Demystifying Batch Normalization: Analysis of Normalizing Layer Inputs in Neural Networks

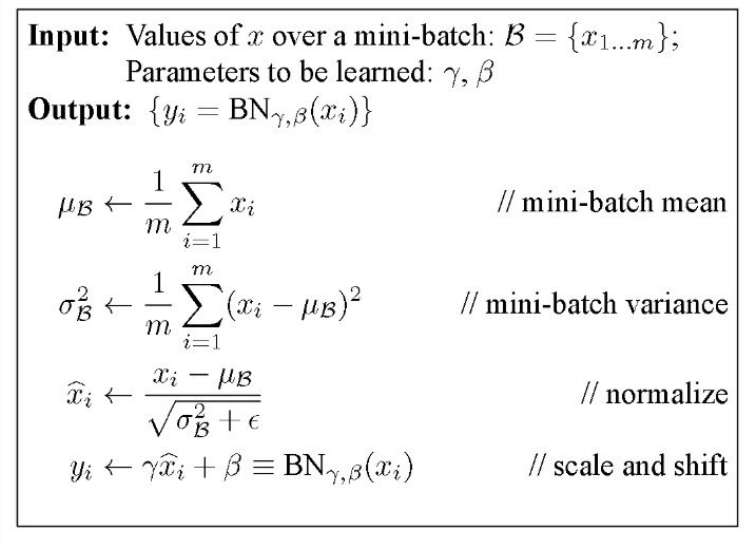
Dinko Franceschi (df2603), Jun Hyek Jang (jj2883), Rahul Vallivel Subbiah (rs3871)
Professor Predrag R. Jelenković
Mathematics of Deep Learning, Columbia University
May 14, 2019

Introduction to Batch Normalization

- Ioffe, S., & Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift
  - Novel mechanism for dramatically accelerating the training of deep networks
  - Solution to “internal covariate shift”
  - Result: better performance than the best known system on ImageNet (4.8% test error)
- Batch Normalization (BN) -> ReLu

Introduction to Batch Normalization

- View 1: Batch Normalization -> ReLu (Ioffe, S., & Szegedy, C. (2015))
- View 2: ReLu -> Batch Normalization (Francois Chollet, Keras author)
- View 3: “There is no difference” (Ian Goodfellow)



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

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

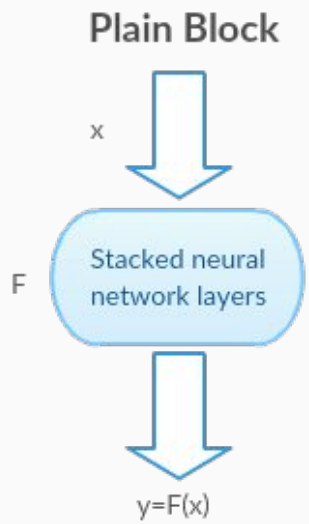
$$\mu_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m} x_i \qquad \text{// mini-batch mean}$$

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

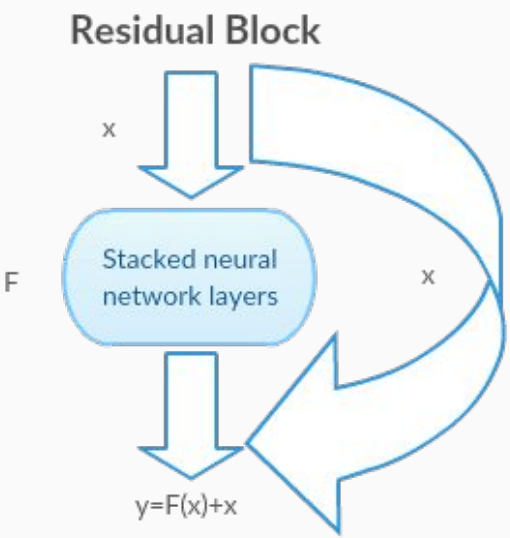
$$\widehat{x}_i \leftarrow \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma^2_{\mathcal{B}} + \epsilon}} \qquad \text{// normalize}$$

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

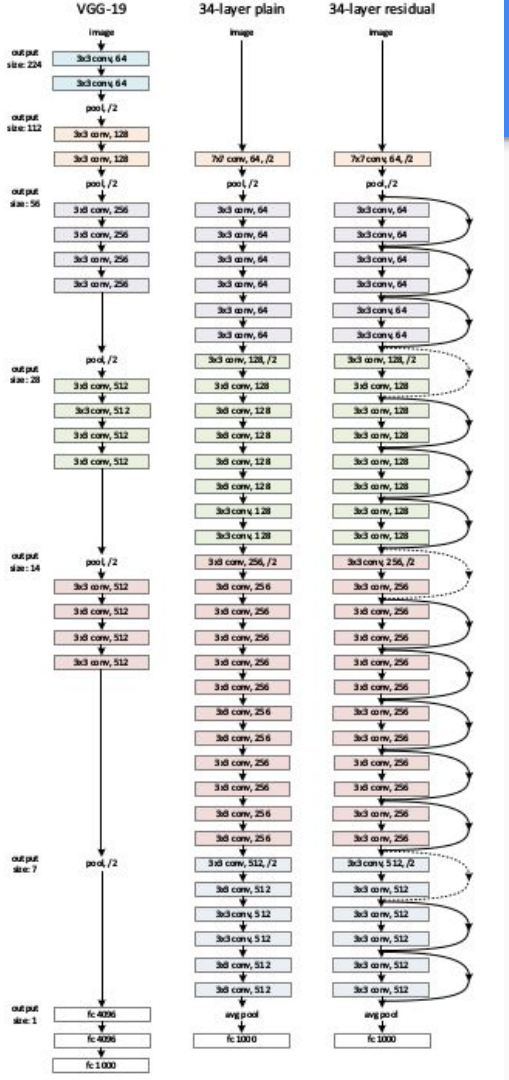
Resnet architecture



Hard to get F(x)=x and make y=x an identity mapping



Easy to get F(x)=0 and make y=x an identity mapping



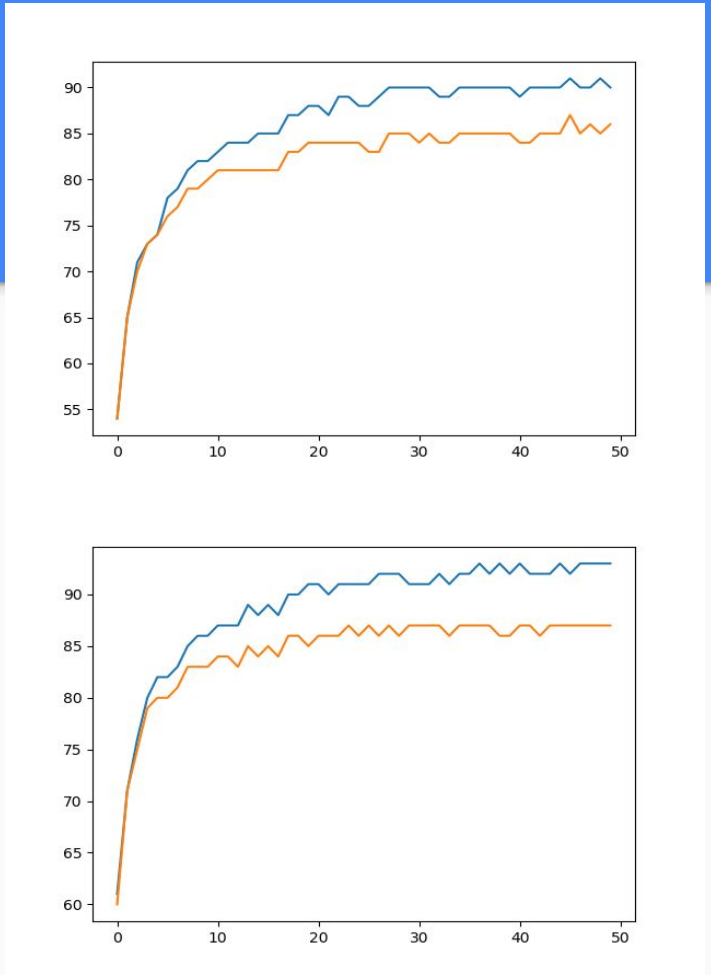
Experiments

- We performed experiments on CNN and Resnet architectures.
- Ran simulations for both the order Batch Normalization->ReLu and ReLU->Batch Normalization and analyzed the results
- Examined the gradients while training for each layer to gain further insight

Results: CNN

- BN->ReLu:
  - Max train accuracy- 91%
  - Max test accuracy- 86%
  - Loss at the end of training: 0.271911

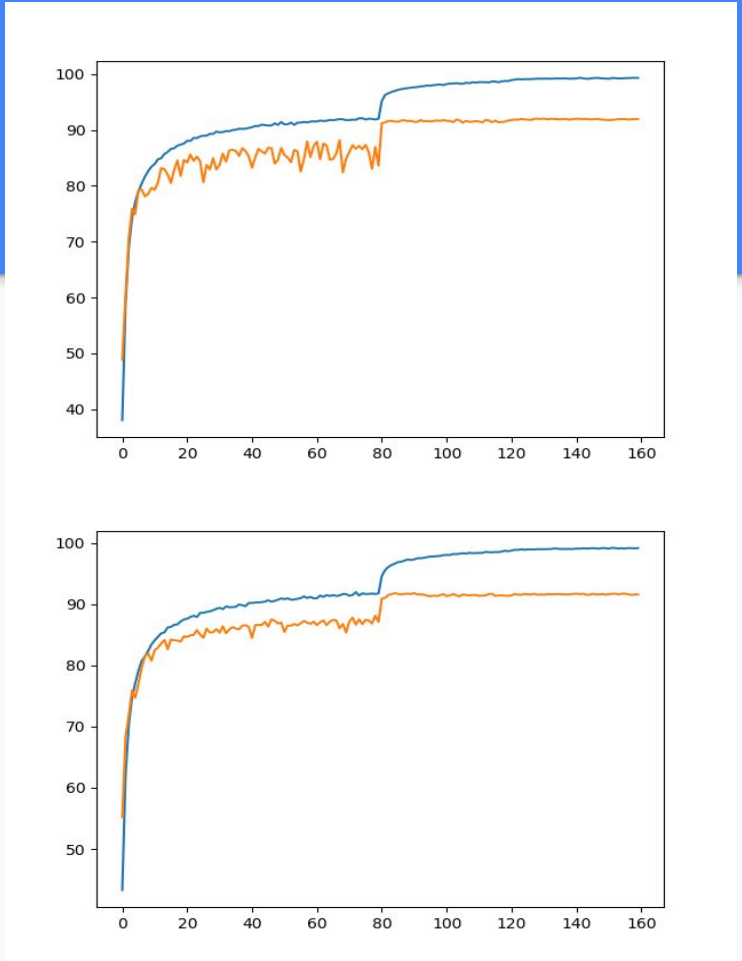
- ReLu->BN:
  - Max train accuracy- 93%
  - Max test accuracy- 87%
  - Loss at the end of training: 0.2120195



Results: Resnet

- BN->ReLu:
  - Max train accuracy- 98%
  - Max test accuracy- 90%
  - Loss at the end of training: 0.1632

- ReLu->BN:
  - Max train accuracy- 98%
  - Max test accuracy- 90%
  - Loss at the end of training: 0.1627

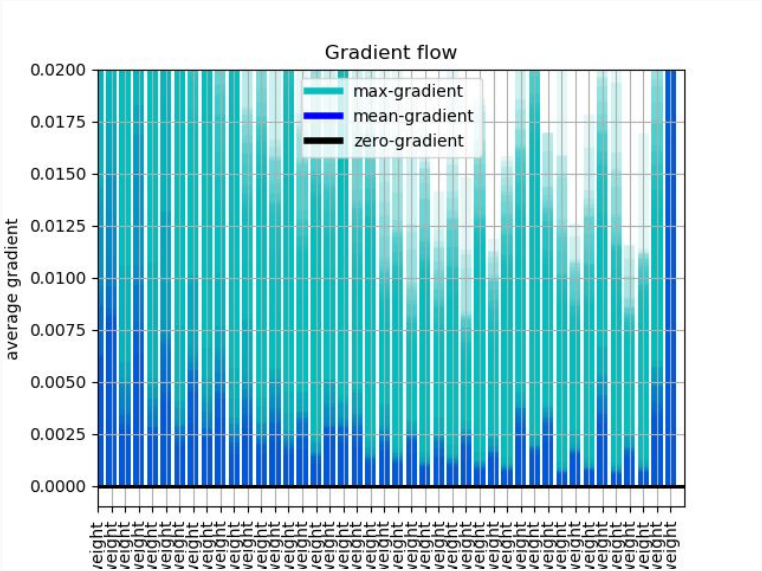


Results

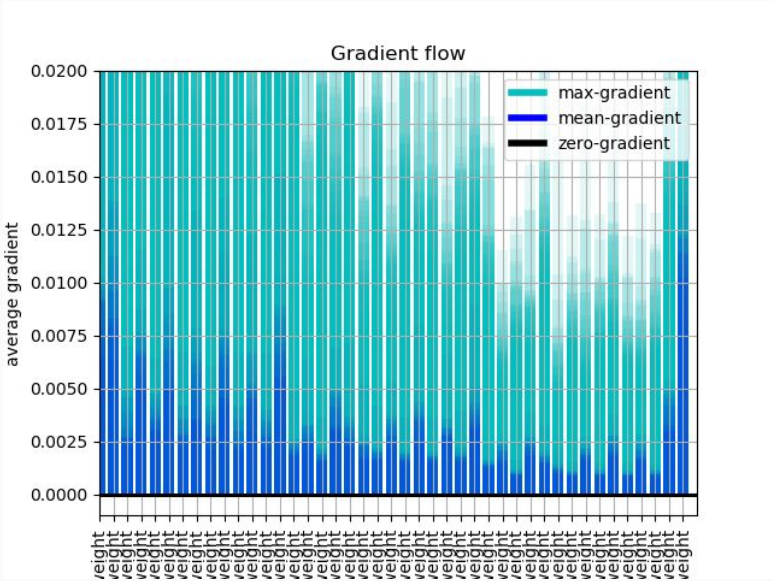
Why this behaviour?

- To explain our results we looked into the gradients while training for each layer. The results below are for Resnet

BN->ReLu

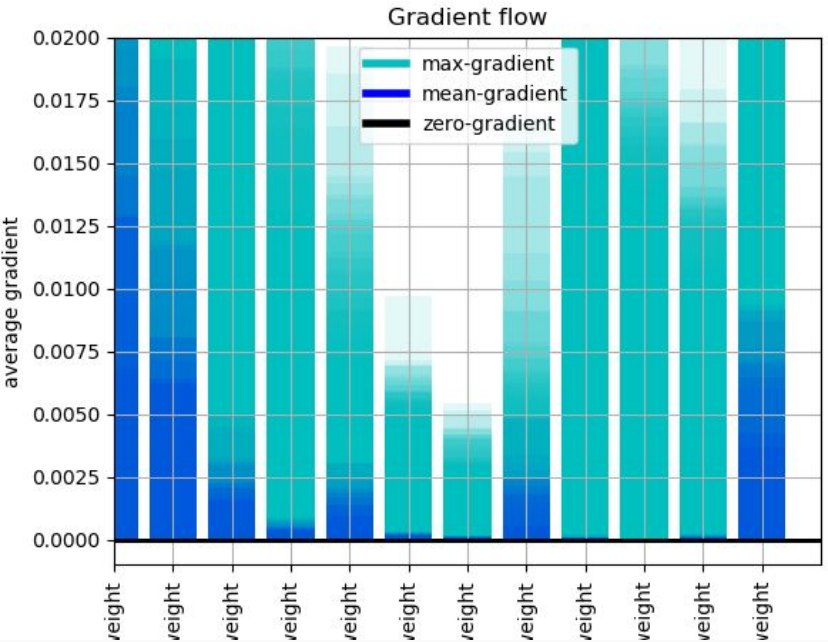


ReLu->BN

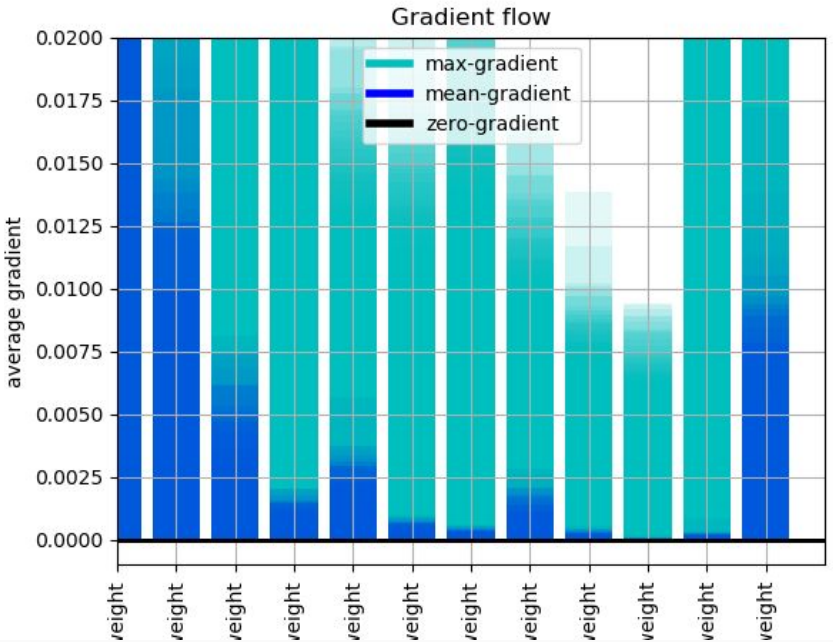


Gradients for CNN

BN->ReLu



ReLu->BN



Inference from the experiments

- Our experiments clearly show that in Convolutional Neural Networks there is an improvement in performance when we switch the order of BN -> ReLu to ReLu->BN.
- This effect was not observed in Residual Nets which have skip connections.
- Examination of gradients shows that in CNN there is an improvement in gradient flow when the ordering is ReLu->BN.
- There is no improvement in gradients for Residual Nets.

Backpropagation of Batch Normalization Overview

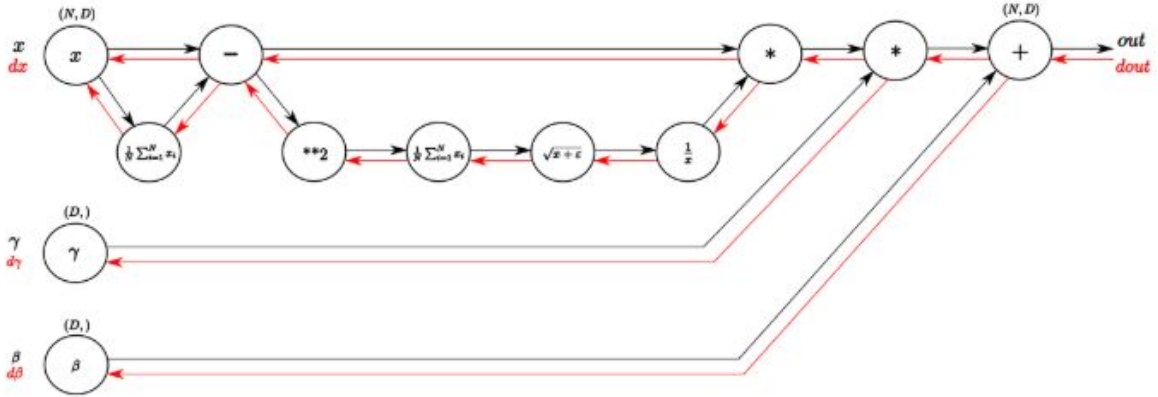
Input: Values of x over a mini-batch: $\mathcal{B} = \{x_{1...m}\}$;
Parameters to be learned: γ, β
Output: $\{y_i = \mathrm{BN}_{\gamma,\beta}(x_i)\}$

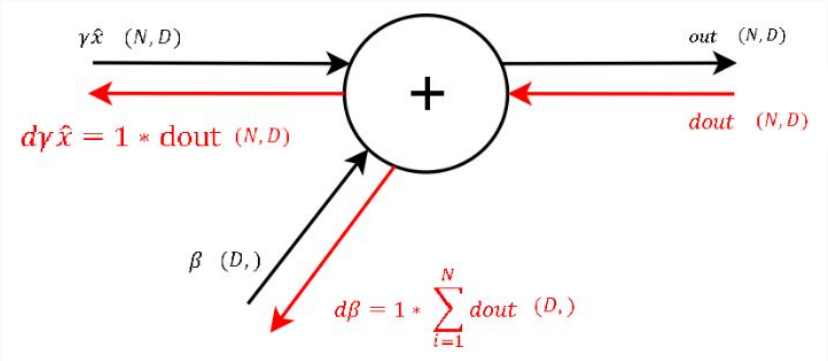
$$\mu_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m} x_i \qquad \text{// mini-batch mean}$$

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

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

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





$$BN = \gamma\hat{x} + \beta$$

$$\frac{\partial Out}{\partial \beta} = \frac{\partial Out}{\partial BN}\frac{\partial BN}{\partial \beta} = \frac{\partial Out}{\partial BN}$$

$$\frac{\partial Out}{\partial \gamma} = \frac{\partial Out}{\partial BN}\frac{\partial BN}{\partial \gamma} = \frac{\partial Out}{\partial BN}\hat{x}$$

Backpropagation of BN/ReLU and ReLU/BN

- BN->ReLu:
  - Sparse Gradient
  - Semi-Definite Positive Matrix

$$\frac{\partial Out}{\partial \beta} = \frac{\partial Out}{\partial BN}$$

$$\sim \frac{\partial ReLU_i}{\partial BN} = \begin{bmatrix} 0 & 0.2 & \dots \\ 1 & 0.6 & \dots \\ \vdots & \vdots & \vdots \end{bmatrix}$$

$$\frac{\partial Out}{\partial \gamma} = \frac{\partial Out}{\partial BN}\hat{x}$$

$$\sim \frac{\partial ReLU_i}{\partial BN} \cdot \hat{x} = \begin{bmatrix} 0 & 0.2 & \dots \\ 0 & 0.6 & \dots \\ \vdots & \vdots & \vdots \end{bmatrix} \cdot \hat{x} = \begin{bmatrix} 0 & 0.2 & \dots \\ 0 & -0.1 & \dots \\ \vdots & \vdots & \vdots \end{bmatrix}$$

- ReLu->BN:
  - More Non-zero gradient

$$\frac{\partial Out}{\partial \beta} = \frac{\partial Out}{\partial BN}$$

$$\sim \frac{\partial w_i}{\partial BN} = \begin{bmatrix} 0 & -0.2 & \dots \\ -1 & 0.6 & \dots \\ \vdots & \vdots & \vdots \end{bmatrix}$$

$$\frac{\partial Out}{\partial \gamma} = \frac{\partial Out}{\partial BN}\hat{x}$$

$$\sim \frac{\partial w_i}{\partial BN} \cdot \hat{x} = \frac{\partial w_i}{\partial BN} \cdot BN(ReLU(Input)) = \begin{bmatrix} -0.8 & 0.2 & \dots \\ -1 & -0.1 & \dots \\ \vdots & \vdots & \vdots \end{bmatrix} \cdot \frac{\partial w_i}{\partial BN}$$

$$= \begin{bmatrix} -0.1 & 0.7 & \dots \\ -1.3 & 0.8 & \dots \\ \vdots & \vdots & \vdots \end{bmatrix}$$

$$BN = \gamma\hat{x} + \beta$$

$$\frac{\partial Out}{\partial \beta} = \frac{\partial Out}{\partial BN}\frac{\partial BN}{\partial \beta} = \frac{\partial Out}{\partial BN}$$

$$\frac{\partial Out}{\partial \gamma} = \frac{\partial Out}{\partial BN}\frac{\partial BN}{\partial \gamma} = \frac{\partial Out}{\partial BN}\hat{x}$$

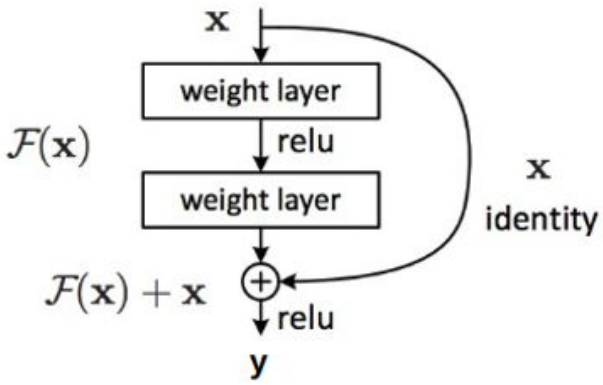
Backpropagation of Residual Networks

- When Gradient of Residual is Dominating
  - Gradient of Error has no effect on overall gradient

$$\frac{\partial E}{\partial y} \ll F^*(x)$$
$$\frac{\partial E}{\partial y} \sim 0$$

- When Gradient of Error is Dominating
  - Gradient of Residual has no effect on overall gradient

$$\frac{\partial E}{\partial y} \gg F^*(x)$$
$$\frac{\partial E}{\partial y} \cdot F^*(x) \sim 0$$



$$y = x + F(x)$$

$$\frac{\delta E}{\delta x} = \frac{\delta E}{\delta y} * \frac{\delta y}{\delta x} = \frac{\delta E}{\delta y} * (1 + F'(x))$$
$$= \frac{\delta E}{\delta y} + \frac{\delta E}{\delta y} * F'(x)$$

Residual

References

- Ioffe, S., & Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*.
- Yang, G., Pennington, J., Rao, V., Sohl-Dickstein, J., & Schoenholz, S. S. (2019). A mean field theory of batch normalization. *arXiv preprint arXiv:1902.08129*.
- Kratzert, F. (n.d.). Understanding the backward pass through Batch Normalization Layer. Retrieved from Flair of Machine Learning