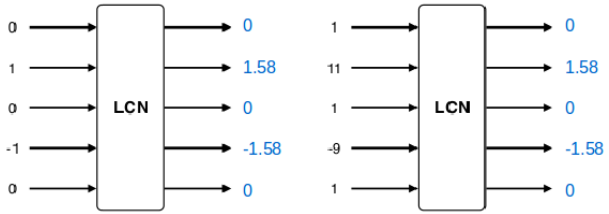
Neural Networks and Deep Learning: Modern Training & Regularization

Nicolas Thome

Conservatoire National des Arts et Métiers (Cnam)
Département Informatique

Local Response/Contrast Normalization

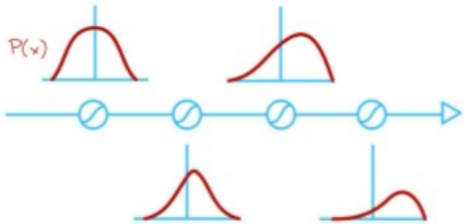
- **Normalize value wrt spatial neighbors $\mathcal{N}(i, j)$**



- Local equalizing effective
- Helps learning more invariant representations
  $\Rightarrow$ regularization, better generalization

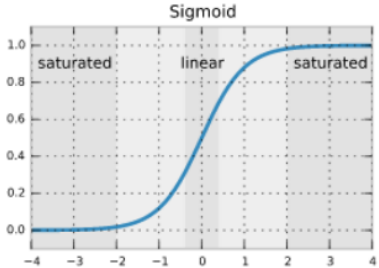
Batch Normalization (BN) [Ioffe and Szegedy, 2015]

- **Recap init: fixed input distribution known to help training**
- Training deep neural networks: distribution of hidden layers unknown, change over training time ⇒ **covariate shift**
- Importance of init, *e.g.* Xavier
- **Batch Normalization (BN):**
  ↓ importance of init, ↓ covariate shift



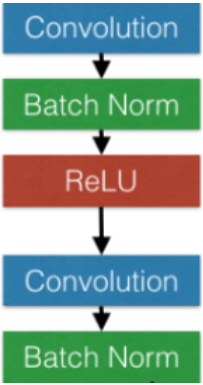
Batch Normalization (BN) [Ioffe and Szegedy, 2015]

- **Normalize input feature distribution** $\sim \mathcal{N}(0, 1)$
  - Normalization across each mini-batch:
  - $\mu_B = \frac{1}{N} \sum_{i=1}^{N} x_i$
  - $\sigma_B^2 = \frac{1}{N} \sum_{i=1}^{N} (x_i - \mu_B)^2$
  - $\hat{x_i} = \frac{x_i - \mu_B}{\sigma_B + \epsilon}$ - ϵ for numerical stability
- **Is input feature distribution** $\sim \mathcal{N}(0, 1)$ **good idea?**
  - Activation may not ever "saturate", *e.g.* sigmoid or tanh
  - Keeping in linear regime: depth useless, $\sim$ global linear model



Batch Normalization (BN) [Ioffe and Szegedy, 2015]

- Scale and shift: $y_i = \gamma \hat{x}_i + \beta$, (γ, β) trained
- Apply after FC / conv and before non-linearity



- Batch Normalization differentiable

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

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

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

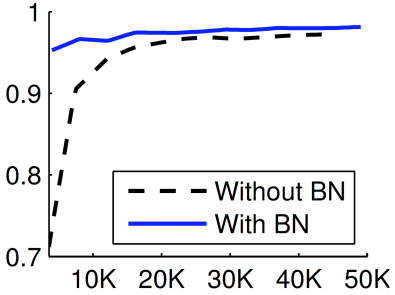
$$\frac{\partial \ell}{\partial x_i} = \frac{\partial \ell}{\partial \hat{x}_i} \cdot \frac{1}{\sqrt{\sigma_{\mathcal{B}}^2 + \epsilon}} + \frac{\partial \ell}{\partial \sigma_{\mathcal{B}}^2} \cdot \frac{2(x_i - \mu_{\mathcal{B}})}{m} + \frac{\partial \ell}{\partial \mu_{\mathcal{B}}} \cdot \frac{1}{m}$$

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

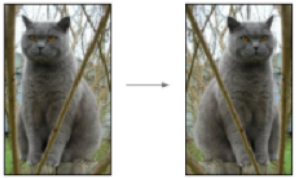
$$\frac{\partial \ell}{\partial \beta} = \sum_{i=1}^{m} \frac{\partial \ell}{\partial y_i}$$

Batch Normalization (BN) [Ioffe and Szegedy, 2015]

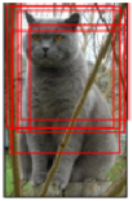
- **Applying BN at test time?**
  ⇒ Use train set statistics
- **BN Strengths**
  - **Faster training convergence ⚔ covariate shift**
  - **Regularization & generalization**
    ⇒ better performances



Training: Data-Augmentation



Flip horizontally



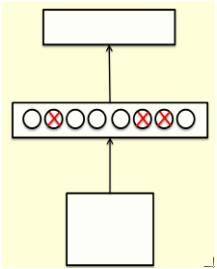
Random crops/scales

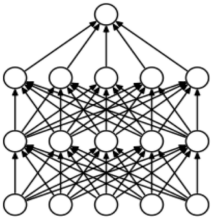
Color jittering

- Jittering, mirroring, color perturbation, rotation, stretching, shearing, lens distortions, *etc* of the original images
- Increases # training samples, adds robustness to irrelevant variations
- **Done in train AND in test**

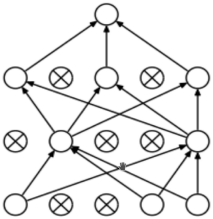
Training: Droupout [Hinton et al., 2012]

- Randomly omit each hidden unit with probability p, *e.g.* $p = 0.5$
- **Regularization technique**, limits over-fitting (better generalization)
  - Prevent co-adaptation, *i.e.* feature only helpful when other specific features present
  - May be viewed as averaging over many NN
  - Slower convergence





Standard Neural Net

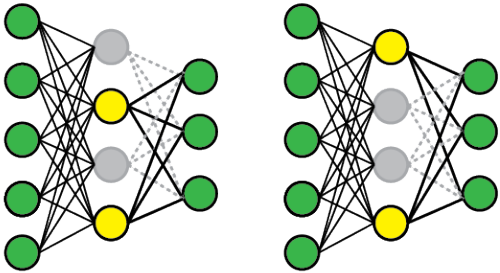


After applying dropout.

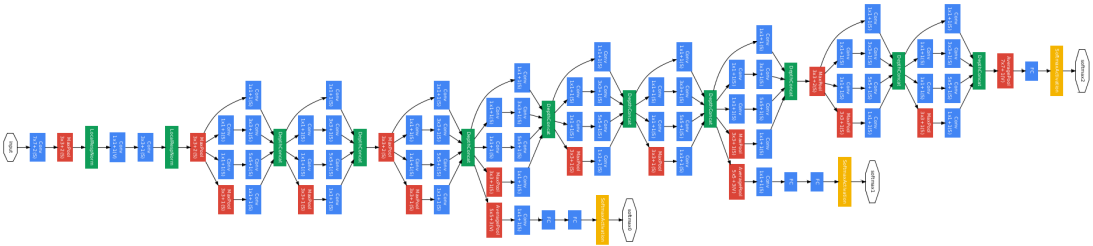
Credits: Geoffrey E. Hinton, NIPS 2012

Training: Droupout [Hinton et al., 2012]

- Training: dropout layer easily differentiable, freezing some weight updates
- What to do at test time ?
  - Sample many different architectures, average output distributions
  - Faster alternative: use all hidden units (but after /2 outgoing weights)
  - Equivalent to the geometric mean in case of single hidden layer
  - Pretty good approximation for multiple layers



Modern Training & Regularization: Conclusion



- **Dropout: important for limiting over-fitting**
  - Used in AlexNet at ImageNet'12
  - Common in current archis, especially in FC layers
- **Batch Normalization:** especially important for very deep models, *e.g.* ResNet
- **Architecture evolution since 2012?**
  ⇒ following!

References I

Hinton, G. E., Srivastava, N., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. (2012).
Improving neural networks by preventing co-adaptation of feature detectors.
CoRR, abs/1207.0580.

Ioffe, S. and Szegedy, C. (2015).
Batch normalization: Accelerating deep network training by reducing internal covariate shift.
In Bach, F. R. and Blei, D. M., editors, *ICML*, volume 37 of *JMLR Workshop and Conference Proceedings*, pages 448–456. JMLR.org.