Layer Normalization

Jimmy Lei Ba, Jamie Ryan Kiros, Geoffrey E.Hinton

정규헌

1.Introduction

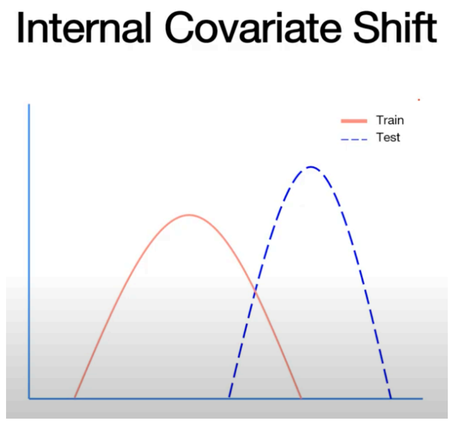
- **DNN**(Deep Neural Network)은 다양한 Supervised learning 문제 에서 좋은 성능을 보이고 있음
- DNN을 훈련 시키는 것에는 종종 **시간적 비용**이 많이 듬 : 어떻게 단축?
  - 병렬 접근 방식 : 계산이나 신경망 자체를 여러 머신에 분산시켜 학습속도를 증가
  - 직렬 접근 방식 : 신경망의 순전파 과정 연산을 보다 쉽게 학습할 수 있도록 수정

1.Introduction

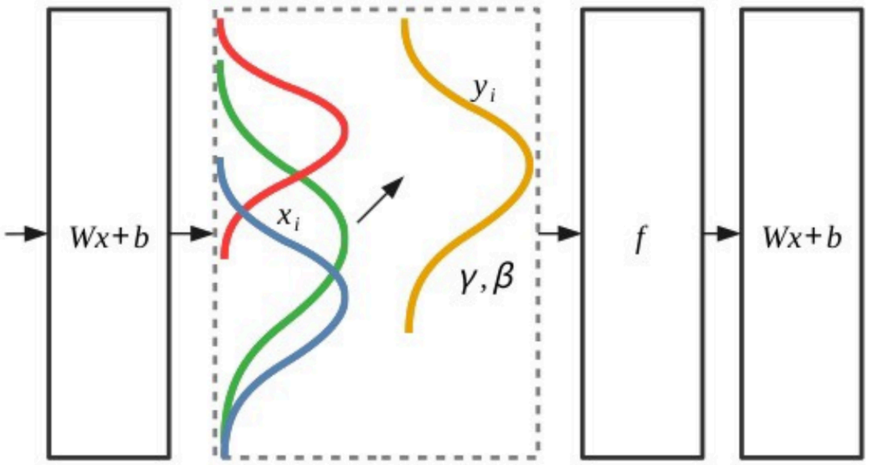
- **병렬 접근 방식** : 데이터나 신경망 자체를 여러 머신에 분산시켜 학습속도를 증가
  - 많은 통신, 복잡한 소프트웨어를 필요로 할 수 있음
  - 병렬의 정도가 증가할 수록 급격하게 return이 감소하는 경향이 있음
- **직렬 접근 방식** : 신경망의 연산과정을 보다 쉽게 학습할 수 있도록 수정
  - 신경망에서 추가 normalization단계를 포함해 학습하는 **batch normalization** 이 사용됨

1-2. Internal Covariate Shift

- Covariate shift : 각 입력층에서 학습 진행할 때 마다 출력층의 분포가 변화
  - Gradient Vanishing, Exploding : 학습 gradient가 너무 작아지거나, 커져서 DNN학습에 문제가 생김
  - 층이 깊어질수록 더욱 영향이 커짐
  - 학습시에 각 층이 입력분포 a를 전제로 학습했다면 실제로 test시에는 다른 분포에 맞게 학습이 되어 엉뚱한 결과를 낼 수 있음
- Normalization : 학습데이터의 평균과 표준 편차를 사용하여 각 합계를 표준화(평균 0, 분산1)
  - 각 단계에서 분포가 비슷하기 때문에 weight를 학습할 때 더 수월함
  - Batch Normalization(BN)(Sergey Ioffe)
  - Weight Normalization(WN)(Tim Salimans)
  - Layer Normalization(LN)(Jimmy Lei Ba)



$$\frac{X-\mu}{\sigma}$$



2. Batch Normalization

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

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

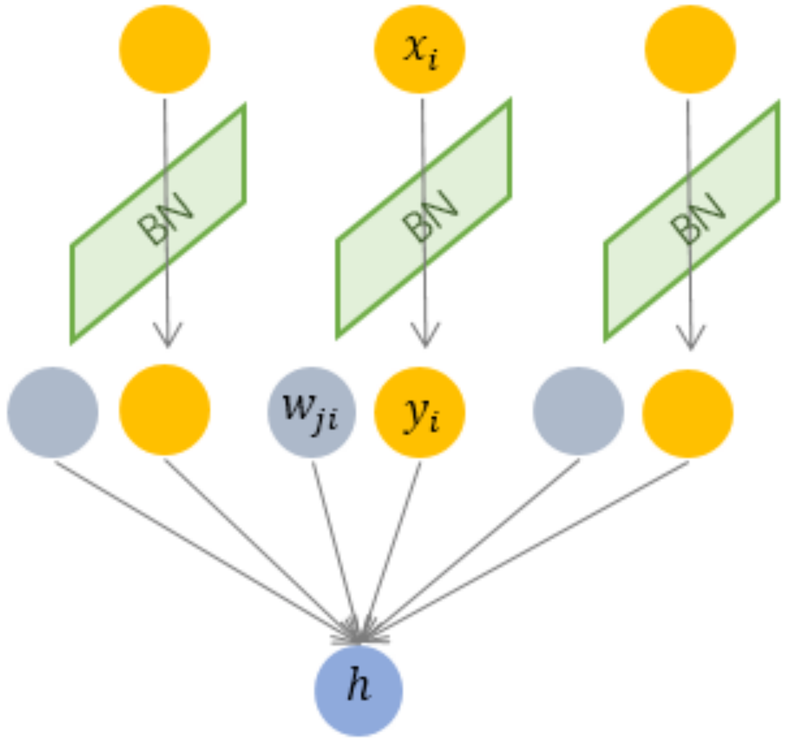
$$\mu_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m} x_i \qquad \text{// mini-batch mean}$$

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

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

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

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



2. Batch Normalization

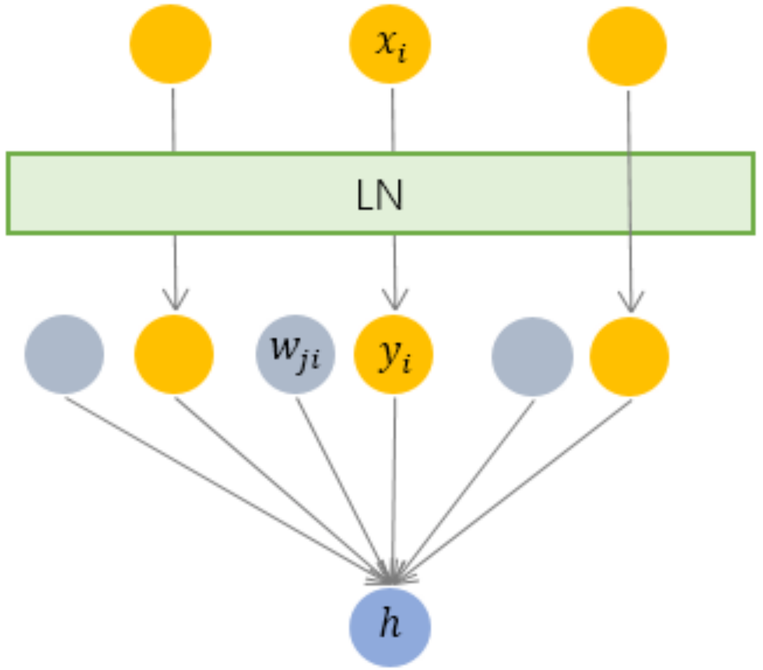
- 각 뉴런의 **Batch** 별로 normalization을 실행
- 단순하면서도 **학습 속도를 증가**시키고, 학습을 안정화 시킴
- Dropout과 같은 **regulation 효과**를 가질수 있음
- 한계
  - Batch size에 영향을 받음(너무 작은 mini-batch의 경우 적용이 잘 안됨) - hyperparameter setting의 영향
  - 총 입력의 통계(평균,표준편차)이 필요함
    - 순방향 네트워크에서는 각각의 layer마다 통계를 저장하는것이 간단함
    - 순방향이 아닌 RNN에서는 적용하기가 어려움
  - train과 test를 할때 연산이 바뀜

3. Layer Normalization

$$\mu^l = \frac{1}{H}\sum_{i=1}^{H} a_i^l \qquad \sigma^l = \sqrt{\frac{1}{H}\sum_{i=1}^{H}\left(a_i^l - \mu^l\right)^2}$$

H : 뉴런의 수

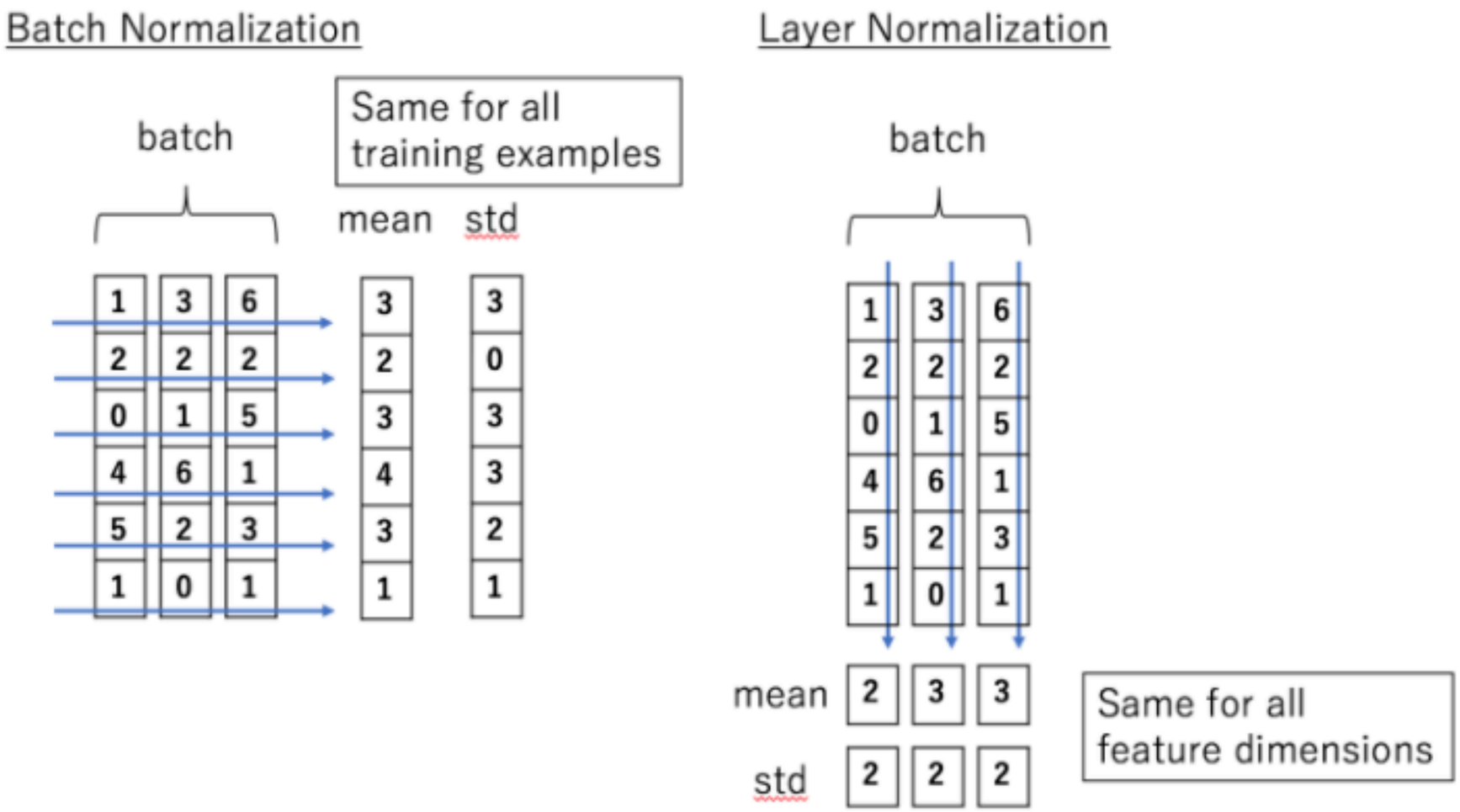
$$\mathbf{h}^t = f\left[\frac{\mathbf{g}}{\sigma^t} \odot \left(\mathbf{a}^t - \mu^t\right) + \mathbf{b}\right] \qquad \mu^t = \frac{1}{H}\sum_{i=1}^{H} a_i^t \qquad \sigma^t = \sqrt{\frac{1}{H}\sum_{i=1}^{H}\left(a_i^t - \mu^t\right)^2}$$



3. Layer Normalization

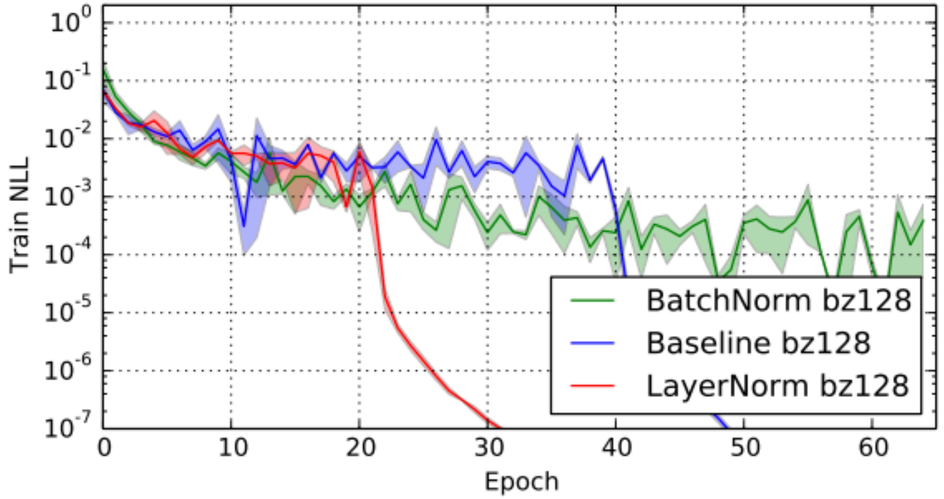
- 각 층의 **뉴런**마다 Normalization : **BN**의 batch size를 뉴런으로 바꾼것
- Batch Normalization과 마찬가지로 학습 속도를 증가시키고, 학습을 안정화 시킴
- **Batch size**에 의존하지 않음
- **RNN**에 적용하기 용이
- CNN에 적용했을 시에는 Batch Normalization보다 성능이 좋지않음
  - **Weight Normalization** : 가중치를 Normalization, CNN에서 효과적

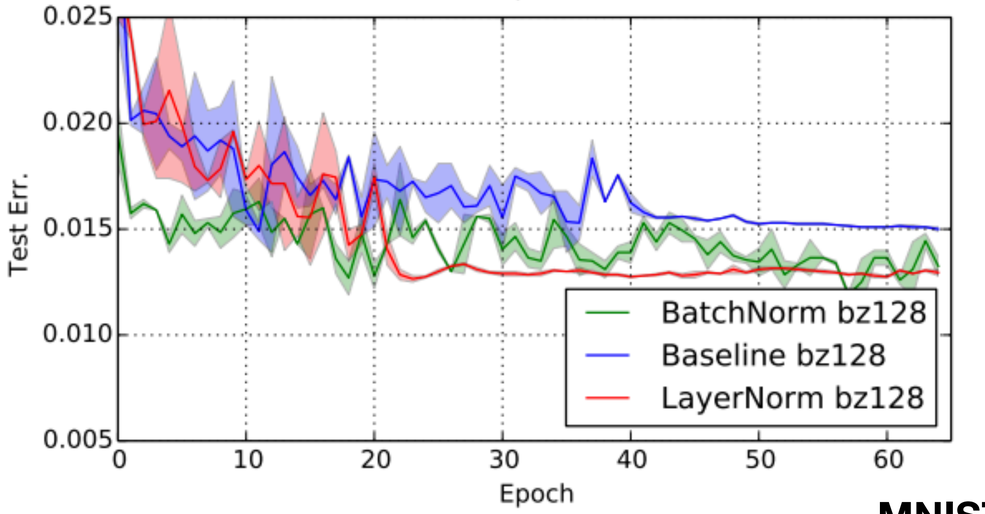
3. Batch Normalization vs Layer Normalization



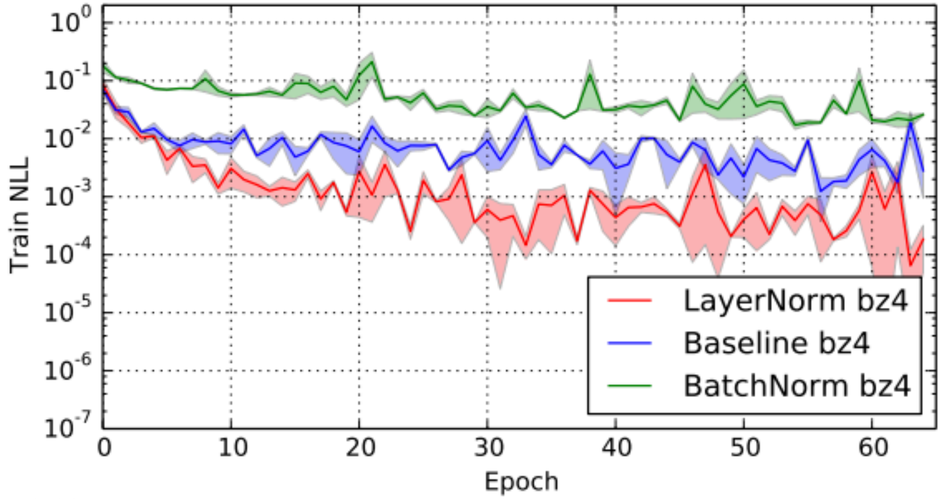
4. Experiment & Result

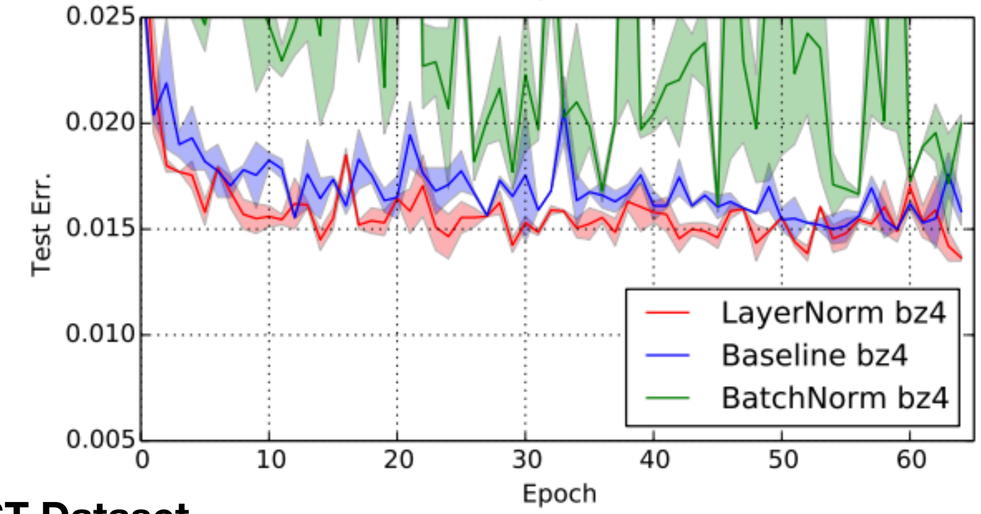
Batch size = 128





Batch size = 4





MNIST Dataset

5. Conclusion

- DNN에서 학습시간을 단축하기 위한 직렬적 접근방식 : Batch Normalization
- Batch Normalization : 몇가지 단점이 존재
- Layer Normalization으로 해결!