Batch Normalization : Accelerating Deep Network Training by Reducing Internal Covariate Shift

SKT Fellowship

LEE JINKYU

Department of Civil, Environmental and Architectural Engineering, Korea University

November 19, 2022

Stochastic Gradient Descent

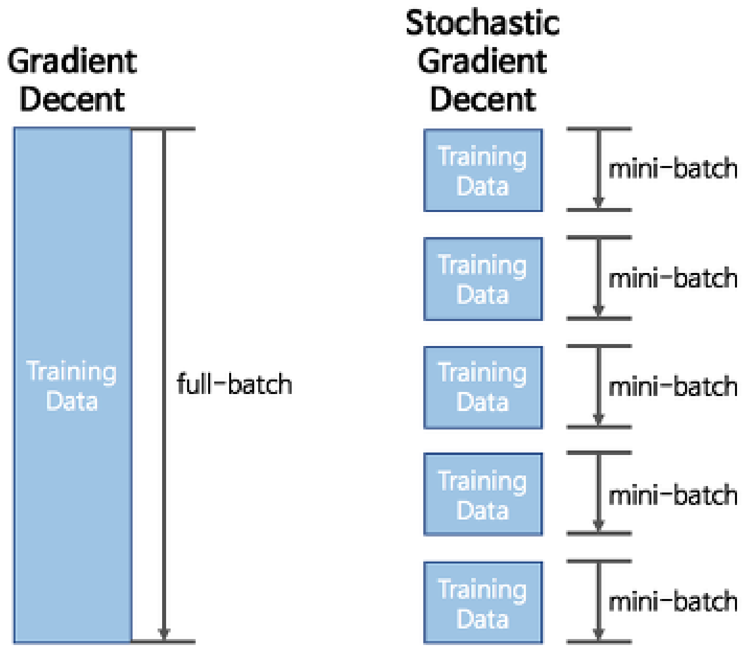
Traditional Optimizer

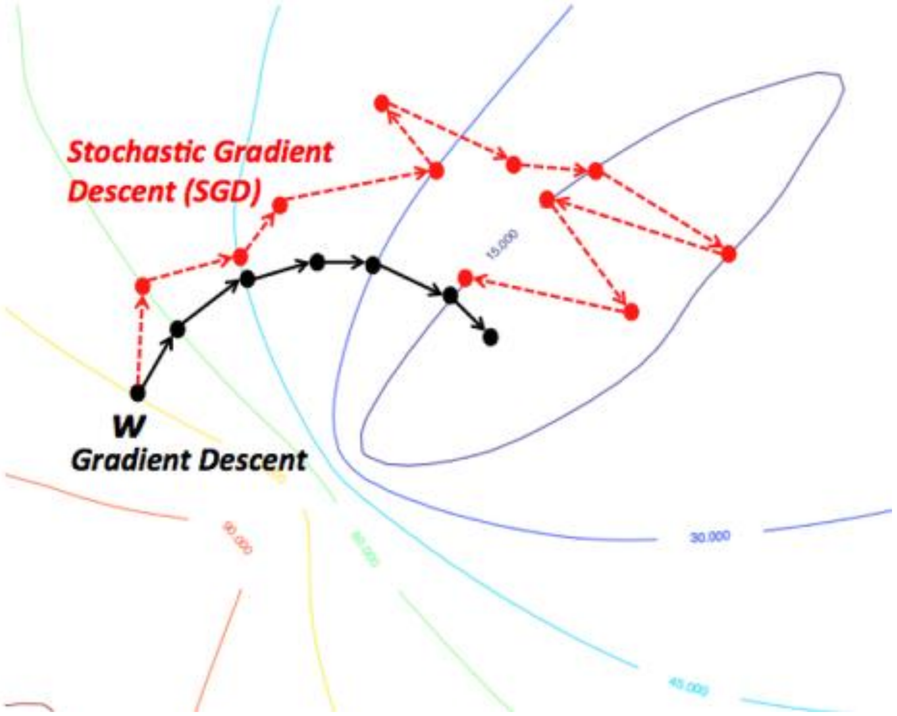
$$\theta=\theta-\eta \nabla \theta J(\theta)$$

손실 값을 통해 기울기를 갱신하는 Gradient Descent 방식에서 mini-batch를 도입한 Optimizer이다

Full-batch(전체 데이터)에 대해 손실 값을 계산하고 갱신해야하는 GD에 비해 계산량이 현저히 줄어든다

훨씬 빠른 속도로 매개변수들이 갱신되며, 많은 반복을 통해 local minina에 빠지지 않고 학습될 가능성이 높다.



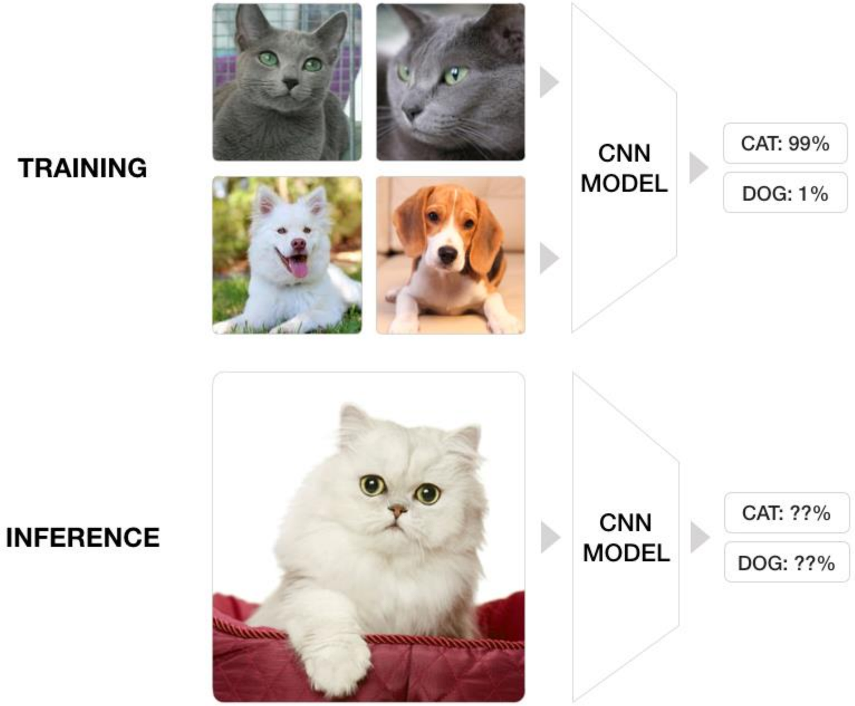


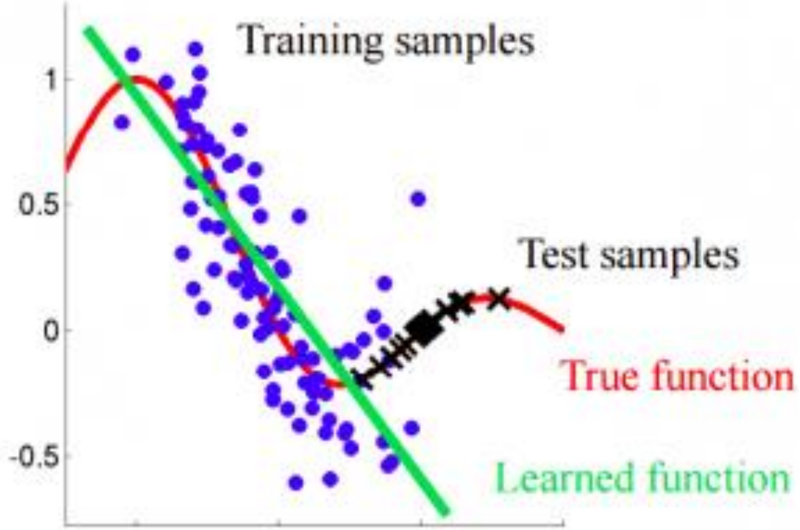
Covariate Shift - Definition

What if train/test data`s distribution are different

훈련 데이터와 검증, 테스트 데이터의 분포가 다른 경우에 나타난다

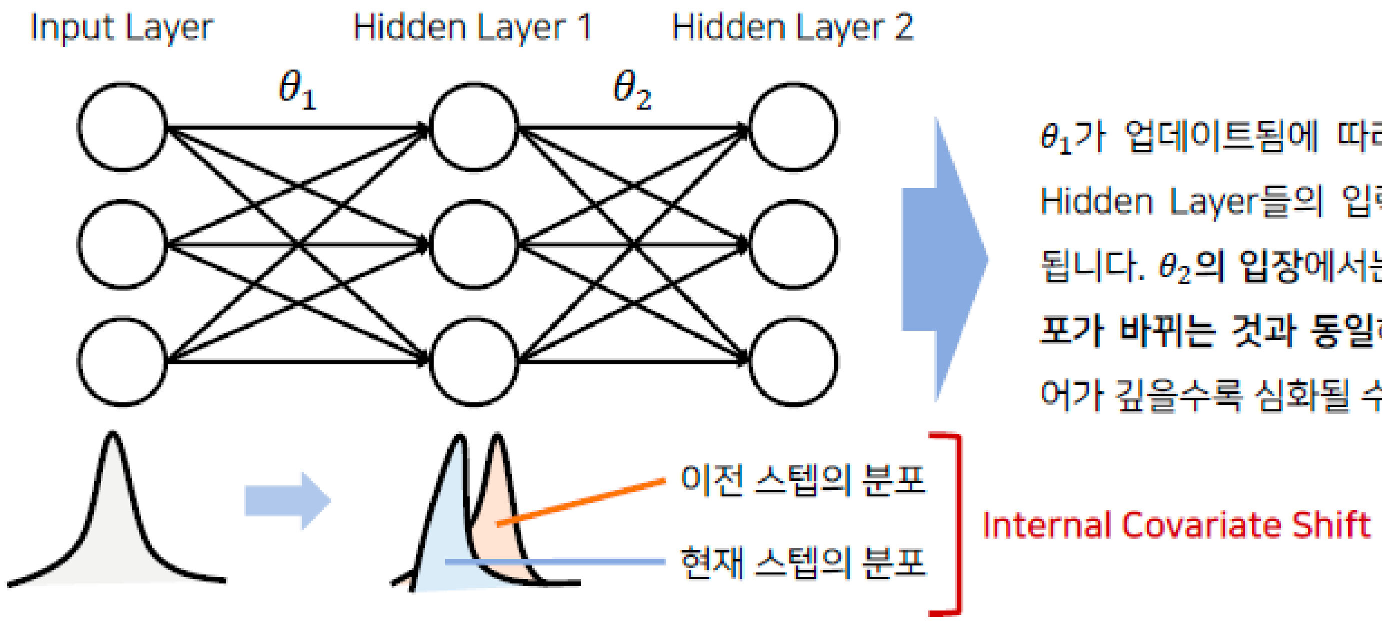
동물 종 분류 / 얼굴 생성 알고리즘 등 다양한 분야에서 나타나며 모델의 성능을 크게 저하시킬 수 있다





Internal Covariate Shift

Covariate Shift in Deep Neural Network



θ_1가 업데이트됨에 따라 뒤쪽에 있는 Hidden Layer들의 입력 분포가 변경됩니다. θ_2의 입장에서는 **매번 입력 분포가 바뀌는 것과 동일**하며 이는 레이어가 깊을수록 심화될 수 있습니다.

Sigmoid의 특성 상 x값이 커지면 saturated 되어 gradient vanishing or slow converge가 발생한다
→ 네트워크 내의 Covariate Shift는 saturated regime을 발생시킨다
→ 네트워크의 깊이가 깊어질수록 Covariate Shift의 문제가 심각해진다

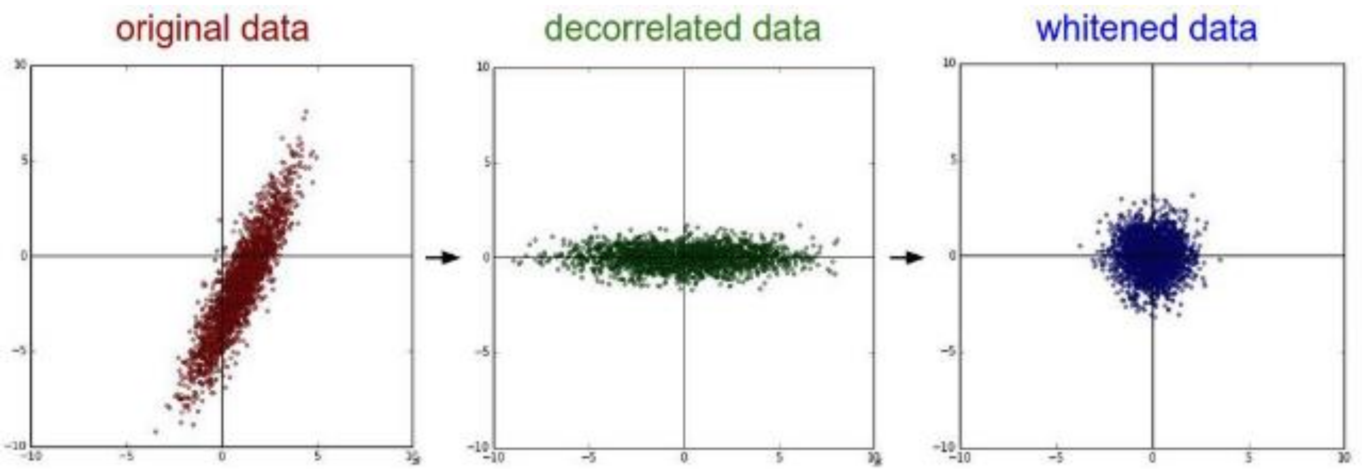
Relu, 좋은 초기값 설정에 더불어 **input 값의 분포를 같게 만드는 것은 Covariate Shift를 방지하고, 학습을 촉진시킨다**

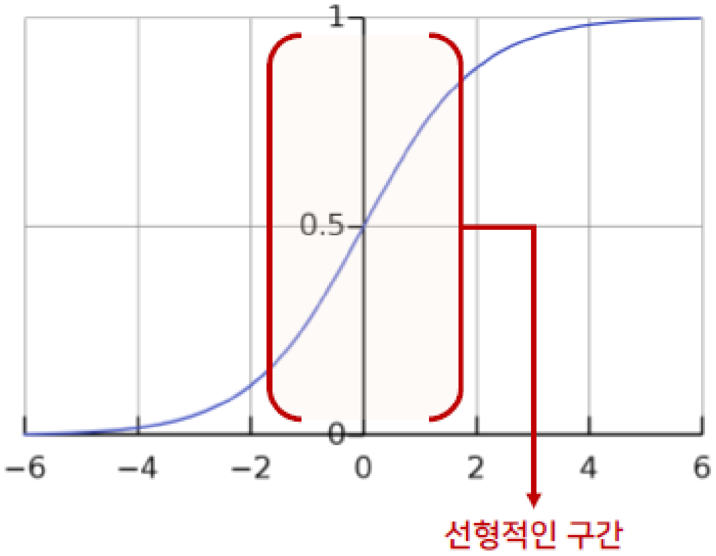
Whitening

Try to reduce Internal Covariate Shift (But failed)

매 훈련 step마다 whitening을 시도했다
→ 정규화가 되는 방향으로 parameters가 update되고, 이는 gradient descent의 성능을 저하시킨다
→ 학습이 가능한 parameters들이 무시, 역전파로 학습이 불가능하게 될 수 있다
→ 평균이 0, 표준편차가 1로 데이터 전체가 fitting 되어 non-linearity로서의 기능을 못할 수 있다
→ 네트워크 밖에 존재하며 갱신되지 않는다

fect of the gradient step. For example, consider a layer with the input u that adds the learned bias b, and normalizes the result by subtracting the mean of the activation computed over the training data: $\widehat{x} = x - E[x]$ where $x = u + b$, $\mathcal{X} = \{x_{1\ldots N}\}$ is the set of values of x over the training set, and $\mathrm{E}[x] = \frac{1}{N}\sum_{i=1}^{N} x_i$. If a gradient descent step ignores the dependence of $\mathrm{E}[x]$ on b, then it will update $b \leftarrow b + \Delta b$, where $\Delta b \propto -\partial\ell/\partial\widehat{x}$. Then $u + (b + \Delta b) - \mathrm{E}[u + (b + \Delta b)] = u + b - \mathrm{E}[u + b]$.





Batch Normalization

Reason why use Gamma, Beta

평균 0, 분산 1로 정규화 시켜 데이터 분포를 균일하게 한다

이 때, Activation F에 따라 non-linearity로서의 기능이 사라질 수 있다
→ Gamma, Beta를 통해 이를 해결한다
→ 학습, 갱신이 가능하며 Optimal uncorrelated distribution을 구할 수 있다

Full-batch가 아닌 mini-batch로 파라미터, 손실 값 갱신한다

0으로 나눠지는 것을 막기 위해 epsilon 추가한다

$$\frac{\partial \ell}{\partial \widehat{x}_i} = \frac{\partial \ell}{\partial y_i} \cdot \gamma$$

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

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

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

$$\frac{\partial \ell}{\partial \gamma} = \sum_{i=1}^{m} \frac{\partial \ell}{\partial y_i} \cdot \widehat{x}_i$$

$$\frac{\partial \ell}{\partial \beta} = \sum_{i=1}^{m} \frac{\partial \ell}{\partial y_i}$$

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

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

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

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

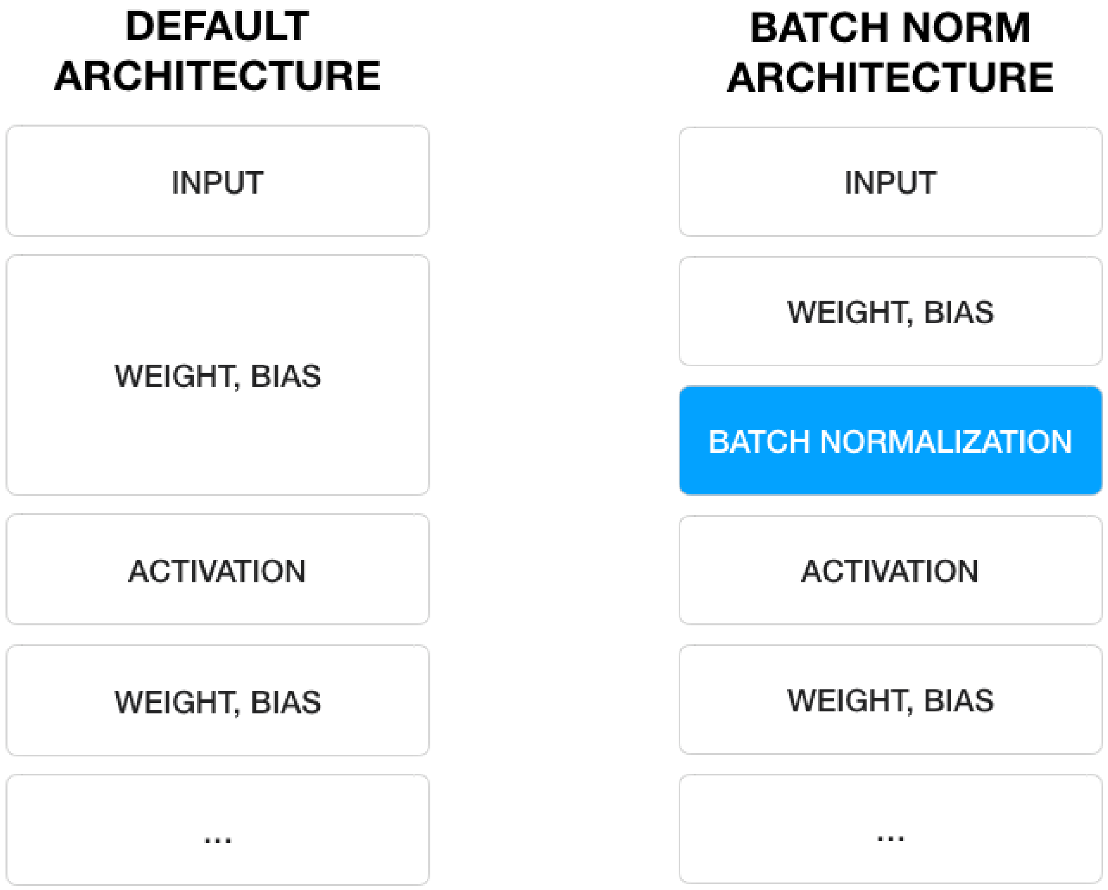
$$\widehat{x}_i \leftarrow \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma_{\mathcal{B}}^2 + \epsilon}} \qquad \text{// normalize}$$

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

Algorithm 1: Batch Normalizing Transform, applied to activation x over a mini-batch.

Batch Normalization

Training and Inference with Batch-Normalized Networks



Input: Network N with trainable parameters Θ; subset of activations $\{x^{(k)}\}_{k=1}^{K}$

Output: Batch-normalized network for inference, $N_{\mathrm{BN}}^{\mathrm{inf}}$

$N_{\mathrm{BN}}^{\mathrm{tr}} \leftarrow N$ // Training BN network
for $k = 1 \ldots K$ **do**
  Add transformation $y^{(k)} = \mathrm{BN}_{\gamma^{(k)},\beta^{(k)}}(x^{(k)})$ to $N_{\mathrm{BN}}^{\mathrm{tr}}$ (Alg. 1)
  Modify each layer in $N_{\mathrm{BN}}^{\mathrm{tr}}$ with input $x^{(k)}$ to take $y^{(k)}$ instead
end for
Train $N_{\mathrm{BN}}^{\mathrm{tr}}$ to optimize the parameters $\Theta \cup \{\gamma^{(k)}, \beta^{(k)}\}_{k=1}^{K}$
$N_{\mathrm{BN}}^{\mathrm{inf}} \leftarrow N_{\mathrm{BN}}^{\mathrm{tr}}$ // Inference BN network with frozen // parameters
for $k = 1 \ldots K$ **do**
  // For clarity, $x \equiv x^{(k)}, \gamma \equiv \gamma^{(k)}, \mu_{\mathcal{B}} \equiv \mu_{\mathcal{B}}^{(k)}$, etc.
  Process multiple training mini-batches $\mathcal{B}$, each of size m, and average over them:

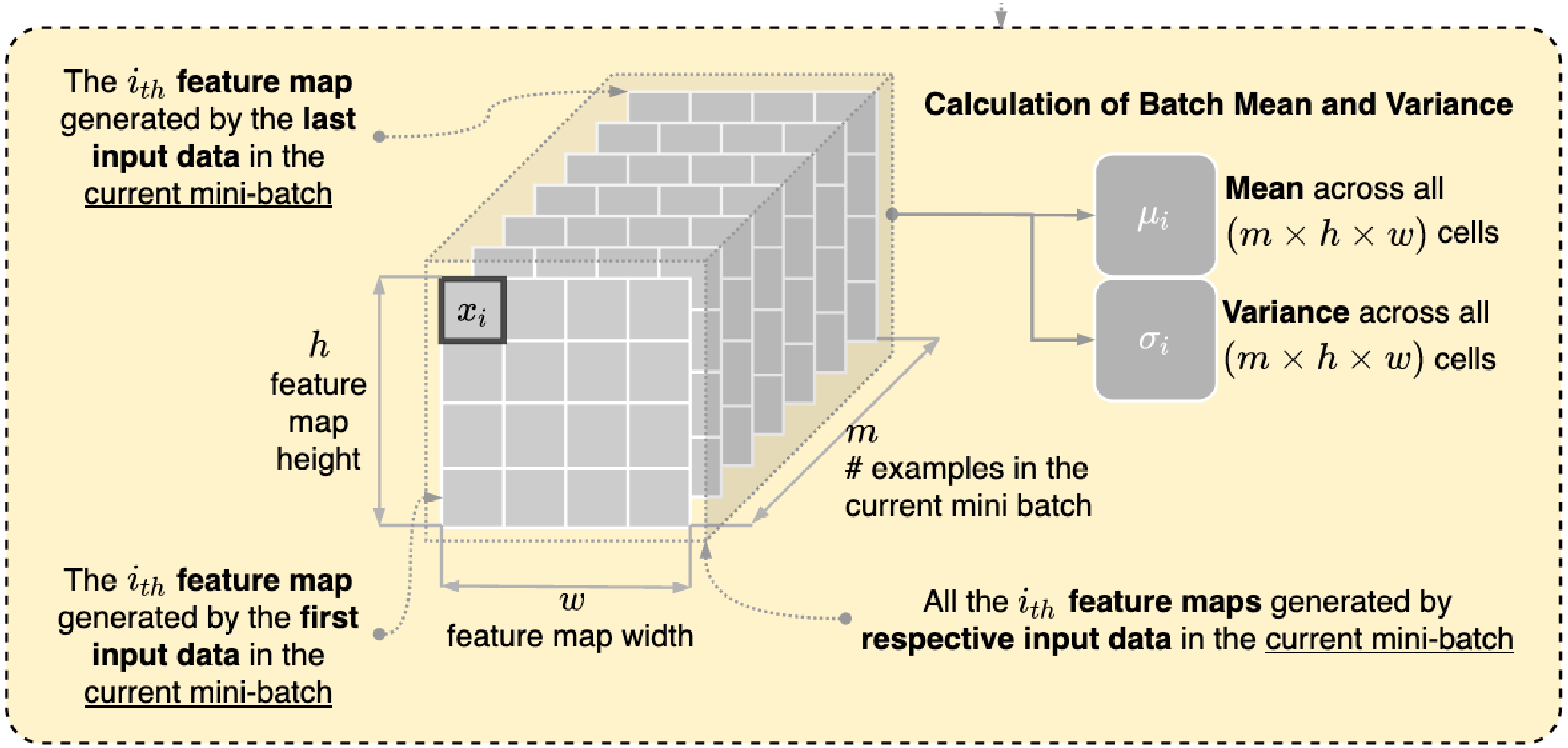
$$\mathrm{E}[x] \leftarrow \mathrm{E}_{\mathcal{B}}[\mu_{\mathcal{B}}]$$
$$\mathrm{Var}[x] \leftarrow \tfrac{m}{m-1}\mathrm{E}_{\mathcal{B}}[\sigma_{\mathcal{B}}^2]$$

  In $N_{\mathrm{BN}}^{\mathrm{inf}}$, replace the transform $y = \mathrm{BN}_{\gamma,\beta}(x)$ with $y = \frac{\gamma}{\sqrt{\mathrm{Var}[x]+\epsilon}} \cdot x + \left(\beta - \frac{\gamma\,\mathrm{E}[x]}{\sqrt{\mathrm{Var}[x]+\epsilon}}\right)$
end for

Algorithm 2: Training a Batch-Normalized Network

Batch Normalization on Convolution

Batch Normalization with Convolution Network



꼭 구현해보자!

To Accelerate Batch Normalization

Advantages

Increase learning rate → 빠른 학습, 수렴

Remove Dropout → Overfitting 방지로서 Batch Normalization

Reduce the L2 weight regularization → Overfitting 방지로서 Batch Normalization

Remove LRN → 효과 없음

Shuffling training examples more thoroughly → mini batch를 사용하기 때문에 같은 batch 안에 데이터가 편향되지 않게 하기 위함

Reduce the photometric distortions → full batch보다 더 적게, 더 빠르게 학습하기 때문에 distorted img보다 real img 사용