



🌙 Type something...

#### 01) Batch Normalization

02\_batchnorm\_forward(x, gamma, beta, bn\_param)

1. Train

2. Test

03\_batchnorm\_backward(dout, cache)

04\_affine\_batchnorm\_forward(x, w, b, gamma, beta, bn\_params)

1. Inline Question 1

2. Inline Question 2

3. Inline Question 3

05\_layer\_norm\_forward(x, gamma, beta, ln\_param)

06\_layer\_norm\_backward(dout, cache)

4. Inline Question 4

## 01) Batch Normalization

### ▼ Batch Normalization

One way to make deep networks easier to train is to use more sophisticated optimization procedures such as SGD+momentum, RMSProp, or Adam. Another strategy is to change the architecture of the network to make it easier to train. One idea along these lines is batch normalization, proposed by [1] in 2015.

To understand the goal of batch normalization, it is important to first recognize that machine learning methods tend to perform better with input data consisting of uncorrelated features with zero mean and unit variance. When training a neural network, we can preprocess the data before feeding it to the network to explicitly decorrelate its features. This will ensure that the first layer of the network sees data that follows a nice distribution. However, even if we preprocess the input data, the activations at deeper layers of the network will likely no longer be decorrelated and will no longer have zero mean or unit variance, since they are output from earlier layers in the network. Even worse, during the training process the distribution of features at each layer of the network will shift as the weights of each layer are updated.

The authors of [1] hypothesize that the shifting distribution of features inside deep neural networks may make training deep networks more difficult. To overcome this problem, they propose to insert into the network layers that normalize batches. At training time, such a layer uses a minibatch of data to estimate the mean and standard deviation of each feature. These estimated means and standard deviations are then used to center and normalize the features of the minibatch. A running average of these means and standard deviations is kept during training, and at test time these running averages are used to center and normalize features.

It is possible that this normalization strategy could reduce the representational power of the network, since it may sometimes be optimal for certain layers to have features that are not zero-mean or unit variance. To this end, the batch normalization layer includes learnable shift and scale parameters for each feature dimension.

- batch normalization을 이해하기 위해서는 평균 0, 분산 1로 구성된 input data가 더 나은 퍼포먼스를 보이는 데에 있다.W
- 이렇게 하면 네트워크의 첫 번째 레이어가 좋은 분포를 따르는 데이터를 보게 된다.
- 그러나 입력 데이터를 전처리하더라도, 네트워크의 깊은 레이어에서 활성화는 더 이상 상관관계가 없거나 평균이 0이거나 분산이 1이 아닐 가능성이 높다.
- 이는 이들 레이어가 네트워크의 이전 레이어에서 출력된 결과이기 때문이다.
- 더 나쁜 것은, 훈련 과정 동안 네트워크의 각 레이어에서 특성의 분포가 각 레이어의 가중치가 업데이트됨에 따라 변화한다는 점이다.W
- 이 문제를 해결하기 위해, 그들은 네트워크에 배치를 정규화하는 레이어를 삽입할 것을 제안한다.
- 훈련 시, 이러한 레이어는 미니배치 데이터를 사용하여 각 특성의 평균과 표준 편차를 추정합니다. 이 추정된 평균과 표준 편차는 미니배치의 특성을 중심에 두고 정규화하는 데 사용됩니다. 훈련 동안 이러한 평균과 표준 편차의 이동 평균이 유지되며, 테스트 시에는 이 이동 평균을 사용하여 특성을 중심에 두고 정규화합니다.
- 이 정규화 전략이 네트워크의 표현력을 감소시킬 가능성도 있다.
- 특정 레이어의 경우 평균이 0이거나 분산이 1이 아닌 특성이 최적일 수 있기 때문이다.
- 이를 위해 배치 정규화 레이어는 각 특성 차원에 대해 학습 가능한 이동 및 스케일 매개변수를 포함합니다. (**beta, gamma**)

<b>Input:</b> Values of $x$ over a mini-batch: $\mathcal{B} = \{x_1 \dots m\}$ ;
Parameters to be learned: $\gamma, \beta$
<b>Output:</b> $\{y_i = BN_{\gamma, \beta}(x_i)\}$
$\mu_B \leftarrow \frac{1}{m} \sum_{i=1}^m x_i \quad // \text{mini-batch mean}$
$\sigma_B^2 \leftarrow \frac{1}{m} \sum_{i=1}^m (x_i - \mu_B)^2 \quad // \text{mini-batch variance}$
$\hat{x}_i \leftarrow \frac{x_i - \mu_B}{\sqrt{\sigma_B^2 + \epsilon}} \quad // \text{normalize}$
$y_i \leftarrow \gamma \hat{x}_i + \beta \equiv BN_{\gamma, \beta}(x_i) \quad // \text{scale and shift}$

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



$$\mu_{\beta} = \frac{1}{m} \sum_{i=1}^m x_i$$

$$\sigma_{\beta}^2 = \frac{1}{m} \sum_{i=1}^m (x_i - \mu_{\beta})^2$$

$$\hat{x}_i = \frac{x_i - \mu_{\beta}}{\sqrt{\sigma_{\beta}^2 + \epsilon}}$$

$$y_i = \gamma \hat{x}_i + \beta = BN_{\gamma, \beta}(x_i)$$

- 이 수식들을 이용해서 forward를 구현할 것이다.

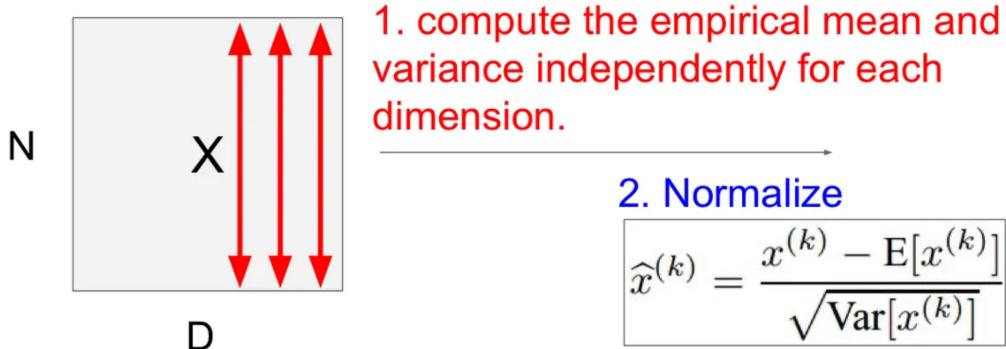
02\_batchnorm\_forward(x, gamma, beta, bn\_param)

1. Train

## Batch Normalization

[Ioffe and Szegedy, 2015]

“you want unit gaussian activations?  
just make them so.”



Fei-Fei Li & Justin Johnson & Serena Yeung

Lecture 6 - 55

April 20, 2017

- 차원별로 평균과 분산을 계산한다. → `axis = 0` 으로 계산하여야 한다.

```
sample_mean = np.mean(x, axis = 0) # feature들의 평균 구하기 200 x 3 → (3, )
sample_var = np.mean((x - sample_mean) ** 2, axis = 0) # (3, )
out = (x - sample_mean) / np.sqrt(sample_var + eps) # (200, 3)
out = gamma * out + beta
cache = (x, sample_mean, sample_var, gamma)

# store for inference
running_mean = momentum * running_mean + (1 - momentum) * sample_mean
running_var = momentum * running_var + (1 - momentum) * sample_var
```

- `sample_mean` 과 `sample_var` 를 구해주고 `out`에다가 저장해준다.
- `cache` 에다가는 `(x, sample_mean, sample_var, gamma)` 값을 저장해준다. 추후에 `backward`에서 사용할 예정이다.

2. Test

- inference 시에 입력되는 값을 통해서 정규화를 하게 되면 모델이 학습을 통해서 입력 데이터의 분포를 추정하는 의미 자체가 없어지게 된다.
- 즉, inference 에서는 결과를 결정하기 위하여 고정된 평균과 분산을 이용하여 정규화를 수행하게 된다.

- 6: Train  $N_{BN}^{tr}$  to optimize the parameters  $\Theta \cup \{\gamma^{(k)}, \beta^{(k)}\}_{k=1}^K$
- 7:  $N_{BN}^{inf} \leftarrow N_{BN}^{tr}$  // Inference BN network with frozen  
// parameters



8: **for**  $k = 1 \dots K$  **do**  
 9:     // For clarity,  $x \equiv x^{(k)}$ ,  $\gamma \equiv \gamma^{(k)}$ ,  $\mu_{\mathcal{B}} \equiv \mu_{\mathcal{B}}^{(k)}$ , etc.  
 10:    Process multiple training mini-batches  $\mathcal{B}$ , each of size  $m$ , and average over them:

$$\begin{aligned} E[x] &\leftarrow E_{\mathcal{B}}[\mu_{\mathcal{B}}] \\ \text{Var}[x] &\leftarrow \frac{m}{m-1} E_{\mathcal{B}}[\sigma_{\mathcal{B}}^2] \end{aligned}$$

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

- train과 방식은 똑같지만  $\text{mean} = \text{running\_mean}$  &  $\text{var} = \text{running\_var}$  을 사용한다는 점이 약간 다르다.

03\_batchnorm\_backward(dout, cache)

$$\begin{aligned} \frac{\partial \ell}{\partial \hat{x}_i} &= \frac{\partial \ell}{\partial y_i} \cdot \gamma \\ \frac{\partial \ell}{\partial \sigma_{\mathcal{B}}^2} &= \sum_{i=1}^m \frac{\partial \ell}{\partial \hat{x}_i} \cdot (x_i - \mu_{\mathcal{B}}) \cdot \frac{-1}{2} (\sigma_{\mathcal{B}}^2 + \epsilon)^{-3/2} \\ \frac{\partial \ell}{\partial \mu_{\mathcal{B}}} &= \left( \sum_{i=1}^m \frac{\partial \ell}{\partial \hat{x}_i} \cdot \frac{-1}{\sqrt{\sigma_{\mathcal{B}}^2 + \epsilon}} \right) + \frac{\partial \ell}{\partial \sigma_{\mathcal{B}}^2} \cdot \frac{\sum_{i=1}^m -2(x_i - \mu_{\mathcal{B}})}{m} \\ \frac{\partial \ell}{\partial x_i} &= \frac{\partial \ell}{\partial \hat{x}_i} \cdot \frac{1}{\sqrt{\sigma_{\mathcal{B}}^2 + \epsilon}} + \frac{\partial \ell}{\partial \sigma_{\mathcal{B}}^2} \cdot \frac{2(x_i - \mu_{\mathcal{B}})}{m} + \frac{\partial \ell}{\partial \mu_{\mathcal{B}}} \cdot \frac{1}{m} \\ \frac{\partial \ell}{\partial \gamma} &= \sum_{i=1}^m \frac{\partial \ell}{\partial y_i} \cdot \hat{x}_i \\ \frac{\partial \ell}{\partial \beta} &= \sum_{i=1}^m \frac{\partial \ell}{\partial y_i} \end{aligned}$$

- backpropagation은 논문을 참고하여 다음과 같은 식을 유도할 수 있었다.

```
eps = 1e-5
N = dout.shape[0]
x, sample_mean, sample_var, gamma = cache
x_normal = (x - sample_mean) / np.sqrt(sample_var + eps)
dgamma = np.sum(dout * x_normal, axis = 0)
dbeta = np.sum(dout, axis = 0)
dx_normal = dout * gamma
dvar = np.sum(dx_normal * (x - sample_mean) * -0.5 * (sample_var + eps)**-1.5, axis = 0)
dmean = np.sum(dx_normal * -1 / np.sqrt(sample_var + eps), axis = 0)
+ dvar * np.sum(-2 * (x - sample_mean), axis = 0) / N
dx = dx_normal * 1 / np.sqrt(sample_var + eps) + dvar * 2 * (x - sample_mean) / N + dmean / N
```



- 논문을 보고 그대로 코드를 작성하였다.
- computational graph를 통해서 유도할 수 있다.
- 추후에 논문을 읽어보면서 다시 한번 정리해보겠다.
- alternative backward pass는 이해하지 못하여 추후 시간이 되면 다시 진행해보겠다.

#### 04\_affine\_batchnorm\_forward(x, w, b, gamma, beta, bn\_params)

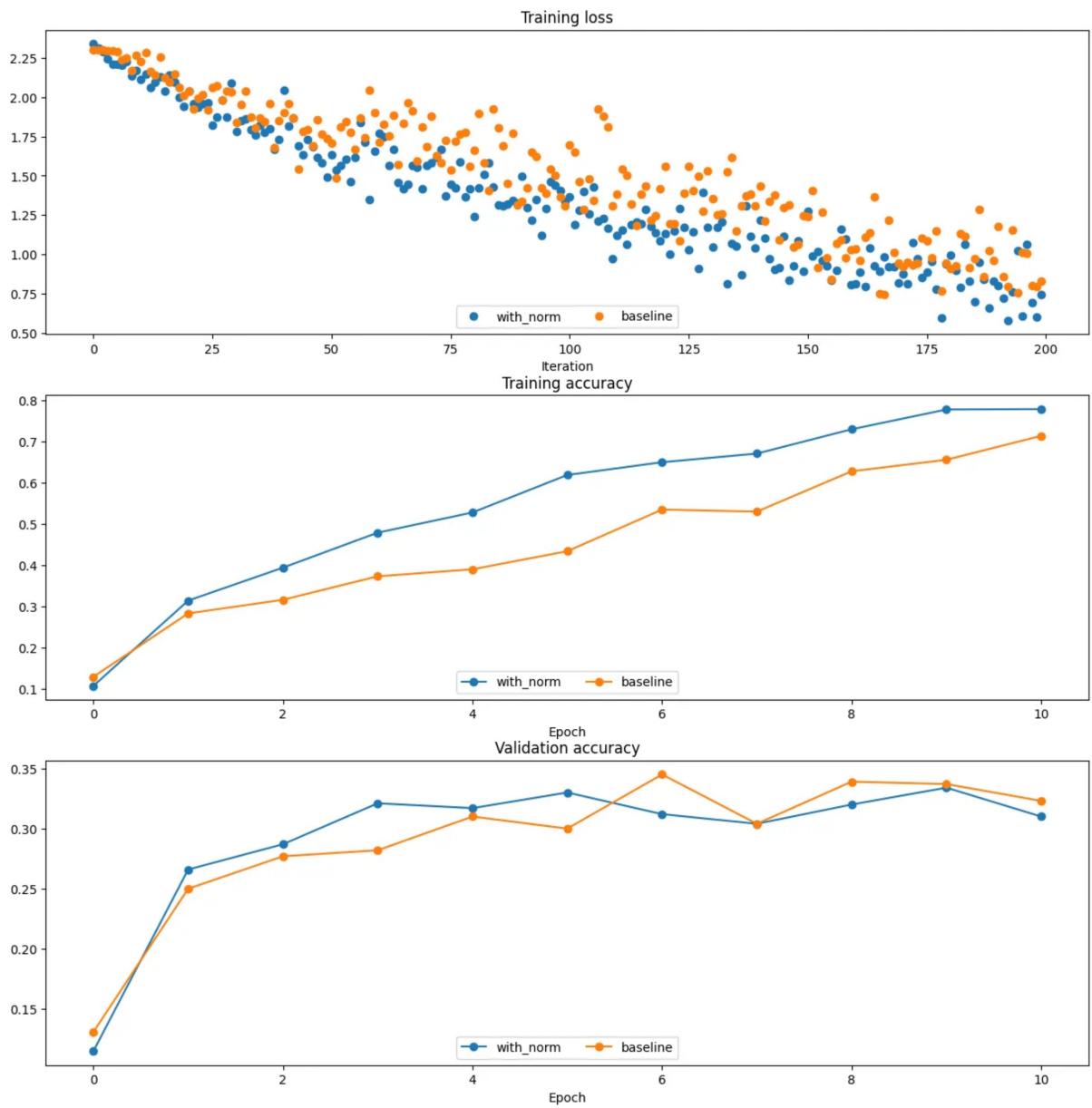
```
def affine_batchnorm_forward(x, w, b, gamma, beta, bn_params):
    a, fc_cache = affine_forward(x, w, b)
    b, batch_cache = batchnorm_forward(a, gamma, beta, bn_params)
    out, relu_cache = relu_forward(b)

    cache = (fc_cache, batch_cache, relu_cache)

    return out, cache
```

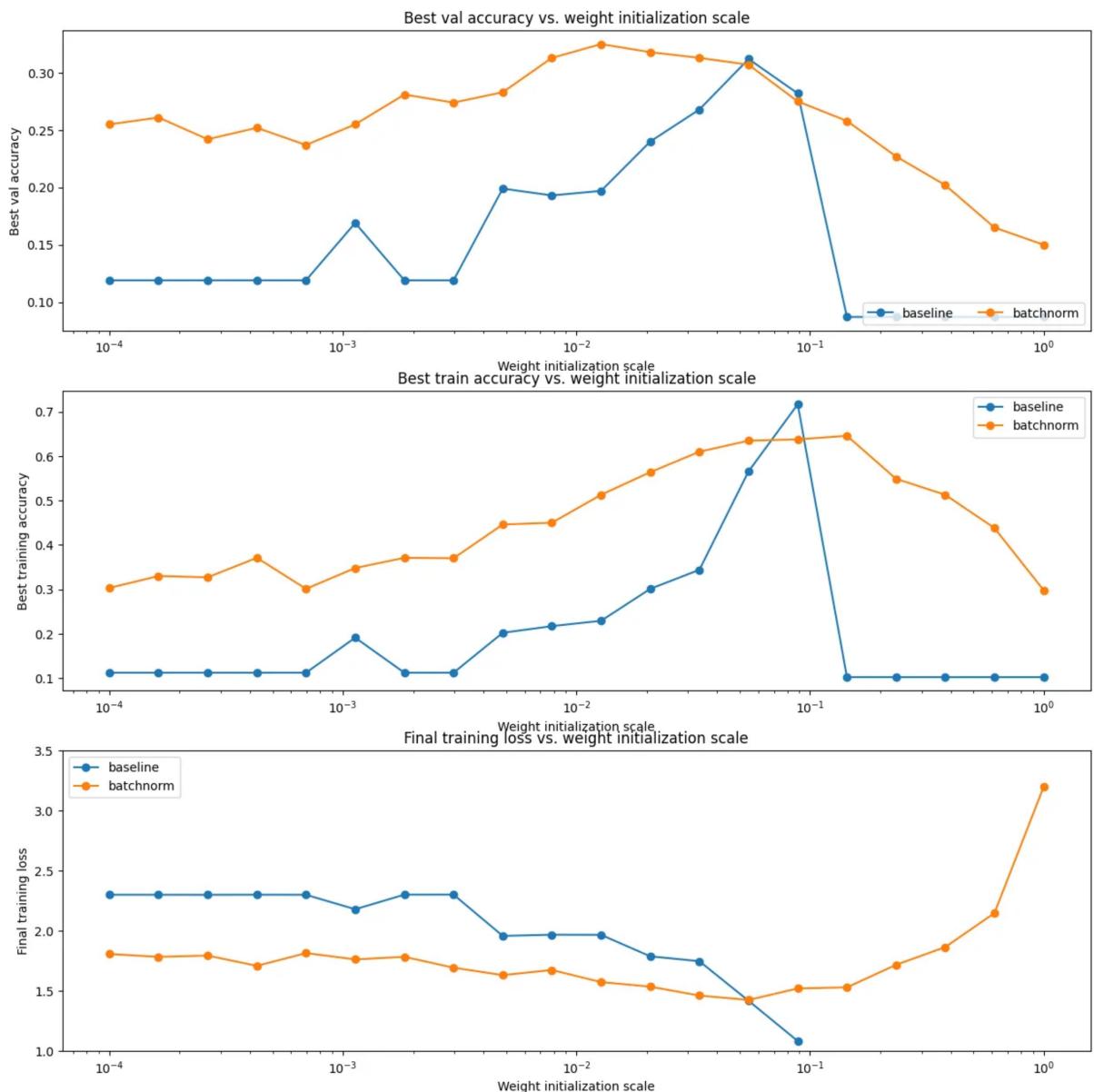
- 편의를 위해 affine + batchnorm + relu를 합친 layer를 만들어주었다.

- 지금까지 만들었던 모든 기능들이 잘 동작하는지 1000 training example를 가지고 실험해보자.



- normalization을 사용하는 것이 성능이 더 좋게 나온다는 사실을 알게 되었다.





#### 1. Inline Question 1

##### Inline Question 1:

Describe the results of this experiment. How does the weight initialization scale affect models with/without batch normalization differently, and why?

Answer:

[FILL THIS IN]

- batch normalization을 사용하지 않은 경우에는 최적의 weight initialization scale이 아니면 좋지 않은 결과를 나타냈다.
- 하지만 batch normalization을 사용하는 경우에는 어떤 경우에는 무난한 결과를 보여준다.

#### 2. Inline Question 2



## Inline Question 2:

Describe the results of this experiment. What does this imply about the relationship between batch normalization and batch size? Why is this relationship observed?

### Answer:

[FILL THIS IN]

- 만약 batch size가 너무 작으면, 해당 batch의 데이터들이 모든 데이터를 표방하는 데이터가 아닐 수 있다.
- batch size 작으면 대표성이 줄어들어 parameter를 이상한 방향으로 업데이트할 것이다.

### 3. Inline Question 3

## Inline Question 3:

Which of these data preprocessing steps is analogous to batch normalization, and which is analogous to layer normalization?

- Scaling each image in the dataset, so that the RGB channels for each row of pixels within an image sums up to 1.
- Scaling each image in the dataset, so that the RGB channels for all pixels within an image sums up to 1.
- Subtracting the mean image of the dataset from each image in the dataset.
- Setting all RGB values to either 0 or 1 depending on a given threshold.

- 이미지의 모든 데이터들의 RGB채널의 각 픽셀을 모두 더했을때 1이 되게 조정
- 각 이미지의 RGB채널의 모든 픽셀을 모두 더했을때 1이 되게 조정
- 모든 데이터의 평균 이미지를 각 이미지에서 뺀 경우
- 모든 RGB 값들을 구분에 따라 0과 1로 조정

A: LN: 1, BN: 2, 3

### 05\_layernorm\_forward(x, gamma, beta, ln\_param)

- Layer Normalization은 data sample 단위로 평균과 표준편차를 계산한다.
- Train 모드와 Test 모드에서 똑같이 동작한다.
- RNN에서 batch normalization을 사용할 수 없는 것을 대체한다.

#### Batch Normalization 수식과의 차이점

- Layer Normalization에서는 같은 layer에 있는 모든 은닉층 유닛들이 정규화 통계량( $\mu, \sigma$ )을 공유한다
- 하지만 배치의 각 훈련 샘플은 서로 다른 정규화 통계량을 갖는다
- mini-batch의 크기에 영향을 받지 않기 때문에 batch size 1의 온라인 학습에도 이용될 수 있다

```
sample_mean = np.mean(x, axis = 1).reshape(-1, 1) # feature들의 평균 구하기 (4, 1)
sample_var = np.var(x, axis = 1).reshape(-1, 1) # (4, 1)#
x_normal = (x - sample_mean) / np.sqrt(sample_var + eps) # (4, 3)
out = gamma * x_normal + beta # 4 x 3
cache = (x, x_normal, sample_mean, sample_var, gamma)
```

- axis = 1로 바꿔준 후 연산을 위해 `reshape` 를 사용하였다.

### 06\_layernorm\_backward(dout, cache)



```

eps = 1e-5
N = dout.shape[1]
x, x_normal, feature_mean, feature_var, gamma = cache
x_normal = (x - feature_mean) / np.sqrt(feature_var + eps)
dgamma = np.sum(dout * x_normal, axis = 0)
dbeta = np.sum(dout, axis = 0)
dx_normal = dout * gamma

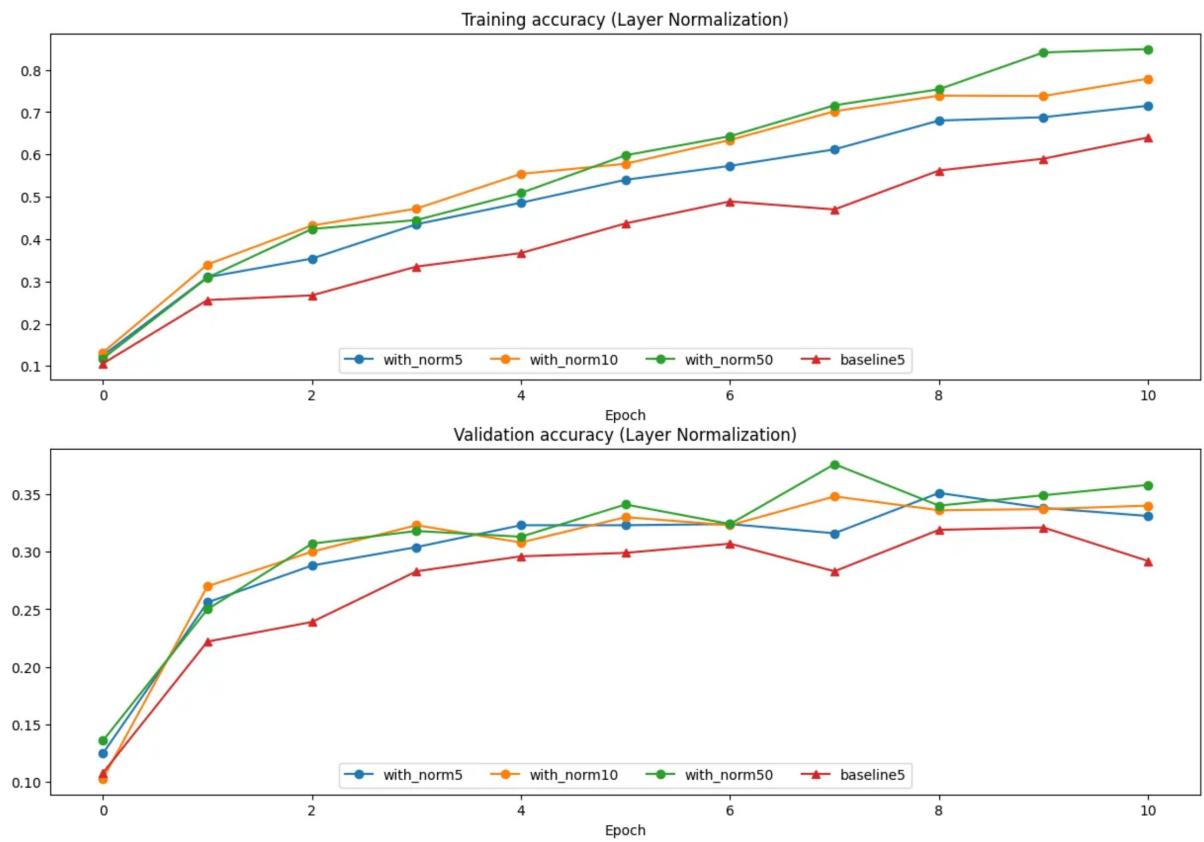
dlvar = np.sum(dx_normal * (x - feature_mean)
* -0.5 * (feature_var + eps)**-1.5, axis = 1).reshape(-1, 1)

dlmean = np.sum(dx_normal * -1 / np.sqrt(feature_var + eps), axis = 1).reshape(-1, 1)
+ dlvar * np.sum(-2 * (x - feature_mean), axis = 1).reshape(-1, 1) / N

dx = dx_normal * 1 / np.sqrt(feature_var + eps) + dlvar * 2 * (x - feature_mean) / N + dlmean / N

```

- batch\_normalization과 일반적으로는 똑같다.
- size를 맞추기 위해 `dgamma` 와 `dbeta` 는 동일하게 했다.
- `axis = 1`로 바꾸기 때문에 `reshape(-1,1)` 을 사용해서 size를 맞춰준다.



#### 4. Inline Question 4

### Inline Question 4:

When is layer normalization likely to not work well, and why?

1. Using it in a very deep network
2. Having a very small dimension of features
3. Having a high regularization term

A: 2. batch normalization과 비슷하게 feature의 차원이 매우 작게 되면 sample들의 대표성이 떨어지게 되어 성능이 떨어진다.

3. LN 자체가 이미 Regularization이기 때문에 여기서 더 높은 regularization term을 주게 되면 나쁜 퍼포먼스가 나오게 된다.

