN is NGD

Covariance of gradients

$$F_{ij} = \mathbb{E}[\mathrm{vec}(\nabla W^i)\mathrm{vec}(\nabla W^j)^T]$$

A DNN with l layers.

$$F = \left[\begin{array}{c:c:c:c} F_{11} & & & F_{ij} \\ \hdashline & F_{22} & & \\ \hdashline & & \ddots & \\ \hdashline F_{ij} & & & F_{ll} \end{array}\right]$$

- Natural gradient with Fisher information matrix (FIM) F is defined as $W_{t+1}^i = W_t^i - \lambda_t F_t^{-1} \nabla W_t^i$.

WNN is NGD

vectorization   gradient: $\nabla W^{i} = o^{i-1} \Delta h^{i^{T}}$

$$F_{ij} = \mathbb{E}[\mathrm{vec}(\nabla W^{i})\mathrm{vec}(\nabla W^{j})^{T}]$$

A DNN with l layers.

$$F = \begin{bmatrix} F_{11} & & & F_{ij} \\ & F_{22} & & \\ & & \ddots & \\ F_{ij} & & & F_{ll} \end{bmatrix}$$

- Natural gradient with Fisher information matrix (FIM) F is defined as $W_{t+1}^{i} = W_{t}^{i} - \lambda_t F_t^{-1} \nabla W_t^{i}$.

WNN is NGD

vectorization gradient: $\nabla W^i = o^{i-1} {\Delta h^i}^T$

$$F_{ij} = \mathbb{E}[\mathrm{vec}(\nabla W^i)\mathrm{vec}(\nabla W^j)^T]$$

$$= \mathbb{E}[(\Delta h^i \otimes o^{i-1})(\Delta h^j \otimes o^{j-1})]$$

A DNN with l layers.

$$F = \begin{bmatrix} F_{11} & & & F_{ij} \\ & F_{22} & & \\ & & \ddots & \\ F_{ij} & & & F_{ll} \end{bmatrix}$$

- Natural gradient with Fisher information matrix (FIM) F is defined as $W_{t+1}^i = W_t^i - \lambda_t F_t^{-1} \nabla W_t^i$.

WNN is NGD

vectorization gradient: $\nabla W^i = o^{i-1} {\Delta h^i}^T$

Kronecker product

$$F_{ij} = \mathbb{E}[\mathrm{vec}(\nabla W^i)\mathrm{vec}(\nabla W^j)^T]$$
$$= \mathbb{E}[(\Delta h^i \otimes o^{i-1})(\Delta h^j \otimes o^{j-1})]$$

A DNN with l layers.

$$F = \begin{bmatrix} F_{11} & & & F_{ij} \\ & F_{22} & & \\ & & \ddots & \\ F_{ij} & & & F_{ll} \end{bmatrix}$$

- Natural gradient with Fisher information matrix (FIM) F is defined as $W_{t+1}^i = W_t^i - \lambda_t F_t^{-1} \nabla W_t^i$.

WNN is NGD

A DNN with l layers.

$$F = \left[\begin{array}{c:c:c:c} F_{11} & & & F_{ij} \\ \hdashline & F_{22} & & \\ \hdashline & & \ddots & \\ \hdashline F_{ij} & & & F_{ll} \end{array}\right]$$

vectorization   gradient: $\nabla W^i = o^{i-1} {\Delta h^i}^T$

Kronecker product

$$F_{ij} = \mathbb{E}[\mathrm{vec}(\nabla W^i)\mathrm{vec}(\nabla W^j)^T]$$

$$= \mathbb{E}[(\Delta h^i \otimes o^{i-1})(\Delta h^j \otimes o^{j-1})]$$

(mixed-product property)

$$= \mathbb{E}[\Delta h^i (\Delta h^j)^T \otimes o^{i-1}(o^{j-1})^T]$$

- Natural gradient with Fisher information matrix (FIM) F is defined as $W_{t+1}^i = W_t^i - \lambda_t F_t^{-1} \nabla W_t^i$.

WNN is NGD

A DNN with l layers.

$$F = \begin{bmatrix} F_{11} & & & F_{ij} \\ & F_{22} & & \\ & & \ddots & \\ F_{ij} & & & F_{ll} \end{bmatrix}$$

vectorization

gradient: $\nabla W^i = o^{i-1} {\Delta h^i}^T$

Kronecker product

$$F_{ij} = \mathbb{E}[\mathrm{vec}(\nabla W^i)\mathrm{vec}(\nabla W^j)^T]$$

$$= \mathbb{E}[(\Delta h^i \otimes o^{i-1})(\Delta h^j \otimes o^{j-1})]$$

(mixed-product property) $$= \mathbb{E}[\Delta h^i (\Delta h^j)^T \otimes o^{i-1}(o^{j-1})^T]$$

(independence) $$\approx \mathbb{E}[\Delta h^i (\Delta h^j)^T] \otimes \mathbb{E}[o^{i-1}(o^{j-1})^T]$$

- Natural gradient with Fisher information matrix (FIM) F is defined as $W_{t+1}^i = W_t^i - \lambda_t F_t^{-1} \nabla W_t^i$.

WNN is NGD

A DNN with l layers.

$$F = \left[\begin{array}{c:c:c:c} F_{11} & & & F_{ij} \\ \hdashline & F_{22} & & \\ \hdashline & & \ddots & \\ \hdashline F_{ij} & & & F_{ll} \end{array}\right]$$

vectorization gradient: $\nabla W^i = o^{i-1} {\Delta h^i}^T$

Kronecker product

$$F_{ij} = \mathbb{E}[\mathrm{vec}(\nabla W^i)\mathrm{vec}(\nabla W^j)^T]$$

$$= \mathbb{E}[(\Delta h^i \otimes o^{i-1})(\Delta h^j \otimes o^{j-1})]$$

(mixed-product property) $= \mathbb{E}[\Delta h^i (\Delta h^j)^T \otimes o^{i-1}(o^{j-1})^T]$

(independence) $\approx \boxed{\mathbb{E}[\Delta h^i (\Delta h^j)^T]} \otimes \boxed{\mathbb{E}[o^{i-1}(o^{j-1})^T]}$

- Natural gradient with Fisher information matrix (FIM) F is defined as $W^i_{t+1} = W^i_t - \lambda_t F_t^{-1} \nabla W^i_t$.

WNN is NGD

A DNN with l layers.

$$F = \left[\begin{array}{c:c:c:c} F_{11} & & & F_{ij} \\ \hdashline & F_{22} & & \\ \hdashline & & \ddots & \\ \hdashline F_{ij} & & & F_{ll} \end{array}\right]$$

vectorization

gradient: $\nabla W^i = o^{i-1} {\Delta h^i}^T$

Kronecker product

$$F_{ij} = \mathbb{E}[\mathrm{vec}(\nabla W^i)\mathrm{vec}(\nabla W^j)^T]$$

$$= \mathbb{E}[(\Delta h^i \otimes o^{i-1})(\Delta h^j \otimes o^{j-1})]$$

(mixed-product property) $= \mathbb{E}[\Delta h^i(\Delta h^j)^T \otimes o^{i-1}(o^{j-1})^T]$

(independence) $\approx \mathbb{E}[\Delta h^i(\Delta h^j)^T] \otimes \mathbb{E}[o^{i-1}(o^{j-1})^T]$

- Goal: $F_{ii} \approx I$.

WNN is NGD

A DNN with l layers. $\approx I$

$$F = \begin{bmatrix} F_{11} & & & F_{ij} \\ & F_{22} & & \\ & & \ddots & \\ F_{ij} & & & F_{ll} \end{bmatrix}$$

$$\begin{aligned} F_{ij} &= \mathbb{E}[\mathrm{vec}(\nabla W^{i})\mathrm{vec}(\nabla W^{j})^{T}] \\ &= \mathbb{E}[(\Delta h^{i} \otimes o^{i-1})(\Delta h^{j} \otimes o^{j-1})] \\ &= \mathbb{E}[\Delta h^{i}(\Delta h^{j})^{T} \otimes o^{i-1}(o^{j-1})^{T}] \\ &\approx \mathbb{E}[\Delta h^{i}(\Delta h^{j})^{T}] \otimes \mathbb{E}[o^{i-1}(o^{j-1})^{T}] \end{aligned}$$

$$\boxed{F_{ii} = \mathbb{E}[\Delta h^{i}(\Delta h^{i})^{T}] \otimes \mathbb{E}[o^{i-1}(o^{i-1})^{T}]}$$

WNN is NGD

A DNN with l layers.

$$F = \begin{bmatrix} F_{11} & & & F_{ij} \\ & F_{22} & & \\ & & \ddots & \\ F_{ij} & & & F_{ll} \end{bmatrix}$$

$$\begin{aligned} F_{ij} &= \mathbb{E}[\mathrm{vec}(\nabla W^{i})\mathrm{vec}(\nabla W^{j})^{T}] \\ &= \mathbb{E}[(\Delta h^{i} \otimes o^{i-1})(\Delta h^{j} \otimes o^{j-1})] \\ &= \mathbb{E}[\Delta h^{i}(\Delta h^{j})^{T} \otimes o^{i-1}(o^{j-1})^{T}] \\ &\approx \mathbb{E}[\Delta h^{i}(\Delta h^{j})^{T}] \otimes \mathbb{E}[o^{i-1}(o^{j-1})^{T}] \end{aligned}$$

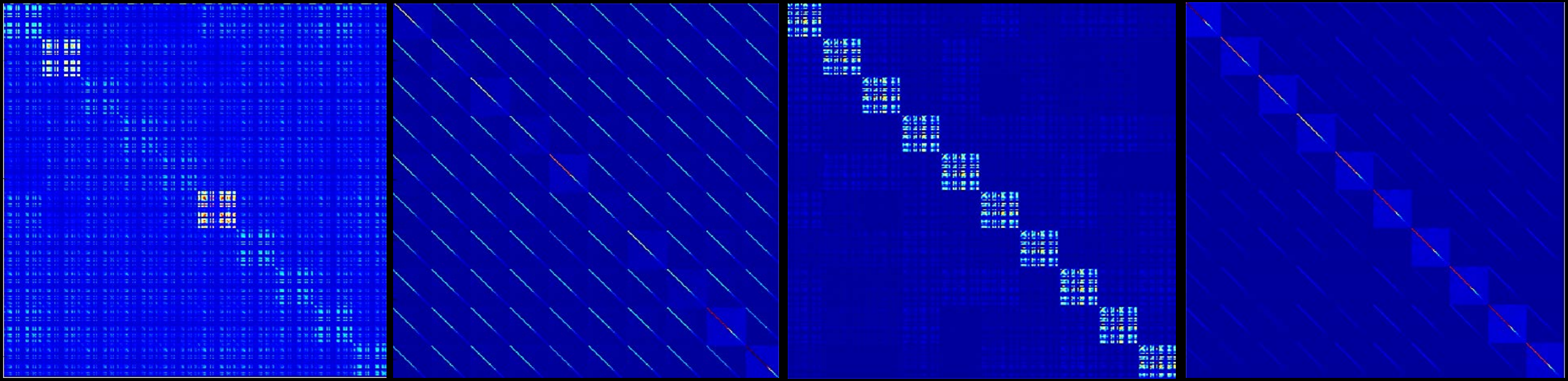
$$F_{ii} = \boxed{\mathbb{E}[\Delta h^{i}(\Delta h^{i})^{T}]} \otimes \boxed{\mathbb{E}[o^{i-1}(o^{i-1})^{T}]}$$

EigenNet, IJCAI17

WNN, NIPS15
GWNN, ICML17

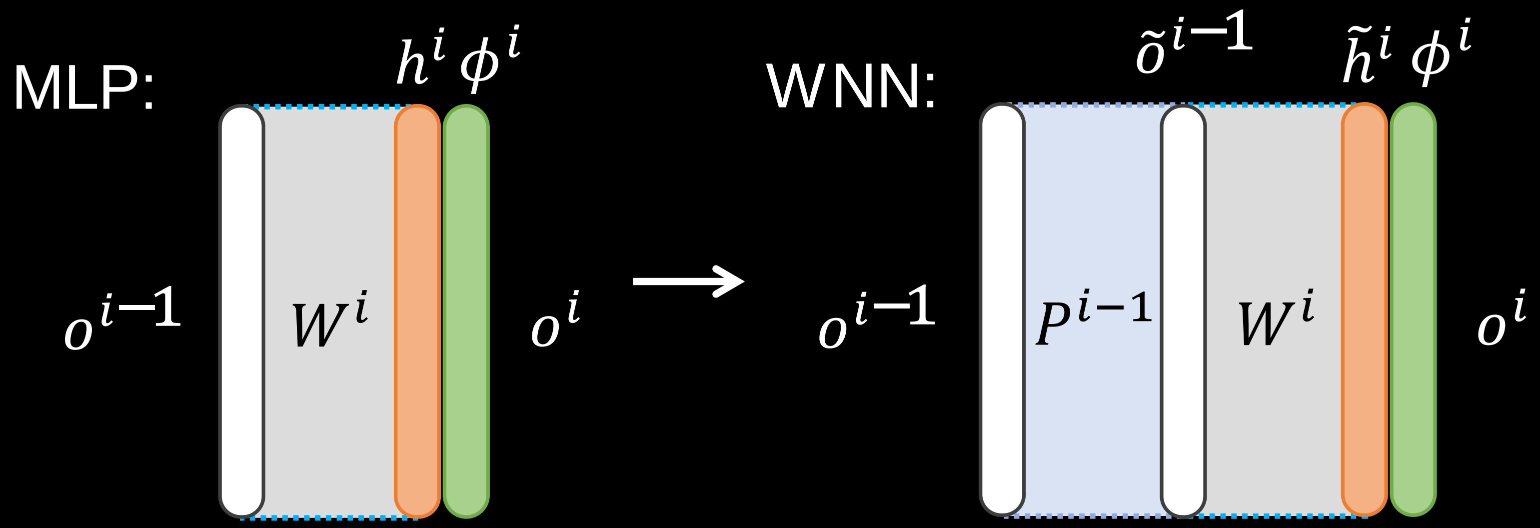
WNN is NGD

1) BN 2) Whiten forward 3) Whiten backward 4) Whiten both

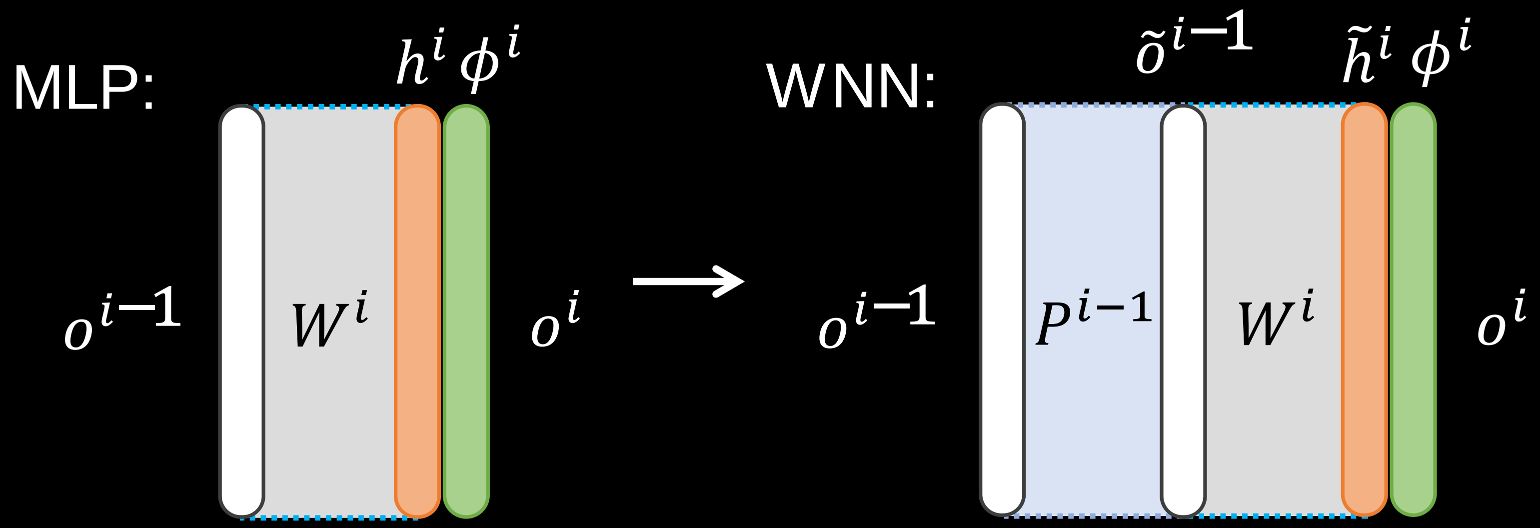


- Approximate $F_{ii} = \mathbb{E}[\Delta h^i(\Delta h^i)^T] \otimes \mathbb{E}[o^{i-1}(o^{i-1})^T] \approx I.$

Whitened Neural Network (WNN)

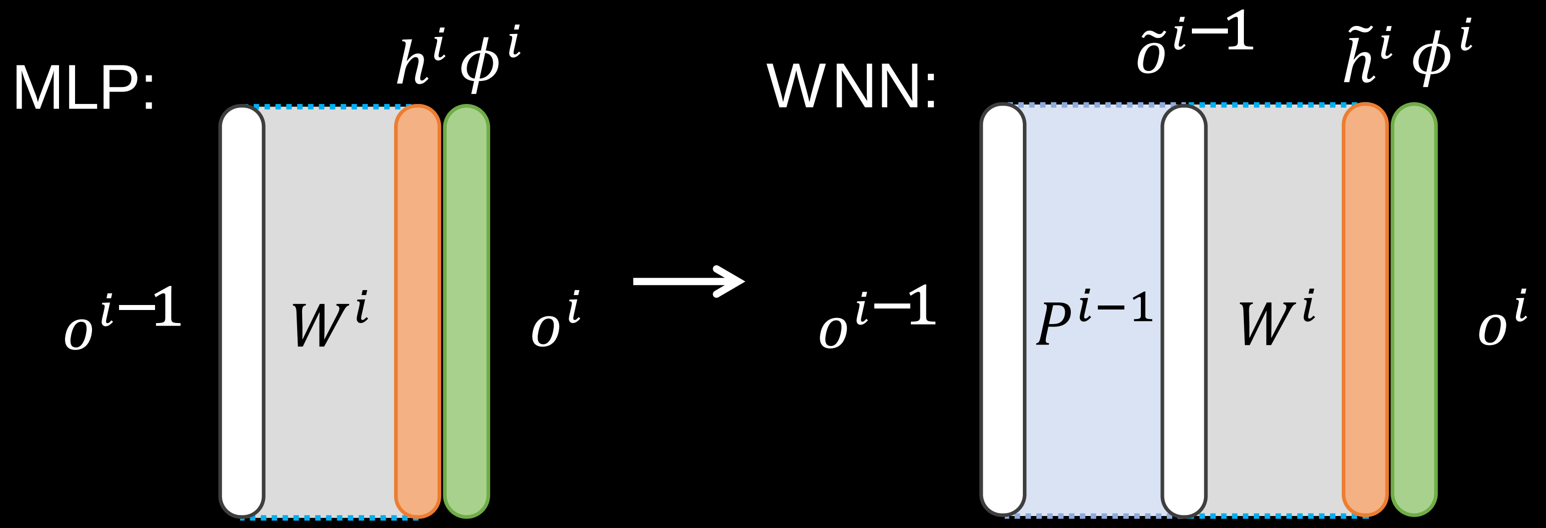


Whitened Neural Network (WNN)



$$\boxed{\min_{\{W^i\}} L}$$

Whitened Neural Network (WNN)

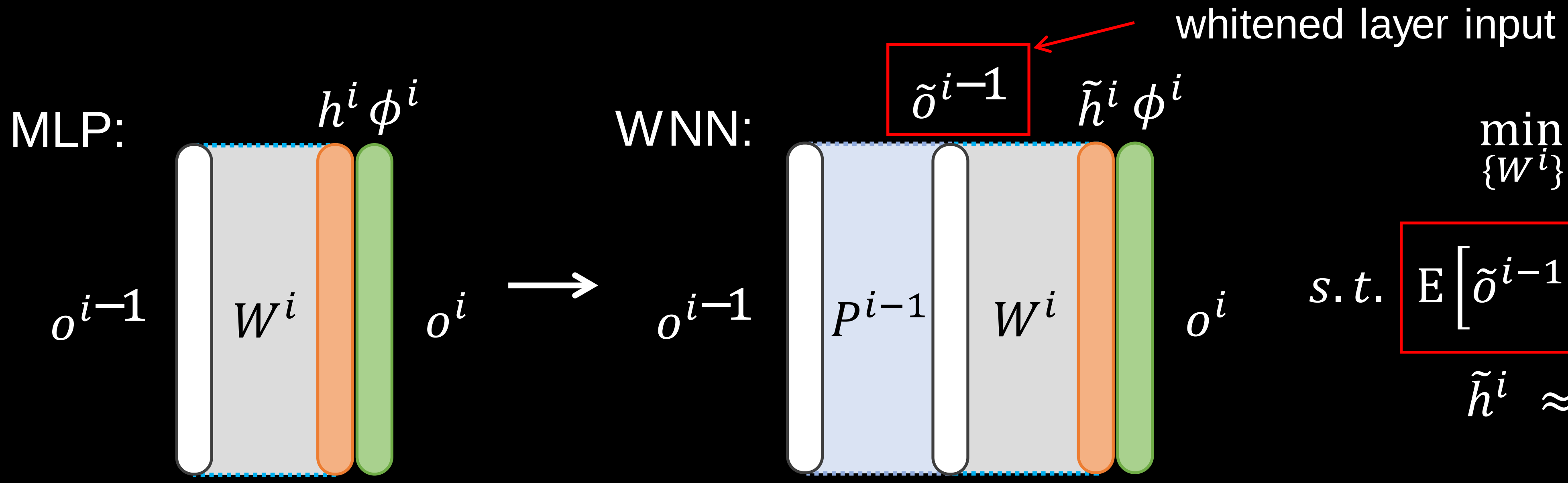


$$\min_{\{W^i\}} L$$

$$s.t.\ \mathrm{E}\left[\tilde{o}^{i-1}{\tilde{o}^{i-1}}^T\right] = I,$$

$$\tilde{h}^i \approx h^i.$$

Whitened Neural Network (WNN)

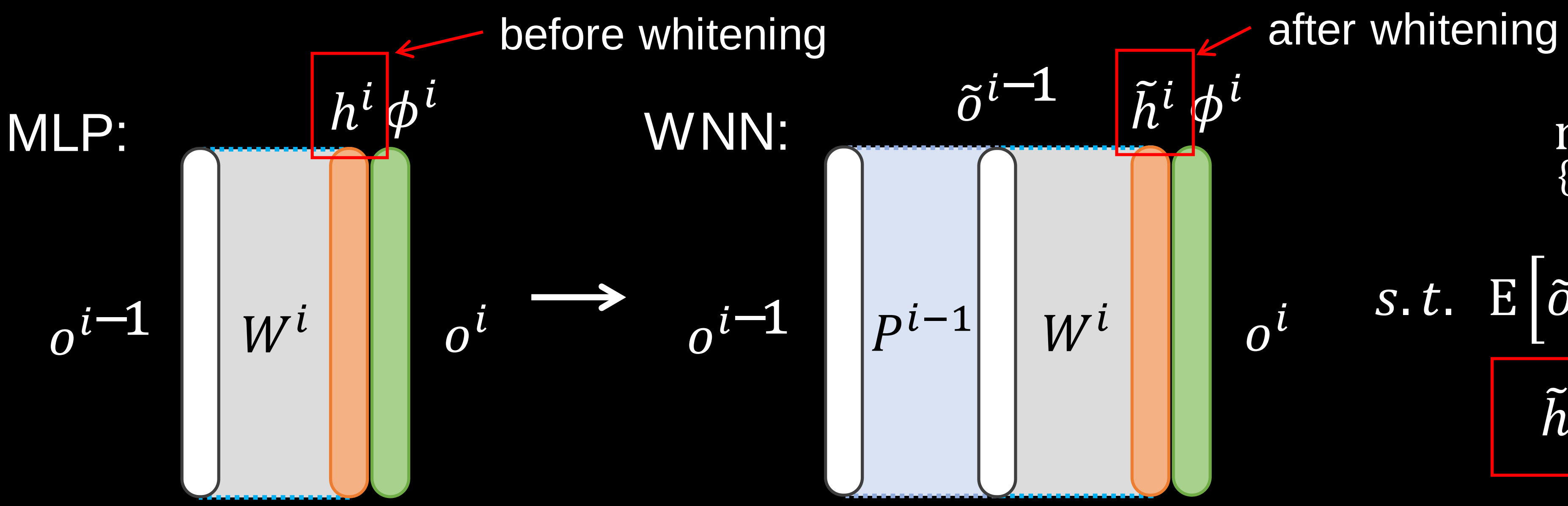


$$\min_{\{W^i\}} L$$

$$s.t.\ \boxed{\mathrm{E}\left[\tilde{o}^{i-1}{\tilde{o}^{i-1}}^{T}\right] = I,}$$

$$\tilde{h}^i \approx h^i\,.$$

Whitened Neural Network (WNN)



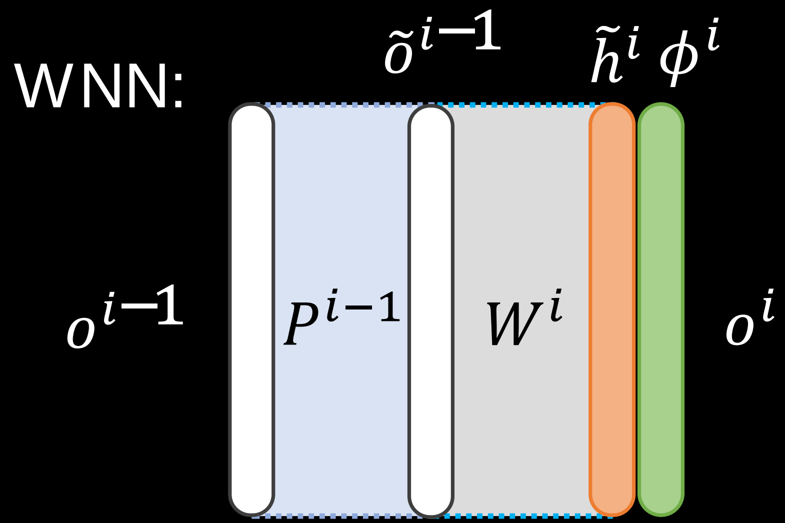
$$\min_{\{W^i\}} L$$

$$s.t.\ \ \mathrm{E}\left[\tilde{o}^{i-1}\tilde{o}^{i-1^T}\right] = I,$$

$$\boxed{\tilde{h}^i \approx h^i\,.}$$

WNN smooth solution space while maintains representation capacity.

Whitened Neural Networks (WNN)

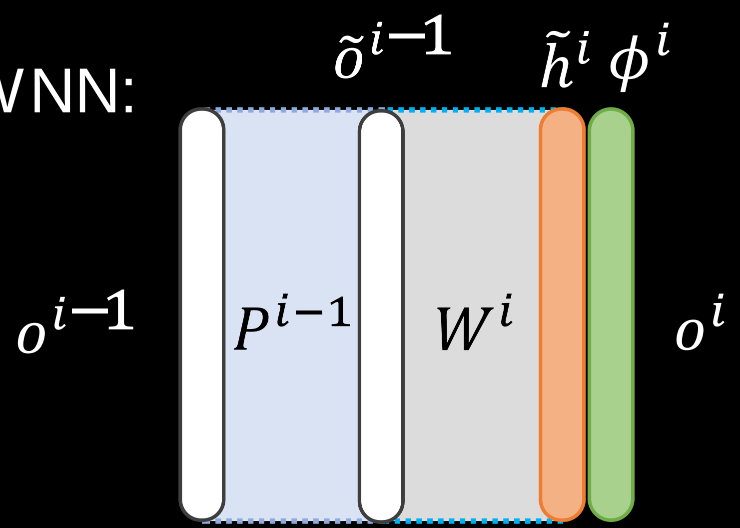


Forward: $\tilde{o}^{i-1} = P^{i-1}(o^{i-1} - \mu^{i-1}), \mathrm{E}\left[\tilde{o}^{i-1}{\tilde{o}^{i-1}}^T\right] = I$

(whitening)

Whitened Neural Networks (WNN)

WNN:



Forward: $\tilde{o}^{i-1} = P^{i-1}(o^{i-1} - \mu^{i-1}), \mathrm{E}\left[\tilde{o}^{i-1}\tilde{o}^{i-1^T}\right] = I$

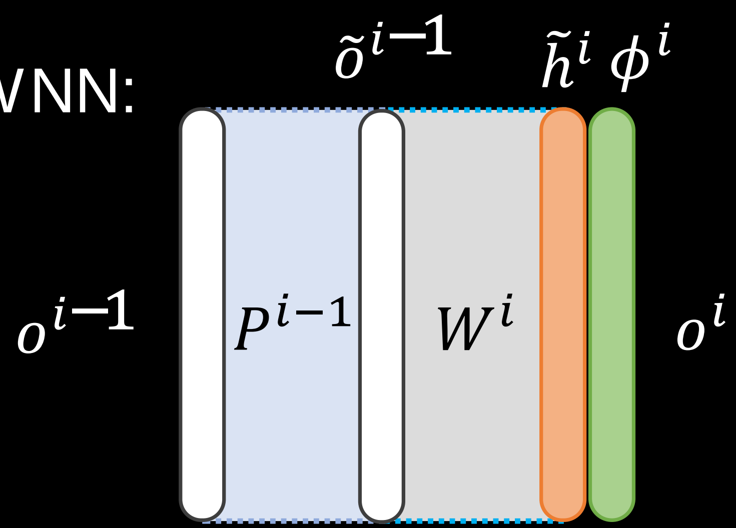
(whitening)

$\tilde{h}^{i} = W^{i}\tilde{o}^{i-1}$

(linear transformation)

Whitened Neural Networks (WNN)

WNN:



Forward: $\tilde{o}^{i-1} = P^{i-1}(o^{i-1} - \mu^{i-1}), \mathrm{E}\left[\tilde{o}^{i-1}{\tilde{o}^{i-1}}^{T}\right] = I$ (whitening)

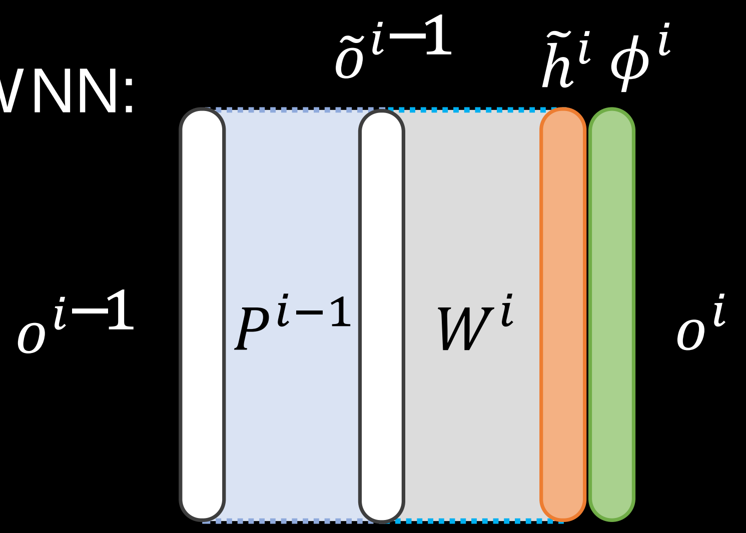
$$\tilde{h}^{i} = W^{i}\tilde{o}^{i-1}$$ (linear transformation)

$$\phi^{i} = \frac{\tilde{h}^{i}}{\sqrt{\mathrm{Var}(\tilde{h}^{i})}}$$ (batch normalization)

(nonlinear activation)

Whitened Neural Networks (WNN)

WNN:



Forward: $\tilde{o}^{i-1} = P^{i-1}(o^{i-1} - \mu^{i-1}), \mathrm{E}\left[\tilde{o}^{i-1}\tilde{o}^{i-1^T}\right] = I$ (whitening)

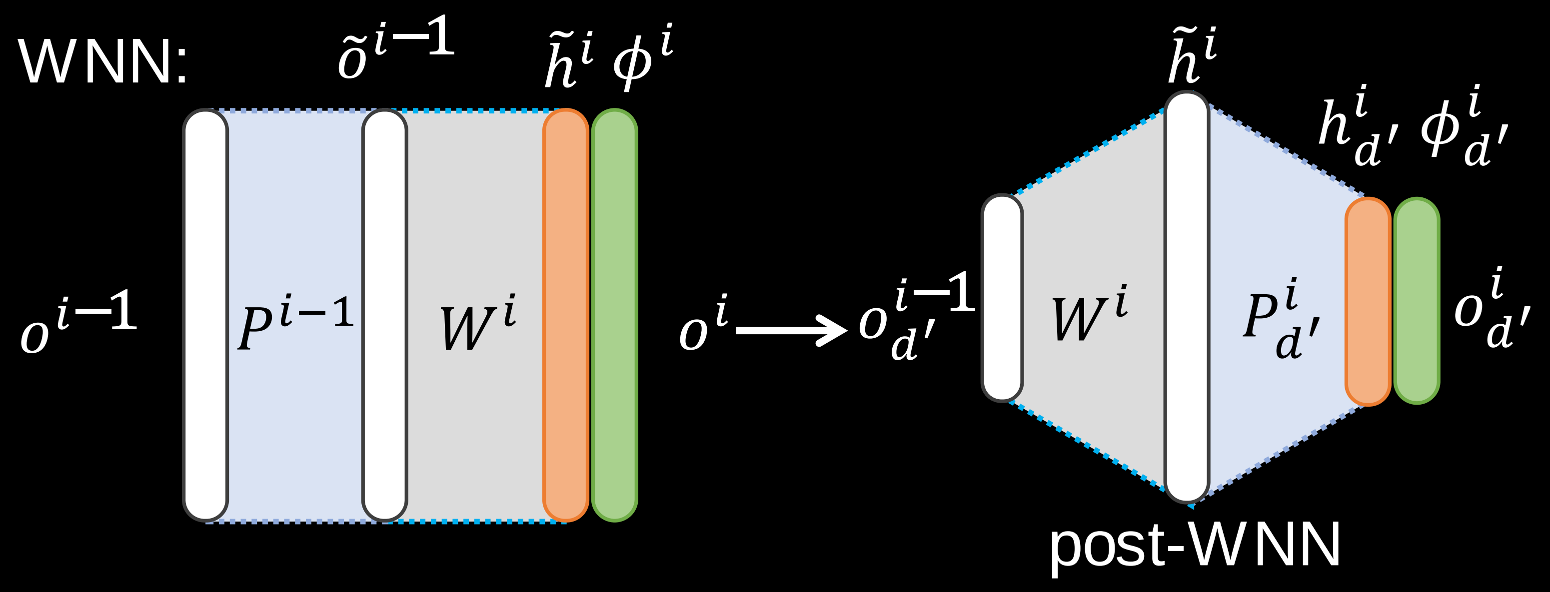
$\tilde{h}^i = W^i \tilde{o}^{i-1}$ (linear transformation)

$\phi^i = \frac{\tilde{h}^i}{\sqrt{\mathrm{Var}(\tilde{h}^i)}}$ (batch normalization)

(nonlinear activation)

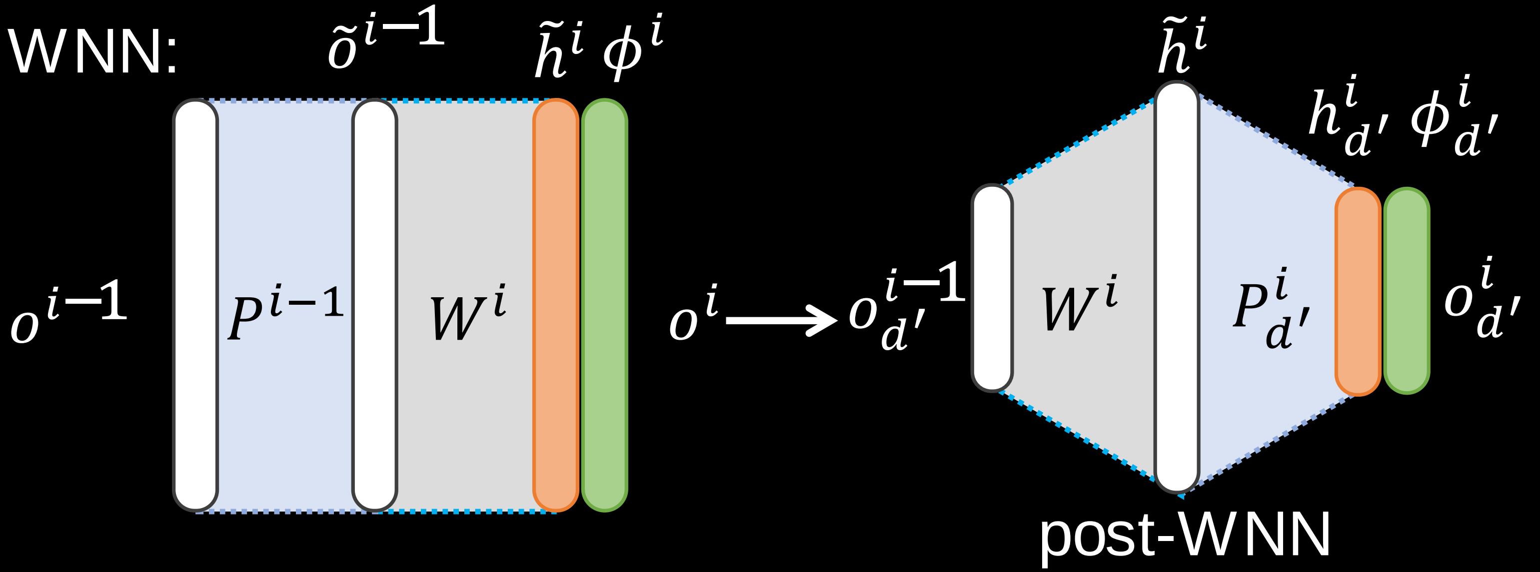
- In WNN *(Desjardins etal. NIPS15)* and GWNN *(Luo ICML17)*, $E\left[\tilde{o}^{i-1}\tilde{o}^{i-1^T}\right] = I$.
- In EigenNet *(Luo IJCAI17)*, each diagonal block of Fisher matrix F, $F_{ii} = E[\delta\tilde{h}^i \, \delta\tilde{h}^{i^T} \otimes \tilde{o}^{i-1}\tilde{o}^{i-1^T}] \approx I$.

From WNN to Post-WNN



Forward：

From WNN to Post-WNN



Forward：

$$\tilde{h}^i = W^i(o^{i-1}_{d'} - \mu^{i-1}_{d'})$$

(linear transformation)

$$h^i_{d'} = P^i_{d'}\tilde{h}^i, \qquad \mathrm{E}\left[h^i_{d'} {h^i_{d'}}^T\right] = I$$

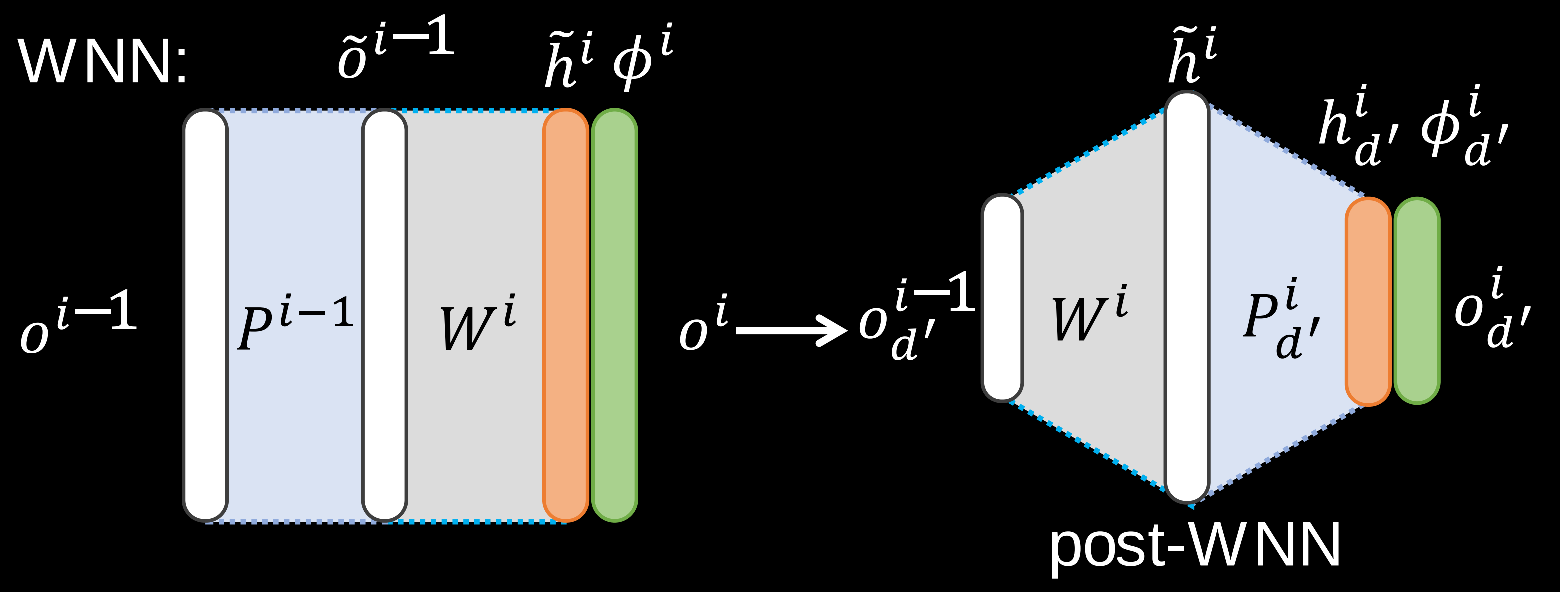
(truncated whitening)

$$\phi^i_{d'} = \frac{h^i_{d'}}{\sqrt{\mathrm{Var}(h^i_{d'})}}$$

(batch normalization)

(nonlinear activation)

From WNN to Post-WNN



Forward：

$$\tilde{h}^i = W^i(o_{d'}^{i-1} - \mu_{d'}^{i-1})$$

(linear transformation)

$$h_{d'}^i = P_{d'}^i \tilde{h}^i, \qquad \mathrm{E}\left[h_{d'}^i {h_{d'}^i}^T\right] = I$$

(truncated whitening)

$$\phi_{d'}^i = \frac{h_{d'}^i}{\sqrt{\mathrm{Var}(h_{d'}^i)}}$$

(batch normalization)

(nonlinear activation)

Post-whitening the i-th layer smooth solution space of W^{i+1}.

Summary

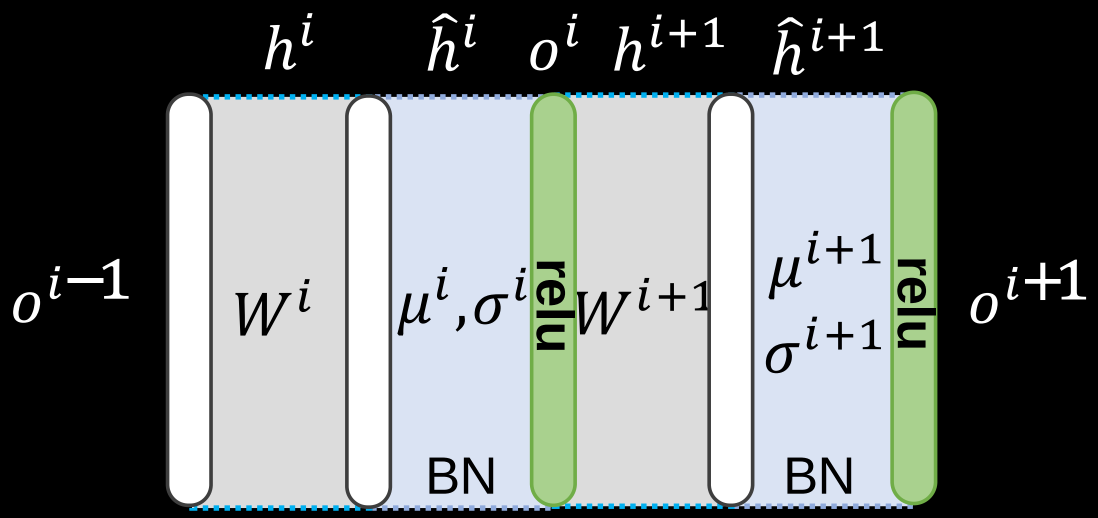
WNN, GWNN, and EigenNet are NGD.

Deep Learning turns Optimization Problems into Feed-Forward Network Design.

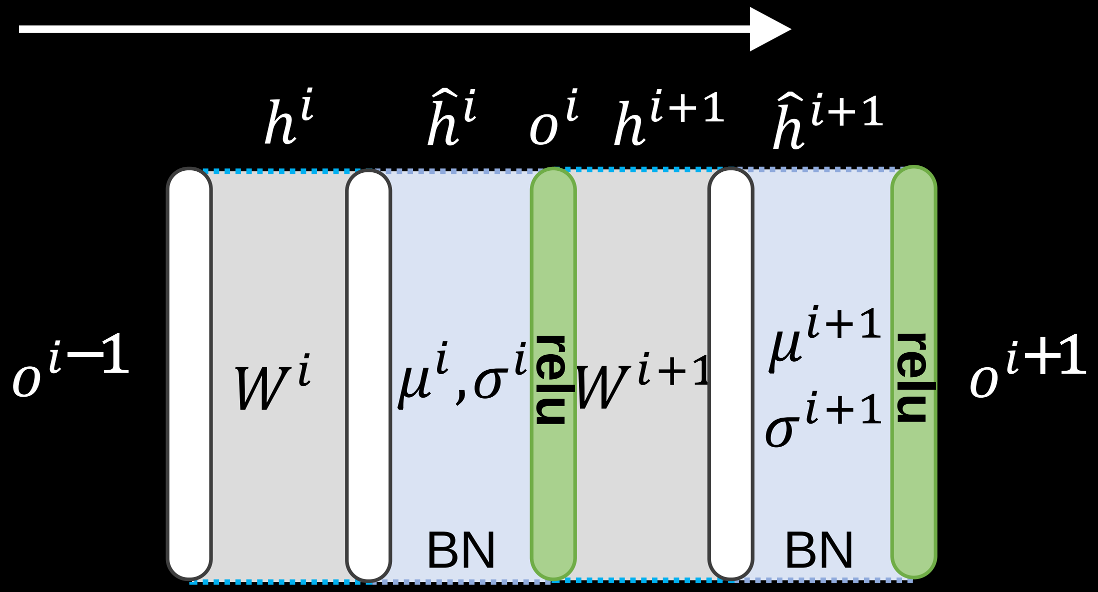
Batch Normalization (BN)

PART 2

Understand BN



Understand BN



Activations of single neuron

BN: $\hat{h}^i = \gamma^i \frac{h^i - \mathbb{E}[h^i]}{\sqrt{\mathrm{Var}(h^i)}} + \beta^i$, where

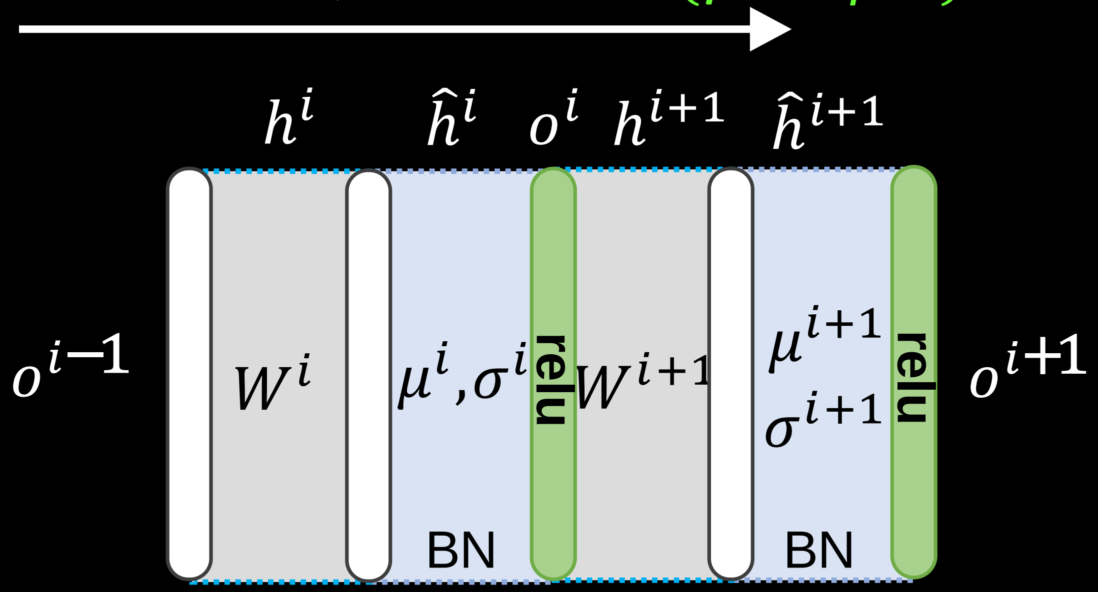
$h^i = [f_1, f_2, \ldots, f_M]$, M is minibatch size.

Understand BN

$\hat{h}^{i^T}\mathbf{1} = 0,\ \hat{h}^{i^T}\hat{h}^i = M({\gamma^i}^2 + {\beta^i}^2)$

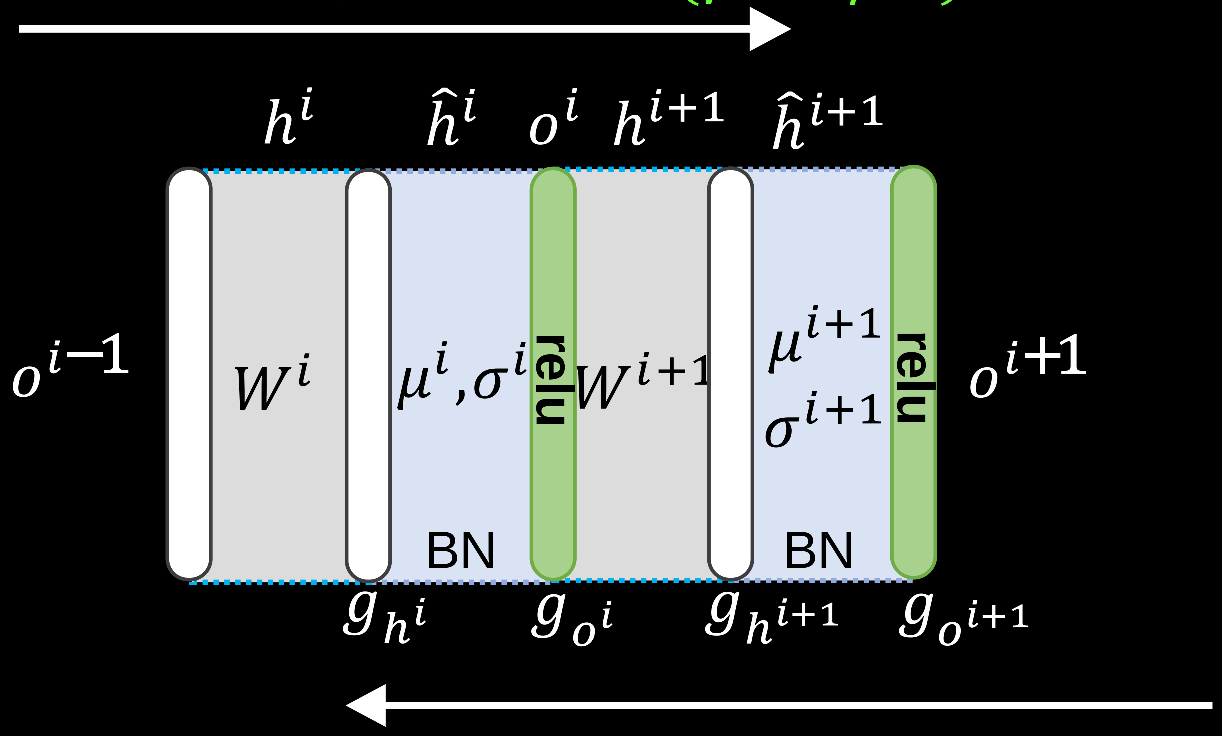
BN: $\hat{h}^i = \gamma^i \frac{h^i - \mathbb{E}[h^i]}{\sqrt{\mathrm{Var}(h^i)}} + \beta^i$, where

$h^i = [f_1, f_2, \dots, f_M]$, M is minibatch size.



Understand BN

$$\hat{h}^{i^T}\mathbf{1} = 0,\ \ \hat{h}^{i^T}\hat{h}^i = M({\gamma^i}^2 + {\beta^i}^2)$$



BN: $\hat{h}^i = \gamma^i \frac{h^i - \mathbb{E}[h^i]}{\sqrt{\mathrm{Var}(h^i)}} + \beta^i$, where

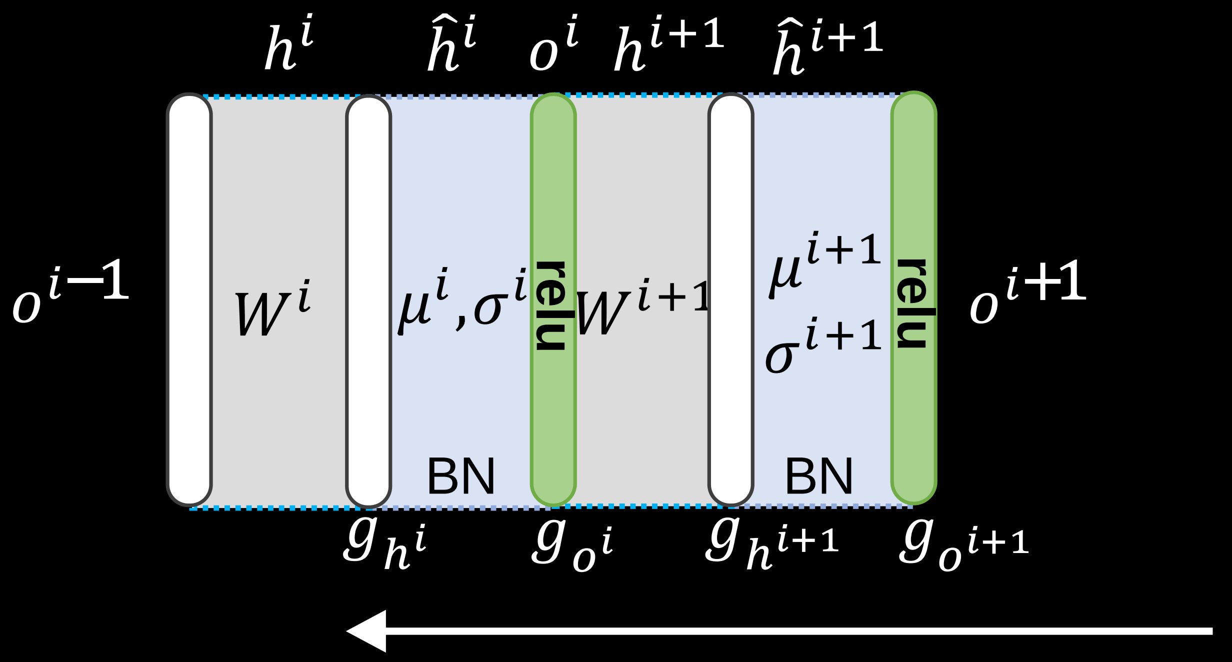
$h^i = [f_1, f_2, \ldots, f_M]$, M is minibatch size.

Understand BN

$$\hat{h}^{i\,T}\mathbf{1} = 0,\ \ \hat{h}^{i\,T}\hat{h}^{i} = M({\gamma^{i}}^{2} + {\beta^{i}}^{2})$$

BN: $\hat{h}^i = \gamma^i \frac{h^i - \mathbb{E}[h^i]}{\sqrt{\mathrm{Var}(h^i)}} + \beta^i$, where

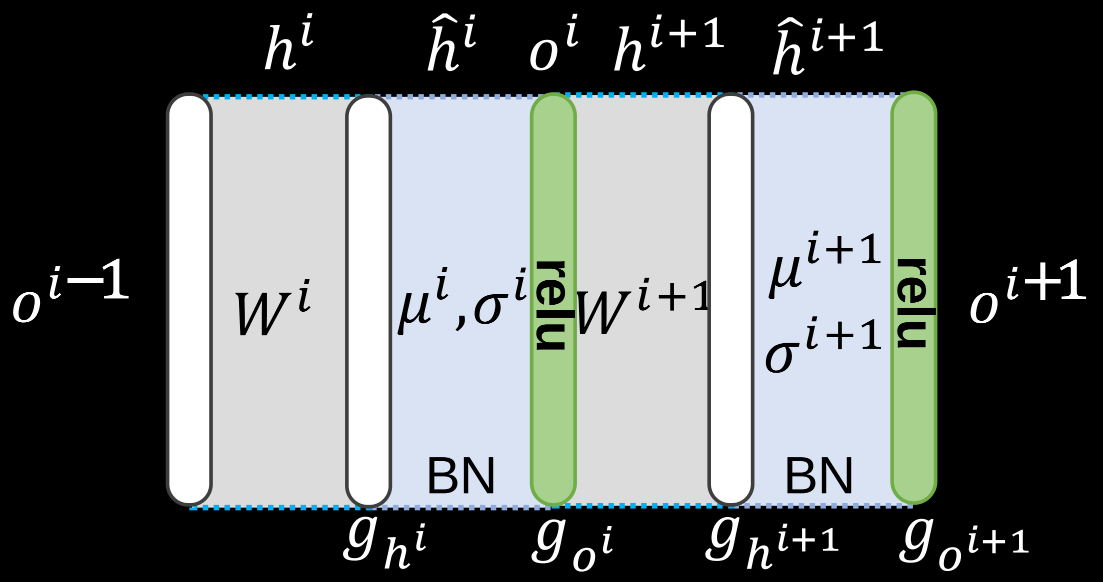
$h^i = [f_1, f_2, \ldots, f_M]$, M is minibatch size.



$$g_{h^{i+1}} = J^{BN} g_{o^{i+1}},\ \ J^{BN} = \frac{\partial \hat{h}^{i+1}}{\partial h^{i+1}} = \frac{\gamma^{i+1}}{\sigma^{i+1}}\left(I - \frac{[\tilde{h}^{i+1}, \mathbf{1}][\tilde{h}^{i+1}, \mathbf{1}]^T}{M}\right)$$

Understand BN

$$\hat{h}^{i^T}\mathbf{1} = 0,\ \ \hat{h}^{i^T}\hat{h}^{i} = M(\gamma^{i^2} + \beta^{i^2})$$



BN: $\hat{h}^i = \gamma^i \frac{h^i - \mathbb{E}[h^i]}{\sqrt{\mathrm{Var}(h^i)}} + \beta^i$, where

$h^i = [f_1, f_2, \ldots, f_M]$, M is minibatch size.

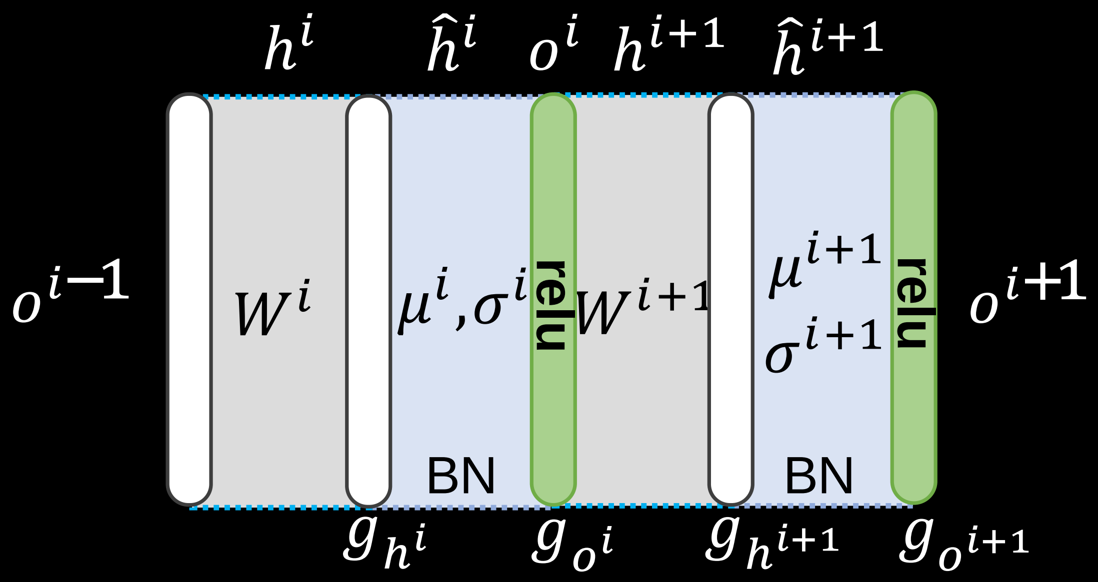
$$\mathrm{tr}(J^{BN}) = \frac{\gamma^{i+1}}{\sigma^{i+1}}(M-2),$$

$$\sigma^{i+1} = f(\gamma^i, \beta^i, W^i).$$

$$g_{h^{i+1}} = J^{BN} g_{o^{i+1}},\ \ J^{BN} = \frac{\partial \hat{h}^{i+1}}{\partial h^{i+1}} = \frac{\gamma^{i+1}}{\sigma^{i+1}}\left(I - \frac{[\tilde{h}^{i+1}, \mathbf{1}][\tilde{h}^{i+1}, \mathbf{1}]^T}{M}\right)$$

Understand BN

$$\hat{h}^{i^T}\mathbf{1} = 0,\ \hat{h}^{i^T}\hat{h}^i = M({\gamma^i}^2 + {\beta^i}^2)$$



BN: $\hat{h}^i = \gamma^i \frac{h^i - \mathbb{E}[h^i]}{\sqrt{\mathrm{Var}(h^i)}} + \beta^i$, where

$h^i = [f_1, f_2, \dots, f_M]$, M is minibatch size.

$$F^{(i+1,i+1)} = \mathbb{E}[g_{h^{i+1}} g_{h^{i+1}}^T] \otimes \mathbb{E}[o^i {o^i}^T],$$

$$o^i = \max(0, \hat{h}^i)$$

$$g_{h^{i+1}} = J^{BN} g_{o^{i+1}},\quad J^{BN} = \frac{\partial \hat{h}^{i+1}}{\partial h^{i+1}} = \frac{\gamma^{i+1}}{\sigma^{i+1}}\left(I - \frac{[\tilde{h}^{i+1}, \mathbf{1}][\tilde{h}^{i+1}, \mathbf{1}]^T}{M}\right)$$

Understand BN

BN Preserves Forward and Backward Information Flows Depended on γ.

Forward of i*-th BN and Backward of* $(i+1)$*-th BN Smooth the* $(i+1)$*-th ConvLayer.*

Regularization in BN

BN: $\hat{h}_j = \gamma \frac{h_j - \mathbb{E}[h]}{\sqrt{\mathrm{Var}(h)}} + \beta$, where $\boldsymbol{h} = [h_1, h_2, \ldots h_j, \ldots, h_M]$

Regularization in BN

BN: $\hat{h}_j = \gamma \frac{h_j - \mathbb{E}[h]}{\sqrt{\mathrm{Var}(h)}} + \beta$, where $\boldsymbol{h} = [h_1, h_2, \dots h_j, \dots, h_M]$

BN is an implicit Regularizer.

Regularization in BN

sample: x^j

BN: $\hat{h}_j = \gamma \frac{h_j - \mathbb{E}[h]}{\sqrt{\mathrm{Var}(h)}} + \beta$, where $\boldsymbol{h} = [h_1, h_2, \dots h_j, \dots, h_M]$

BN is an implicit Regularizer.

Regularization in BN

sample: x^j

BN: $\hat{h}_j = \gamma \frac{h_j - \mathbb{E}[h]}{\sqrt{\mathrm{Var}(h)}} + \beta$, where $\boldsymbol{h} = [h_1, h_2, \dots h_j, \dots, h_M]$

$\mu_{\mathcal{B}} = \mathbb{E}[h_j],\ \sigma_{\mathcal{B}} = \sqrt{\mathrm{Var}(h_j)}$

Regularization in BN

BN: $\hat{h}_j = \gamma \frac{h_j - \mu_{\mathcal{B}}}{\sigma_{\mathcal{B}}} + \beta$

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

Regularization in BN

BN: $\hat{h}_j = \gamma \frac{h_j - \mu_{\mathcal{B}}}{\sigma_{\mathcal{B}}} + \beta$

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

When $M \to P$, $\mu_{\mathcal{B}} \to \mu_{\mathcal{P}}$ and $\sigma_{\mathcal{B}} \to \sigma_{\mathcal{P}}$.

Regularization in BN

$$\mathcal{BN} \approx \mathcal{PN} + \mathcal{G}amma\ \mathcal{D}ecay$$

Regularization in BN

BN: $\hat{h}_j = \gamma \frac{h_j - \mu_{\mathcal{B}}}{\sigma_{\mathcal{B}}} + \beta$

Population Normalization (PN): $\bar{h}_j = \gamma \frac{h_j - \mu_{\mathcal{P}}}{\sigma_{\mathcal{P}}} + \beta$

Regularization in BN

BN: $\hat{h}_j = \gamma \frac{h_j - \mu_{\mathcal{B}}}{\sigma_{\mathcal{B}}} + \beta$ PN: $\bar{h}_j = \gamma \frac{h_j - \mu_{\mathcal{P}}}{\sigma_{\mathcal{P}}} + \beta$

An explicit regularizer:

Regularization in BN

BN: $\hat{h}_j = \gamma \frac{h_j - \mu_{\mathcal{B}}}{\sigma_{\mathcal{B}}} + \beta$ PN: $\bar{h}_j = \gamma \frac{h_j - \mu_{\mathcal{P}}}{\sigma_{\mathcal{P}}} + \beta$

An explicit regularizer:

$$\frac{1}{P}\sum_{j=1}^{P} \mathbb{E}_{\mu_{\mathcal{B}},\sigma_{\mathcal{B}}}[\mathcal{L}(\hat{h}_j^i)] \approx \frac{1}{P}\sum_{j=1}^{P} \mathcal{L}\left(\bar{h}_j^i\right) + \zeta\left(h_j^i\right)\gamma^2,$$

and $$\zeta\left(h_j^i\right) = \frac{\rho + 2}{8M} F_\gamma + \frac{1}{2M}\frac{1}{P}\sum_{j=1}^{P} \sigma(\bar{h}_j^i).$$

Regularization in BN

BN: $\hat{h}_j = \gamma \frac{h_j - \mu_{\mathcal{B}}}{\sigma_{\mathcal{B}}} + \beta$ PN: $\bar{h}_j = \gamma \frac{h_j - \mu_{\mathcal{P}}}{\sigma_{\mathcal{P}}} + \beta$

An explicit regularizer:

$$\frac{1}{P}\sum_{j=1}^{P} \mathbb{E}_{\mu_{\mathcal{B}},\sigma_{\mathcal{B}}}[\mathcal{L}(\hat{h}_j^i)] \approx \frac{1}{P}\sum_{j=1}^{P} \mathcal{L}(\bar{h}_j^i) + \zeta(h_j^i)\gamma^2,$$

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

Regularization in BN

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

Regularization in BN

punish gradient norm and correlations.

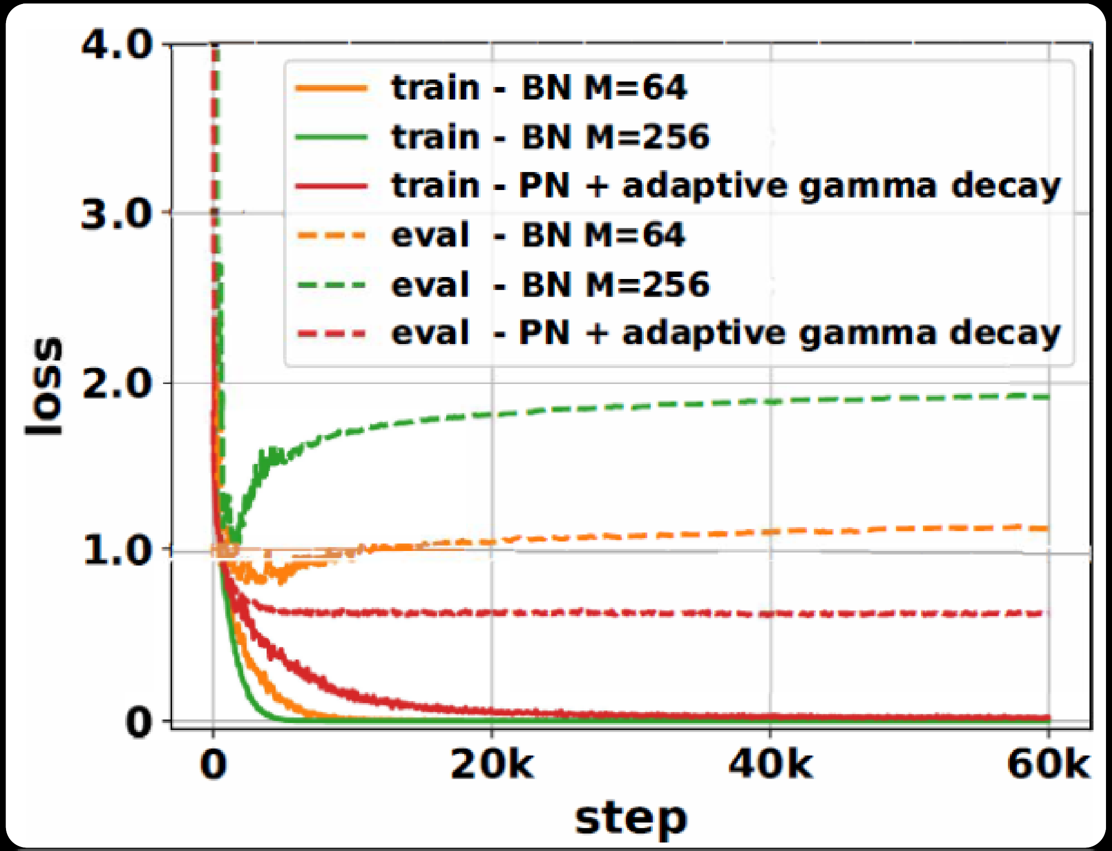
prevent reliance on single neuron.

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

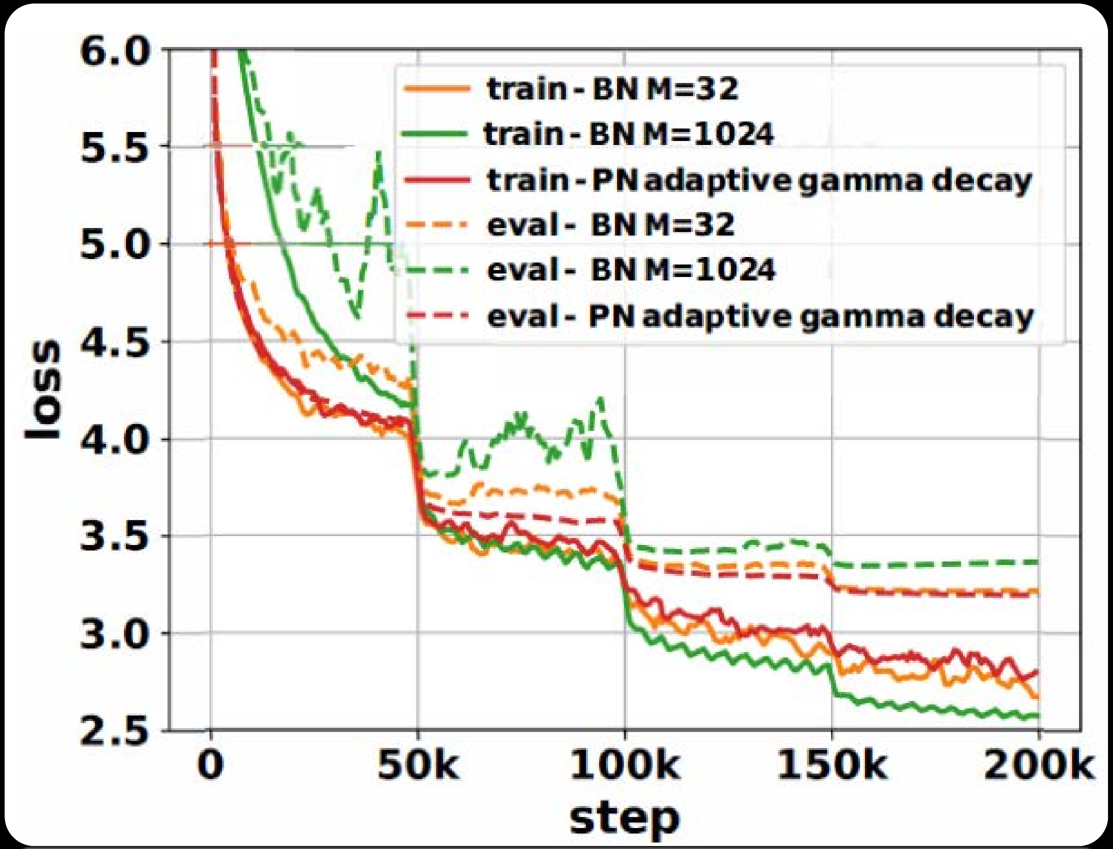
Regularization in BN

μ and σ in BN have Different Regularization Impacts.

Regularization in BN

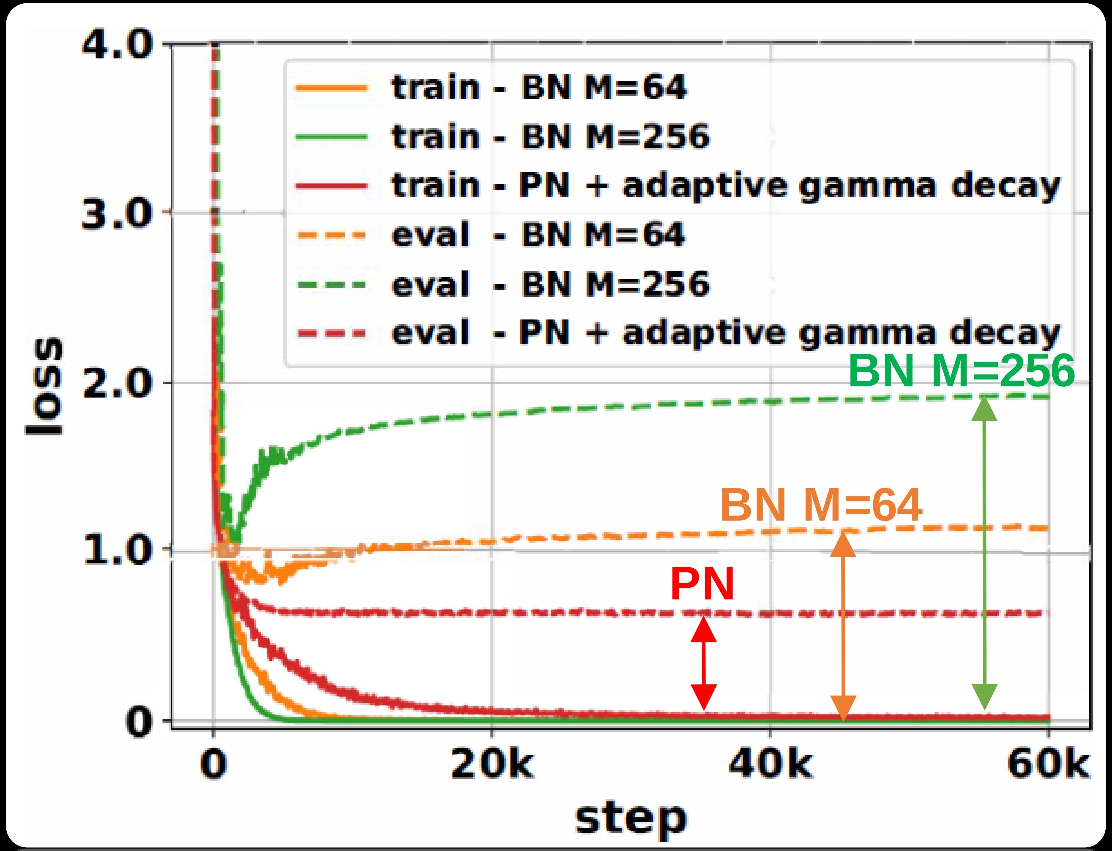


CIFAR10 (CNN6)

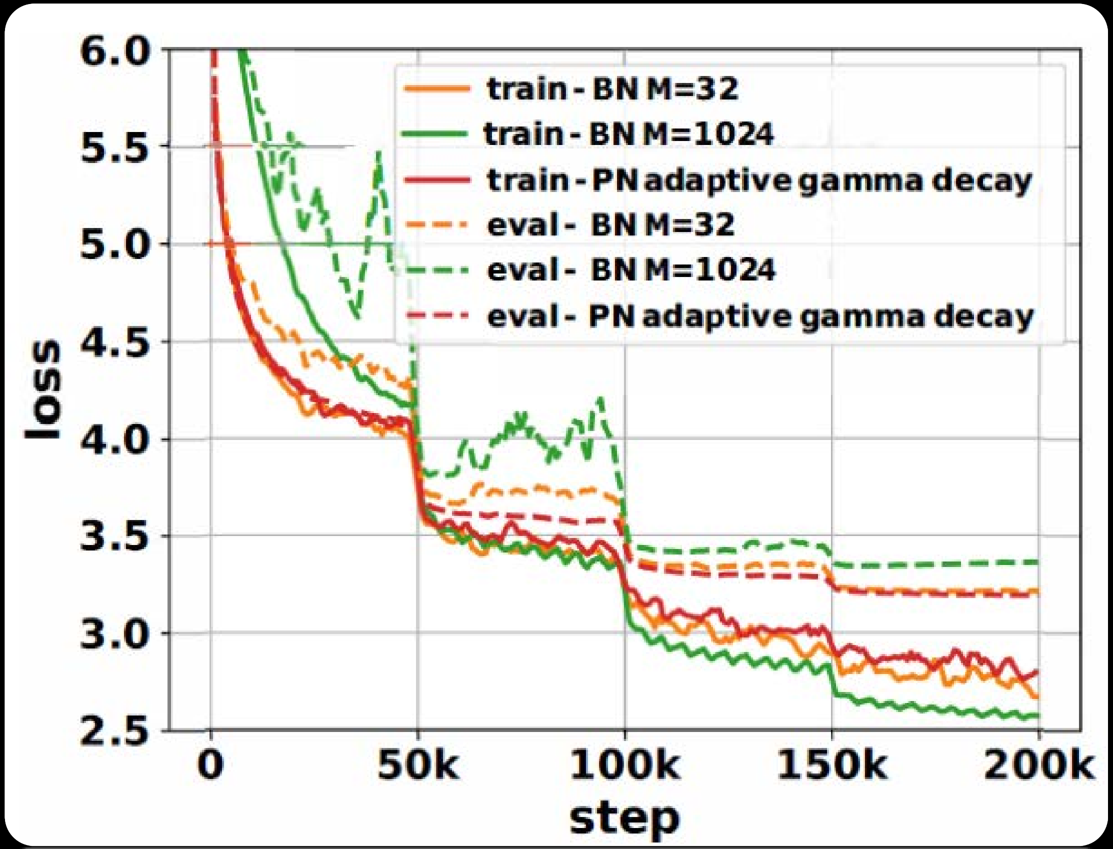


down-sampled ImageNet (R18)

Regularization in BN



CIFAR10 (CNN6)



down-sampled ImageNet (R18)

Summary

BN Preserves Forward and Backward Information Flows Depended on γ.

Forward of i*-th BN and Backward of* $(i+1)$*-th BN Smooth the* $(i+1)$*-th ConvLayer.*

Summary

BN is an implicit Regularizer.

BN ≈ PN + Gamma Decay

μ and σ in BN have Different Regularization Impacts.

BN turns data-dependent Regularizations into Feed-forward Estimation of Batch Statistics.

Switchable Normalization (SN)

PART 3

Switchable Normalization (SN)

A new Perspective for Deep Learning:

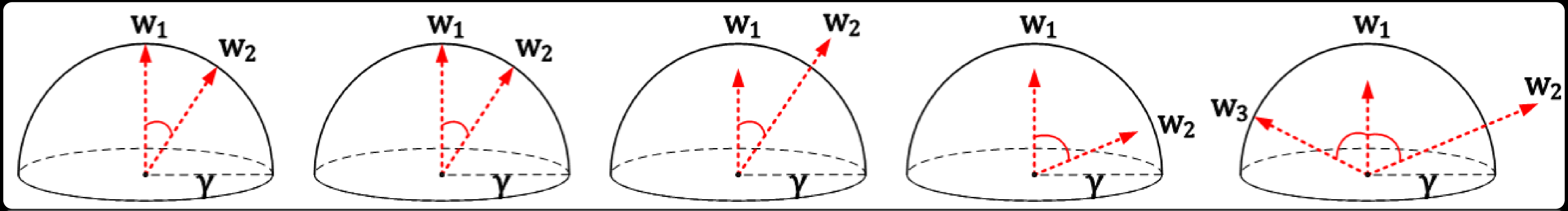
Each ConvLayer in a ConvNet needs its own Normalizer.

Switchable Normalization (SN)

$$\hat{h}_{\mathrm{SN}} = \frac{h - \sum_{z\in\Omega} \lambda_z \mu_z}{\sum_{z\in\Omega} \lambda'_z \sigma_z}, \quad \Omega = \{\mathrm{IN}, \mathrm{LN}, \mathrm{BN}\},$$

$$\forall z,\ \sum_{z\in\Omega} \lambda_z = 1,\ \sum_{z\in\Omega} \lambda'_z = 1.$$

Switchable Normalization (SN)



WN: $\frac{\boldsymbol{w}_c^T \boldsymbol{x}}{\|\boldsymbol{w}_c\|}$

IN: $\frac{\boldsymbol{w}_c^T \boldsymbol{x}}{\|\boldsymbol{w}_c\|}$

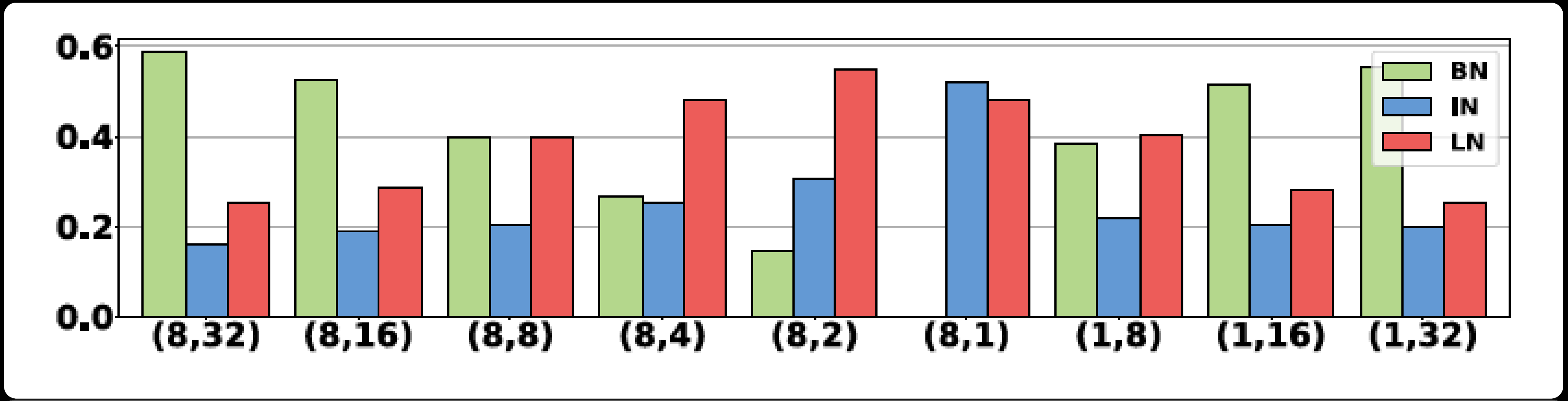
LN: $\frac{\boldsymbol{w}_c^T \boldsymbol{x}}{\sum_{c=1}^{C} \|\boldsymbol{w}_c\|}$

BN: $\frac{\boldsymbol{w}_c^T \boldsymbol{x}}{\|\boldsymbol{w}_c\|}$

reg. $\zeta(h)\gamma^2$

SN: $\hat{h}_{\mathrm{SN}} = \lambda_{\mathrm{BN}} \hat{h}_{\mathrm{BN}} + \lambda_{\mathrm{IN}} \hat{h}_{\mathrm{IN}} + \lambda_{\mathrm{LN}} \hat{h}_{\mathrm{LN}}$

Switchable Normalization (SN)



When batch size decreases, LN ratio increases while BN decreases.

Switchable Normalization (SN)

backbone	head	AP	$AP_{.5}$	$AP_{.75}$	AP_l	AP_m	AP_s
$BN^\dagger$	–	36.7	58.4	39.6	48.1	39.8	21.1
$BN^\dagger$	GN	37.2	58.0	40.4	48.6	40.3	21.6
$BN^\dagger$	SN	38.0	59.4	41.5	48.9	41.3	22.7
GN	GN	38.2	58.7	41.3	49.6	41.0	22.4
SN	SN	**39.3**	**60.9**	**42.8**	**50.3**	**42.7**	**23.5**

Table 3: **Faster R-CNN+FPN** using ResNet50 and FPN with 1x LR schedule. $BN^\dagger$ represents BN is frozen. The best results are bold.

backbone	head	AP^b	$AP^b_{.5}$	$AP^b_{.75}$	AP^m	$AP^m_{.5}$	$AP^m_{.75}$
$BN^\dagger$	–	38.6	59.5	41.9	34.2	56.2	36.1
$BN^\dagger$	GN	39.5	60.0	43.2	34.4	56.4	36.3
$BN^\dagger$	SN	40.0	61.0	43.3	34.8	57.3	36.3
GN	GN	40.2	60.9	43.8	35.7	57.8	38.0
GN	SN	40.4	61.4	44.2	36.0	58.4	38.1
SN	SN	**41.0**	**62.3**	**45.1**	**36.5**	**58.9**	**38.7**

Table 4: **Mask R-CNN** using ResNet50 and FPN with 2x LR schedule. $BN^\dagger$ represents BN is frozen without finetuning. The best results are bold.

Switchable Normalization (SN)

	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	**38.7**	**39.2**	**71.2**	**75.1**

Table 5: **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 mutli-GPU synchronization of BN. SN finetunes from $(8, 2)$ pretrained model.

Switchable Normalization (SN)

	batch=8, length=32		batch=4, length=32	
	top1	top5	top1	top5
BN	73.2	90.9	72.1	90.0
GN	73.0	90.6	72.8	90.6
SN	**73.5**	**91.3**	**73.3**	**91.2**

Table 6: **Results of Kinetics dataset.** In training, the clip length of 32 frames is regularly sampled with a frame interval of 2. We study a batch size of 8 or 4 clips per GPU. BN is not synchronized across GPUs. SN finetunes from $(8, 2)$ pretrained model.

Summary

A Novel Viewpoint for Deep Learning:

Different Normalization Layers in a Deep ConvNet

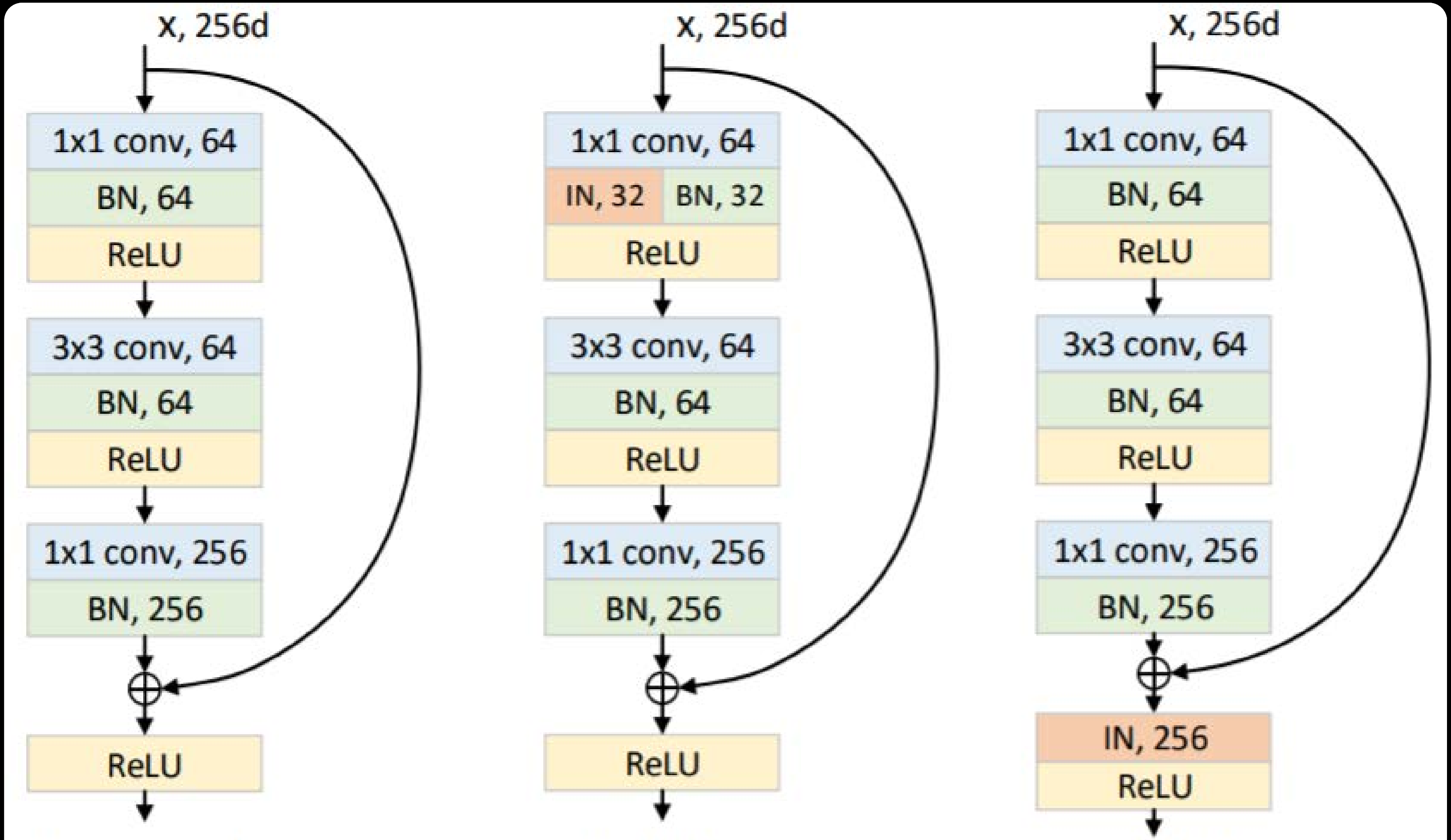
Require Different Normalizers.

SN would be valuable in “any” problem that needs normalizations, according to its theoretical property.

Instance-Batch Normalization (IBN-Net) and Kalman Normalization (KN)

PART 4

Instance-Batch Normalization



(a) original (b) IBN-a (c) IBN-b

Instance-Batch Normalization

Table 2. Results of IBN-Net over other CNNs on ImageNet validation set. The performance gains are shown in the brackets. More detailed descriptions of these IBN-Nets are provided in the supplementary material.

Model	original	re-implementation	IBN-Net-a
	top1/top5 err.	top1/top5 err.	top1/top5 err.
DenseNet121 [13]	25.0/-	24.96/7.85	24.47/7.25 (0.49/0.60)
DenseNet169 [13]	23.6/-	24.02/7.06	23.25/6.51 (0.79/0.55)
ResNet50 [8]	24.7/7.8	24.27/7.08	22.54/6.32 (1.73/0.76)
ResNet101 [8]	23.6/7.1	22.48/6.23	21.39/5.59 (1.09/0.64)
ResNeXt101 [31]	21.2/5.6	21.31/5.74	20.88/5.42 (0.43/0.32)
SE-ResNet101 [12]	22.38/6.07	21.68/5.88	21.25/5.51 (0.43/0.37)

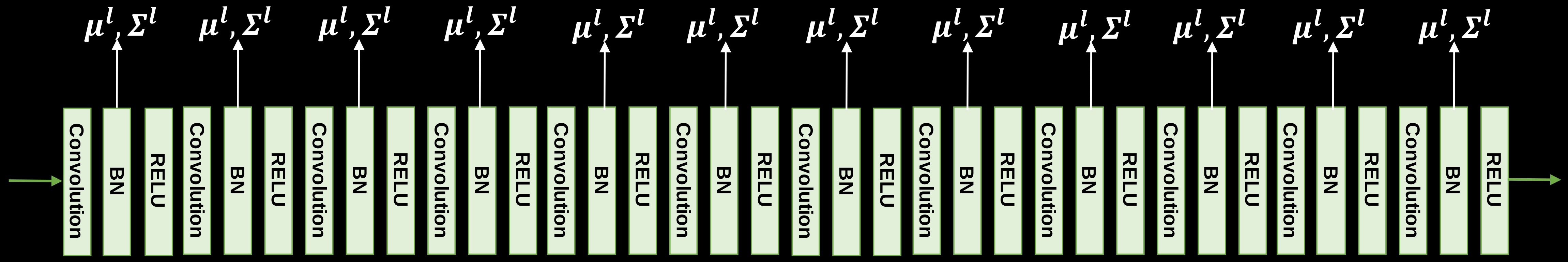
Instance-Batch Normalization

Table 6. Results on Cityscapes-GTA dataset. Mean IoU for both within domain evaluation and cross domain evaluation is reported.

Train	Test	Model	mIoU(%)	Pixel Acc.(%)
Cityscapes	Cityscapes	ResNet50	64.5	93.4
		IBN-Net50-a	**69.1**	**94.4**
		IBN-Net50-b	67.0	94.3
	GTA5	ResNet50	29.4	71.9
		IBN-Net50-a	32.5	71.4
		IBN-Net50-b	**37.9**	**78.8**
GTA5	GTA5	ResNet50	61.0	91.5
		IBN-Net50-a	**64.8**	**92.5**
		IBN-Net50-b	64.2	92.4
	Cityscapes	ResNet50	22.2	53.5
		IBN-Net50-a	26.0	60.9
		IBN-Net50-b	**29.6**	**66.8**

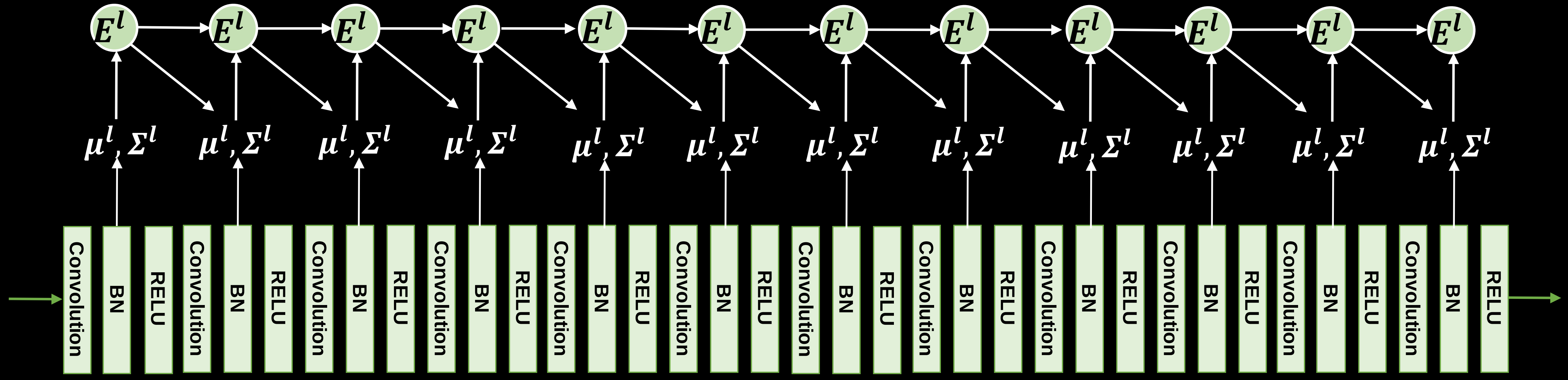
Kalman Normalization (KN)

- Propagate statistics in BN through the entire CNN by learning a transition matrix E.
- KN is designed for training with micro-batch.



Kalman Normalization (KN)

- Propagate statistics in BN through the entire CNN by learning a transition matrix E.
- KN is designed for training with micro-batch.



Comparisons

ResNet50	BN	GN	SN	KN
256, 32	76.4	75.9	**77.2**	76.8
32, 4	72.7	75.8	**75.9**	-
256, 4	-	-	**77.2**	76.1

Discussions and Future Work

PART 5

Recap of Results

1. Deep Learning turns optimization problem into feed-forward computations, e.g. WNN, GWNN, and EigenNet are NGD.

Recap of Results

1. Deep Learning turns optimization problem into feed-forward computations, e.g. WNN, GWNN, and EigenNet are NGD.
2. BN is an implicit regularizer, whose explicit form is “BN ≈ PN + Gamma Decay”.
   - Compute regularization in batch to replace batch statistics in BN.

Recap of Results

1. Deep Learning turns optimization problem into feed-forward computations, e.g. WNN, GWNN, and EigenNet are NGD.
2. BN is an implicit regularizer, whose explicit form is “BN ≈ PN + Gamma Decay”.
   - Compute regularization in batch to replace batch statistics in BN.
3. New research direction: *different normalization layers in a ConvNet use different normalizers.*
   - Switchable Normalization (SN) would be applicable in “any” problem because of its characteristic.

Working Papers

- ✓ “Do Normalization Layers in a Deep ConvNet Really Need to be Distinct?”
- ✓ “Learning Sparse Switchable Normalization via SparsestMax”
  - SoftMax $\rightarrow$ SparseMax $\rightarrow$ SparsestMax
- ✓ Understanding Learning Dynamics and Generalization of SN
  - Loss functions, input distributions, over-parameterization
- ✓ SNv2

Codebase

https://github.com/switchablenorms

Switchable-Normalization

Code for Switchable Normalization from "Differentiable Learning-to-Normalize via Switchable Normalization", https://arxiv.org/abs/1806.10779

HTML 498 71

SwitchNorm_Detection

Forked from roytseng-tw/Detectron.pytorch

The code of Switchable Normalization for object detection based on Detectron.pytorch.

Python 68 9

SwitchNorm_Segmentation

Switchable Normalization for semantic image and scene parsing.

Python 18 1

SSN, SparsestMax, SNv2

SN: Kinetics, MegaFace, GANs, …

https://github.com/XingangPan/IBN-Net

IBN-Net

Instance-Batch Normalization Networks (ECCV2018)

Python 321 49

Team Members

Jiamin Ren
SenseTime Research

Xinjiang Wang
SenseTime Research

Ruimao Zhang
SenseTime Research

Zhanglin Peng
SenseTime Research

Lingyun Wu
SenseTime Research

Xingang Pan
CUHK

Wenqi Shao
CUHK

Guangrun Wang
Sun Yat-sen U

Reference

[1] Guillaume Desjardins, Karen Simonyan, Razvan Pascanu, Koray Kavukcuoglu. "Natural Neural Networks", NIPS 2015

[2] Sergey Ioffe, Christian Szegedy. "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift", ICML 2015

[3] **Ping Luo**. "Learning Deep Architectures via Generalized Whitened Neural Networks", ICML 2017

[4] **Ping Luo**. "EigenNet: Towards Fast and Structural Learning of Deep Neural Networks", IJCAI 2017

[5] **Ping Luo**, Jiamin Ren, Zhanglin Peng, Ruimao Zhang, Jingyu Li. "Differentiable Learning-to-Normalize via Switchable Normalization", arXiv:1806.10779, 2018

[6] **Ping Luo**, Xinjiang Wang, Wenqi Shao, Zhanglin Peng. "Towards Understanding Regularization in Batch Normalization", arXiv:1809.00846, 2018

[7] Guangrun Wang, Jiefeng Peng, **Ping Luo**, Xinjiang Wang, Liang Lin. "Kalman Normalization: Normalizing the Normalizers across Layers", NIPS 2018

[8] Xingang Pan, **Ping Luo**, Jianping Shi, Xiaoou Tang. "Two at Once: Enhancing Learning and Generalization Capacities via IBN-Net", ECCV2018.

Appendix

PART 6

Kalman Normalization (KN)

True value of h^l:

$$h^l = E^l h^{l-1} + u^l, \quad u^l \sim \mathcal{N}(0, R^l)$$

transition matrix   bias

Kalman Normalization (KN)

True value of h^l:

$$h^l = E^l h^{l-1} + u^l, \quad u^l \sim \mathcal{N}(0, R^l)$$

$$z^l = h^l + v^l$$

bias

observed value of a mini-batch

Kalman Normalization (KN)

True value of h^l:

$$h^l = E^l h^{l-1} + u^l, \quad u^l \sim \mathcal{N}(0, R^l)$$

$$z^l = h^l + v^l$$

Estimated mean of h^l:

$$\hat{\mu}^{l|l-1} = \mathbb{E}[h^l] = \mathbb{E}[E^l h^{l-1} + u^l]$$

Kalman Normalization (KN)

True value of h^l:

$$h^l = E^l h^{l-1} + u^l, \quad u^l \sim \mathcal{N}(0, R^l)$$

$$z^l = h^l + v^l$$

Estimated mean of h^l:

$$\hat{\mu}^{l|l-1} = \mathbb{E}[h^l] = \mathbb{E}[E^l h^{l-1} + u^l]$$

Estimated variance of h^l:

$$\hat{\Sigma}^{l|l-1} = \mathrm{Cov}(h^l - \hat{\mu}^{l|l-1}) = E^l \hat{\Sigma}^{l-1|l-1} {E^l}^T + R^l$$

Kalman Normalization (KN)

True value of h^l:

$$h^l = E^l h^{l-1} + u^l, \quad u^l \sim \mathcal{N}(0, R^l)$$

$$z^l = h^l + v^l$$

Estimated mean of h^l:

$$\hat{\mu}^{l|l-1} = \mathbb{E}[h^l] = \mathbb{E}[E^l h^{l-1} + u^l]$$

$$\hat{\mu}^{l|l} = \hat{\mu}^{l|l-1} + q^l (z^l - \hat{\mu}^{l|l-1})$$

estimation of mean
in current layer

estimation of mean
in previous layer

Estimated variance of h^l:

$$\hat{\Sigma}^{l|l-1} = \mathrm{Cov}(h^l - \hat{\mu}^{l|l-1}) = E^l \hat{\Sigma}^{l-1|l-1} {E^l}^T + R^l$$

$$\hat{\Sigma}^{l|l} = \mathrm{Cov}(h^l - \hat{\mu}^{l|l}) = \hat{\Sigma}^{l|l-1} + q^l (C^l - \hat{\Sigma}^{l|l-1}) + \left(1 - q^l\right) q^l (z^l - \hat{\mu}^{l|l-1})^2$$

Kalman Normalization (KN)

True value of h^l:

$$h^l = E^l h^{l-1} + u^l, \quad u^l \sim \mathcal{N}(0, R^l)$$

$$z^l = h^l + v^l$$

Estimated mean of h^l:

$$\hat{\mu}^{l|l-1} = \mathbb{E}[h^l] = \mathbb{E}[E^l h^{l-1} + u^l]$$

$$\hat{\mu}^{l|l} = \hat{\mu}^{l|l-1} + q^l \underbrace{(z^l - \hat{\mu}^{l|l-1})}_{\text{bias between the observed mean and intermediate estimation}}$$

estimation of mean in current layer

Estimated variance of h^l:

$$\hat{\Sigma}^{l|l-1} = \mathrm{Cov}(h^l - \hat{\mu}^{l|l-1}) = E^l \hat{\Sigma}^{l-1|l-1} {E^l}^T + R^l$$

$$\hat{\Sigma}^{l|l} = \mathrm{Cov}(h^l - \hat{\mu}^{l|l}) = \hat{\Sigma}^{l|l-1} + q^l (C^l - \hat{\Sigma}^{l|l-1}) + (1 - q^l) q^l (z^l - \hat{\mu}^{l|l-1})^2$$

Kalman Normalization (KN)

True value of h^l:

$$h^l = E^l h^{l-1} + u^l, \quad u^l \sim \mathcal{N}(0, R^l)$$

$$z^l = h^l + v^l$$

Estimated mean of h^l:

$$\hat{\mu}^{l|l-1} = \mathbb{E}[h^l] = \mathbb{E}[E^l h^{l-1} + u^l]$$

$$\hat{\mu}^{l|l} = \hat{\mu}^{l|l-1} + q^l (z^l - \hat{\mu}^{l|l-1})$$

estimation of mean
in current layer

gain

Estimated variance of h^l:

$$\hat{\Sigma}^{l|l-1} = \mathrm{Cov}(h^l - \hat{\mu}^{l|l-1}) = E^l \hat{\Sigma}^{l-1|l-1} {E^l}^T + R^l$$

$$\hat{\Sigma}^{l|l} = \mathrm{Cov}(h^l - \hat{\mu}^{l|l}) = \hat{\Sigma}^{l|l-1} + q^l (C^l - \hat{\Sigma}^{l|l-1}) + \left(1 - q^l\right) q^l (z^l - \hat{\mu}^{l|l-1})^2$$

Kalman Normalization (KN)

True value of h^l:

$$h^l = E^l h^{l-1} + u^l, \quad u^l \sim \mathcal{N}(0, R^l)$$

$$z^l = h^l + v^l$$

Estimated mean of h^l:

$$\hat{\mu}^{l|l-1} = \mathbb{E}[h^l] = \mathbb{E}[E^l h^{l-1} + u^l]$$

$$\hat{\mu}^{l|l} = \hat{\mu}^{l|l-1} + q^l(z^l - \hat{\mu}^{l|l-1})$$

Estimated variance of h^l:

$$\hat{\Sigma}^{l|l-1} = \mathrm{Cov}(h^l - \hat{\mu}^{l|l-1}) = E^l \hat{\Sigma}^{l-1|l-1} {E^l}^T + R^l$$

$$\hat{\Sigma}^{l|l} = \mathrm{Cov}(h^l - \hat{\mu}^{l|l}) = \hat{\Sigma}^{l|l-1} + q^l(\boxed{C^l} - \hat{\Sigma}^{l|l-1}) + (1 - q^l)q^l(z^l - \hat{\mu}^{l|l-1})^2$$

observed covariance matrix

Kalman Normalization (KN)

True value of h^l:

$$h^l = E^l h^{l-1} + u^l, \quad u^l \sim \mathcal{N}(0, R^l)$$

$$z^l = h^l + v^l$$

Estimated mean of h^l:

$$\hat{\mu}^{l|l-1} = \mathbb{E}[h^l] = \mathbb{E}[E^l h^{l-1} + u^l]$$

$$\hat{\mu}^{l|l} = \hat{\mu}^{l|l-1} + q^l(z^l - \hat{\mu}^{l|l-1})$$

Estimated variance of h^l:

$$\hat{\Sigma}^{l|l-1} = \mathrm{Cov}(h^l - \hat{\mu}^{l|l-1}) = \boxed{E^l} \hat{\Sigma}^{l-1|l-1} {E^l}^T + \boxed{R^l}$$

$$\hat{\Sigma}^{l|l} = \mathrm{Cov}(h^l - \hat{\mu}^{l|l}) = \hat{\Sigma}^{l|l-1} + \boxed{q^l}(C^l - \hat{\Sigma}^{l|l-1}) + (1 - q^l)q^l(z^l - \hat{\mu}^{l|l-1})^2$$

optimized in training