Ch05. Drop out and Normalization

Hwanjo Yu

POSTECH

http://hwanjoyu.org

Overfitting in Deep Nets

- Large weights in deep nets suffer from overfitting
- L_1 or L_2 regularization often does not work due to **co-adaption** (neuron grouping behavior).
- Ensembles of neural networks with different model configurations may reduce overfitting, but require additional expense of training and maintaining multiple models.

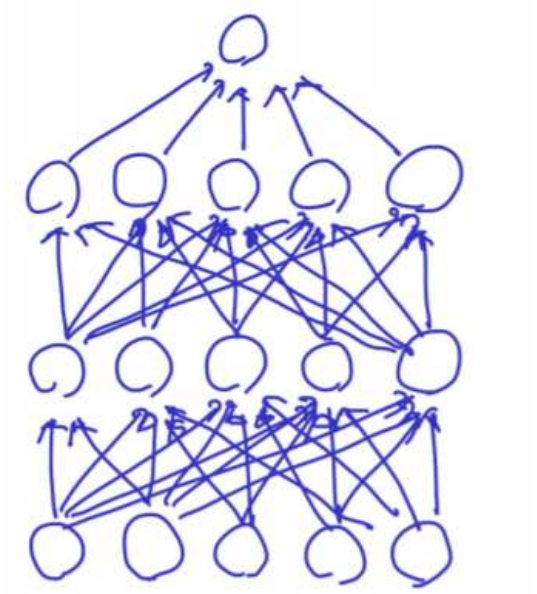
Dropout

- A new regularization method to resolve co-adaption.
- A single model is used to simulate having a large number of different network architectures.

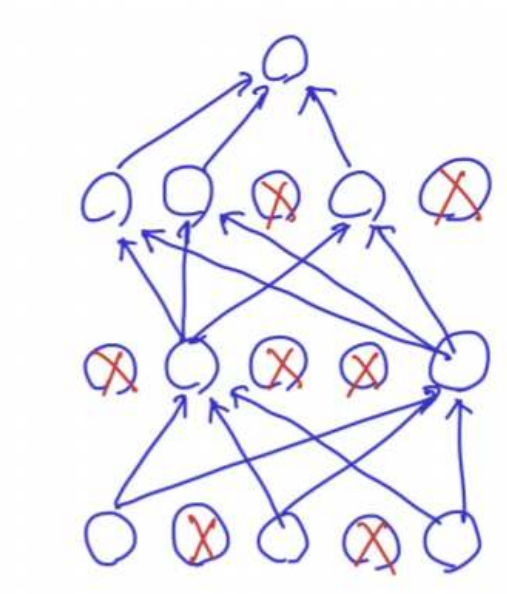
Dropout

Geoffrey Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov (2012), ”Improving neural networks by preventing co-adaptation of feature detectors,” Preprint arXiv:1207.0580.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov (2014), ”Dropout: A Simple Way to Prevent Neural Networks from Overfitting,” JMLR.



(a) NN

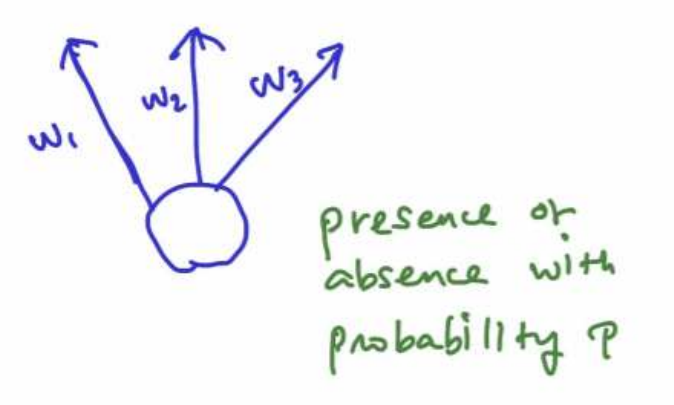


(b) Dropout applied

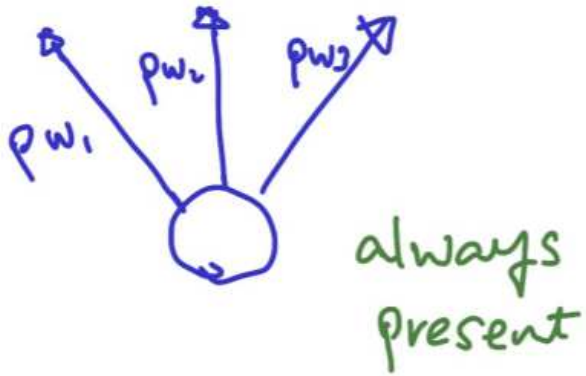
- Form a vector of independent Bernoulli random variables, $\mathbf{z}^{(l)}$, where $z^{(l)} \sim \mathrm{Bern}(p)$.
- Feedforward operations are:

$$h_i^{(l+1)} = \sigma\left(\mathbf{w}_i^{(l+1)\mathrm{T}}\left(\mathbf{h}^{(l)} \odot \mathbf{z}^{(l)}\right) + b_i^{(l+1)}\right).$$

Dropout at Test Time



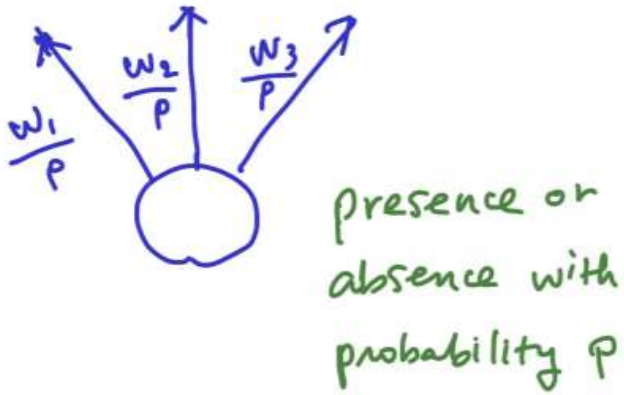
(a) At training time



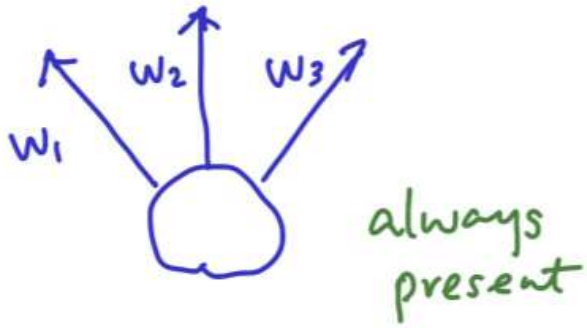
(b) At test time

- Approximates the effect of averaging the predictions of all these thinned networks by simply using a single unthinned network that has smaller weights.

Inverted Dropout



(a) At training time



(b) At test time

- At training time, rescale the weights $\mathbf{w}_i^{(l)} \leftarrow \mathbf{w}_i^{(l)}/p$ after applying the dropout.
- Do nothing at test time, which makes inference faster.

Batch Normalization

S. Ioffe and C. Szegedy (2015), ”Batch normalization: Accelerating deep network training by reducing internal covariate shift,” ICML.

- Given a set of samples, $\{x_1, \ldots, x_N\}$, compute the sample mean and variance

$$\mu = \frac{1}{N}\sum_{n=1}^{N} x_n\,, \qquad \sigma^2 = \frac{1}{N}\sum_{n=1}^{N}(x_n - \mu)^2\,.$$

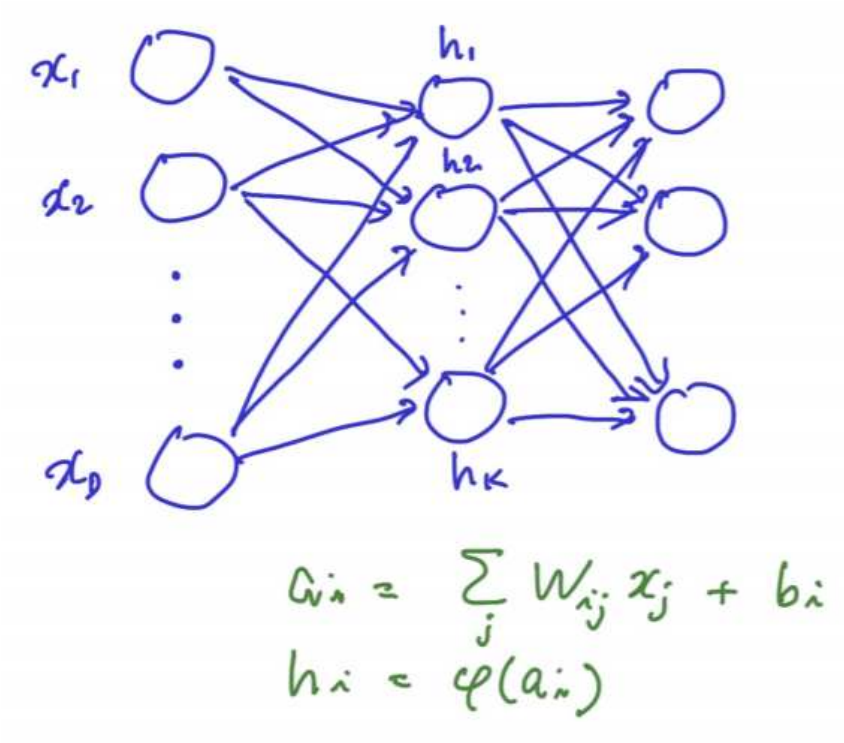
- Then, normalize the data via the following transformation:

$$z_n = \frac{x_n - \mu}{\sigma}$$

- such that $\{z_n\}$ obeys zero mean and unit variance.

Bath Normalization: Internal Covariate Shift

- The change in network parameters during training causes the change in the distribution of network activations.
- It would help if the pre-activations at each layer were unit Gaussians.



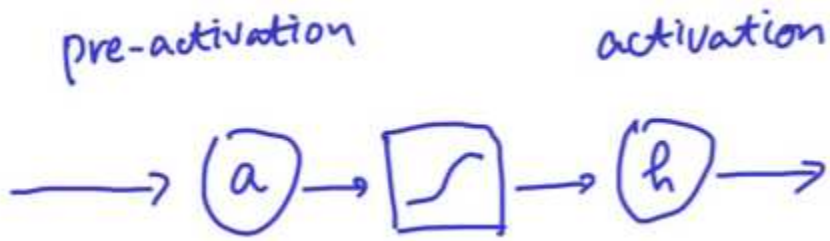
Batch Normalization: Original Network

- Pre-activation is calculated as a linear sum of incoming inputs:

$$a_i^{(l)} = \sum_j W_{i,j}^{(l)} h_j^{(l-1)} + b_i^{(l)} .$$

- Activation is a non-linear transform of the pre-activation:

$$h_i^{(l)} = \varphi\left(a_i^{(l)}\right).$$



Batch Normalization: adding BN layer

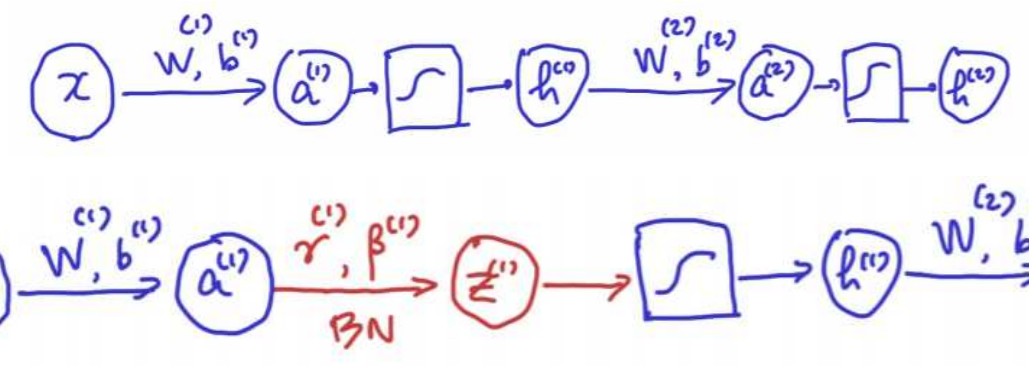
- BN is applied to individual dimension for each mini-batch of size M.
- Normalize pre-activation a_i:

$$\tilde{z}_i = \frac{a_i - \mu_i}{\sqrt{\sigma_i^2 + \epsilon}}, \quad \text{where} \quad \mu_i = \frac{1}{M}\sum_{m=1}^{M} a_{i,m}\,, \;\; \sigma_i^2 = \frac{1}{M}\sum_{m=1}^{M}\left(a_{i,m} - \mu_i\right)^2 .$$

- Scale and shift:

$$z_i = \gamma_i \tilde{z}_i + \beta_i,$$

- where (γ_i, β_i) are **parameters** that are learned via back-prop.



- Since the mean is subtracted, bias term $b_i^{(l)}$ is not necessary. In other words, simply use $a_i^{(l)} = \sum_j W_{i,j}^{(l)} h_j^{(l-1)}$

Batch Normalization in Inference Phase

- Suppose that $\mu_{i,\mathcal{B}}$ and $\sigma^2_{i,\mathcal{B}}$ are mean and variance computed using a mini-batch $\mathcal{B}$ of size M.
- In inference time, we compute exponentially weighted moving average of them, $\mathbb{E}_{\mathcal{B}}[\mu_{i,\mathcal{B}}]$ and $\mathbb{E}_{\mathcal{B}}[\sigma^2_{i,\mathcal{B}}]$, incrementally.
- Scale and shift:

$$\tilde{z}_i = \frac{a_i - \mathbb{E}_{\mathcal{B}}[\mu_{i,\mathcal{B}}]}{\sqrt{\mathbb{E}_{\mathcal{B}}[\sigma^2_{i,\mathcal{B}}] + \epsilon}} \quad \text{and} \quad z_i = \gamma_i \tilde{z}_i + \beta_i$$

Batch Normalization: Experiments on ImageNet Classification

- Experiments setting
  - Increase the learning rate (step size): Speed up via higher learning rate
  - Remove dropout
  - Accelerate the learning rate decay: Lower the learning rate 6 times faster
  - Reduce the L2 weight regularization

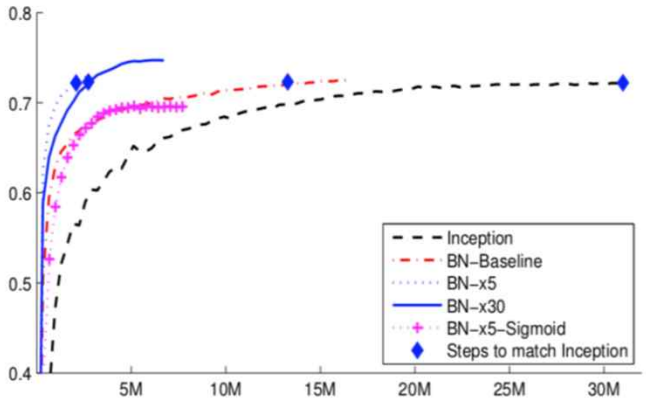


Figure 2. Single crop validation accuracy of Inception and its batch-normalized variants, vs. the number of training steps.

Model	Steps to 72.2%	Max accuracy
Inception	$31.0 \cdot 10^6$	72.2%
BN-Baseline	$13.3 \cdot 10^6$	72.7%
BN-x5	$2.1 \cdot 10^6$	73.0%
BN-x30	$2.7 \cdot 10^6$	74.8%
BN-x5-Sigmoid		69.8%

Figure 3. For Inception and the batch-normalized variants, the number of training steps required to reach the maximum accuracy of Inception (72.2%), and the maximum accuracy achieved by the network.

[S. Ioffe and C. Szegedy (2015), "Batch normalization: Accelerating deep network training by reducing internal covariate shift," ICML]

Layer Normalization

J. L. Ba, J. R. Kiros, and G. E. Hinton (2016), "Layer normalization," arXiv:1607.06450.

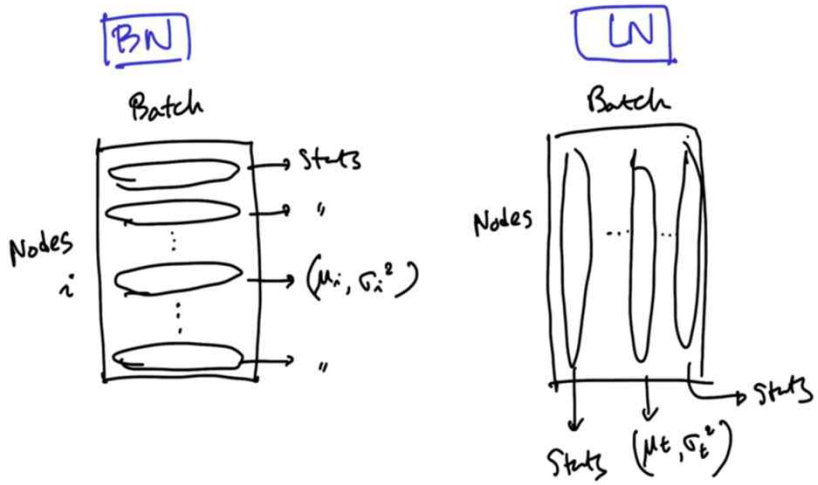
- What's wrong with batch normalization?
  - BN requires running averages of summed input statistics.
  - In feed-forward networks with fixed depth, it is straightforward to store these statistics separately for each hidden layer.
  - **Difficult to apply to recurrent connections:** Applying batch normalization to RNNs appears to require different statistics for different time-steps. (because the summed inputs to the recurrent neurons in a RNN often vary with the length of the sequence)
  - **Lower limit on the batch size:** Batch normalization cannot be applied to online learning tasks where the batch size is small.
- Compute the layer normalization statistics over all the hidden units in the same layer:

$$\mu_t^{(l)} = \frac{1}{H}\sum_{i=1}^{H} a_{i,t}^{(l)}, \qquad \sigma_t^{(l)} = \sqrt{\frac{1}{H}\sum_{i=1}^{H}\left(a_{i,t}^{(l)} - \mu_t^{(l)}\right)^2}.$$

- LN in a RNN is performed:

$$\mathbf{a}_t = \mathbf{W}_{hh}\mathbf{h}_{t-1} + \mathbf{W}_{hx}\mathbf{x}_t,$$
$$\mathbf{h}_t = \varphi\left(\frac{\mathbf{g}}{\sigma_t\mathbf{1}} \odot (\mathbf{a}_t - \mu_t\mathbf{1}) + \mathbf{b}\right).$$

Layer Normalization



Experiments on MS COCO Datasets: