Deep Learning for Visual Recognition Part 2

Injung Kim
Handong Global University

Advances of CNN

- Improved structures
  - Max-out network[Goodfellow13]
  - Network-in-network[Lin13]
  - Spatial pyramid pooling CNN [He14]
  - Very deep CNN [Simonyan15]
  - GoogLeNet[Szegedy14], Inception v2,v3,v4, Inception-ResNet
  - Residual learning networks [He15]
  - Dense convolutional networks [Huang16]
  - Dual-Path Net[Chen17]
  - SENet[Hu17], Non-local nets[Wang18], BAM[Park18], CBAM[Woo18]

- Improved learning algorithms
  - Batch normalization [Ioffe15], layer/weight/group normalization
  - Xavier init[Xavier10], He init[He15], LL-init [Balduzzi17]

- Detection & classification
  - R-CNN[Girshik14], Fast R-CNN[Girshik14], Faster R-CNN[Ren15]
  - Mask R-CNN[He17]
  - Visual attention models [Mnih14, Ba15, Sermanet15]
  - YOLO[Redmon16], SSD[Liu16]
  - FCN[Long16], DeepLab1/2/3/3+[Chen15-18], PSP-Net[Zhao17], GCN[Peng17]

Advances of CNN

- Visualization and understanding
  - Visualization using deconvolution layers [Zeiler13]
  - Class saliency maps [Simonyan13]
  - Inverting CNN [Mahendran14]

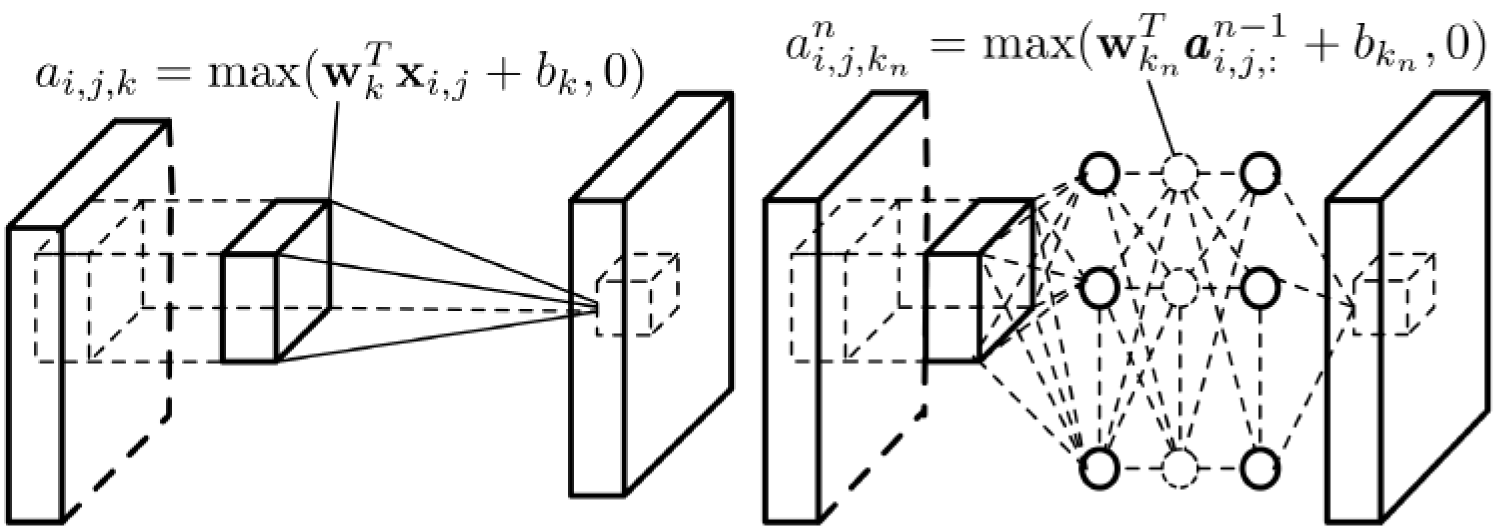
- Building lightweight networks
  - Network compression [Bucilu06]
  - Knowledge distillation [Hinton14]
  - FitNet [Romero14]
  - SqueezNet [Iandola16]
  - ShuffleNet [Zhang17]
  - MobileNet, MobileNet.v2, MobileNet.v3

Agenda

- Quick Review
- <u>Advanced CNN Models</u>
  - Maxout net
  - Network in networks
  - Spatial pyramid pooling
  - VGG net
  - Inception
  - ResNet
- Nonlinearity Functions

Network in Networks

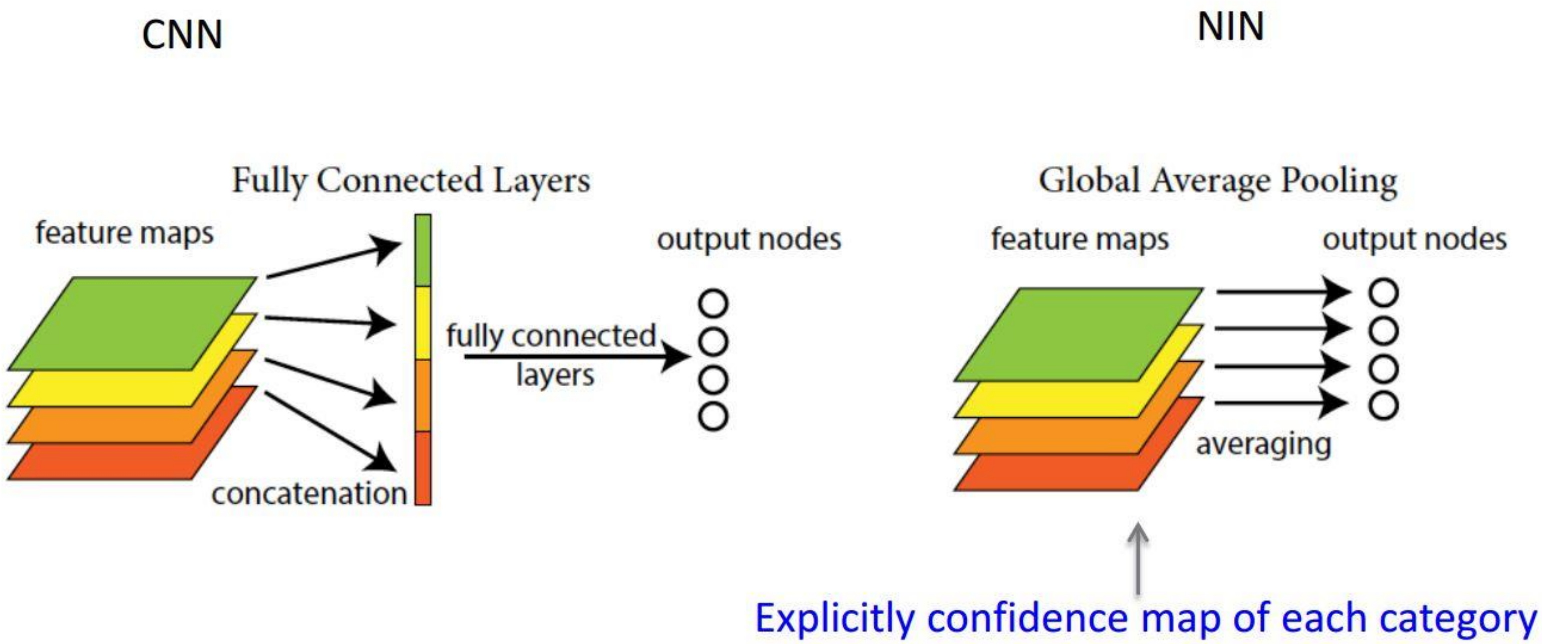
- Lin, et al. "Network In Network", 2014
- MLPConv layer to learn non-linear filters
  - Equivalent to (conv + CCCP)
    - CCCP: a series of 1x1 conv layers
    - 1x1 conv layers are also used to reduce # of feature maps



(a) Linear convolution layer (b) Mlpconv layer

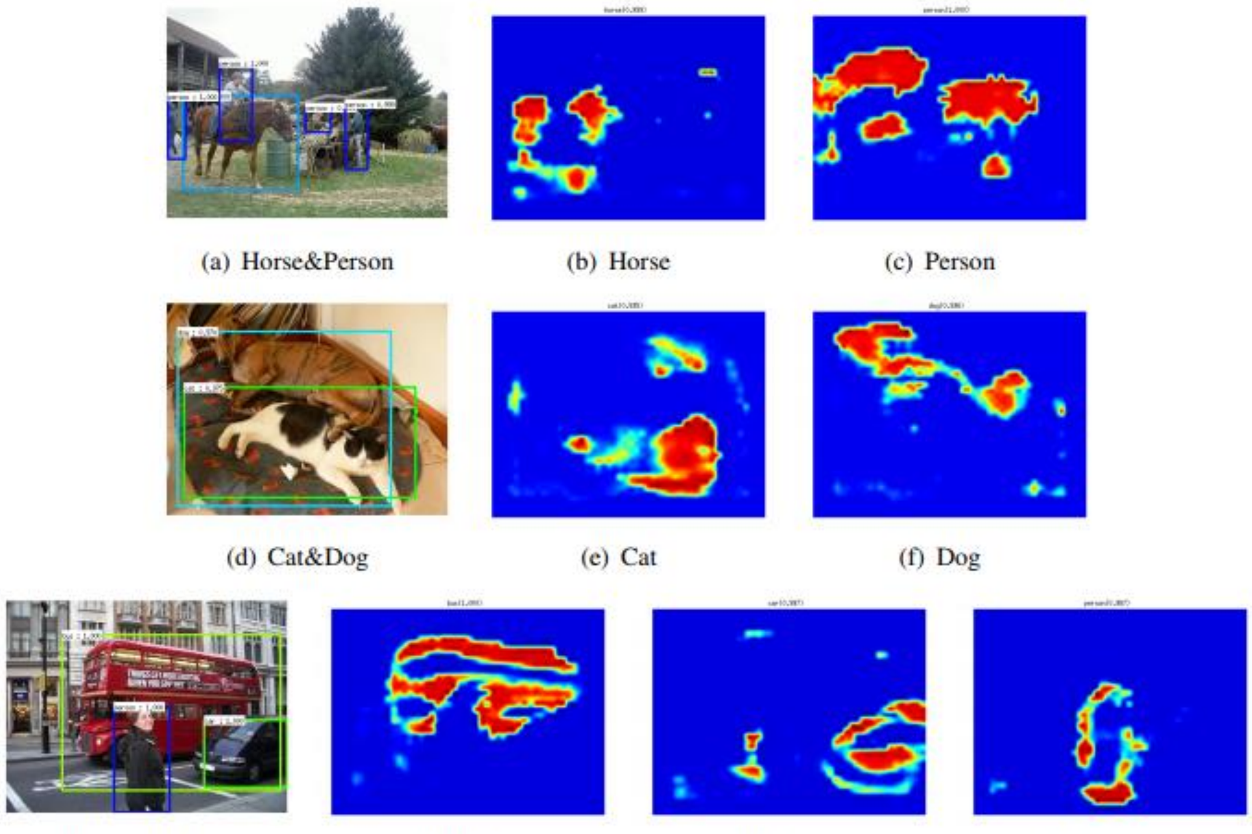
Network In Networks

- Lin, et al. “Network In Network”, 2014
- Global average pooling (GAP) layer Less suffers from overfitting than fully-connected layer
  - Usually, preceded by CCCP
  - Less sensitive to position variation of salient feature



GAP for Object Localization

- Cook, “GAP for Object Localization,” Apr. 2017.
- Qui, “Global Weighted Average Pooling Bridges Pixel-level Localization and Image-level Classification,” Sep. 2018.



(a) Horse&Person (b) Horse (c) Person

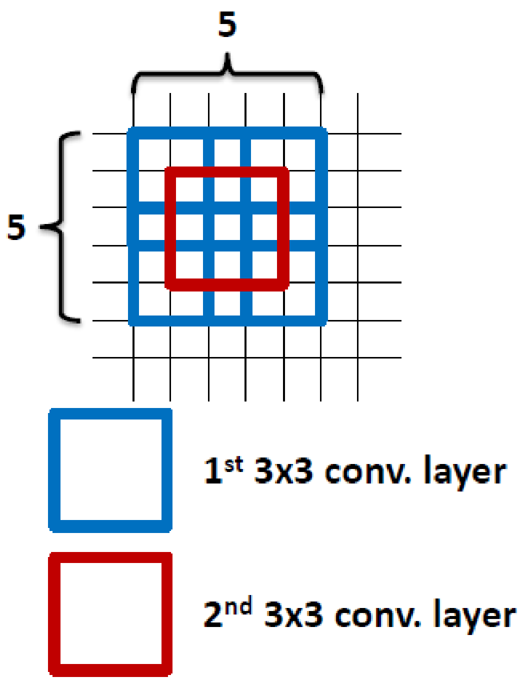
(d) Cat&Dog (e) Cat (f) Dog

(g) Bus&Car&Person (h) Bus (i) Car (j) Person

VGG Net

- Simonyan and Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognition" 2015
  - Stack convolution layers have large receptive field
    - Two 3x3 layers – 5x5 receptive field
    - Three 3x3 layers – 7x7 receptive field
  - More nonlinearity
  - Less parameters

➔ Multiple 3x3 conv layers are better than single conv layer with a large filter



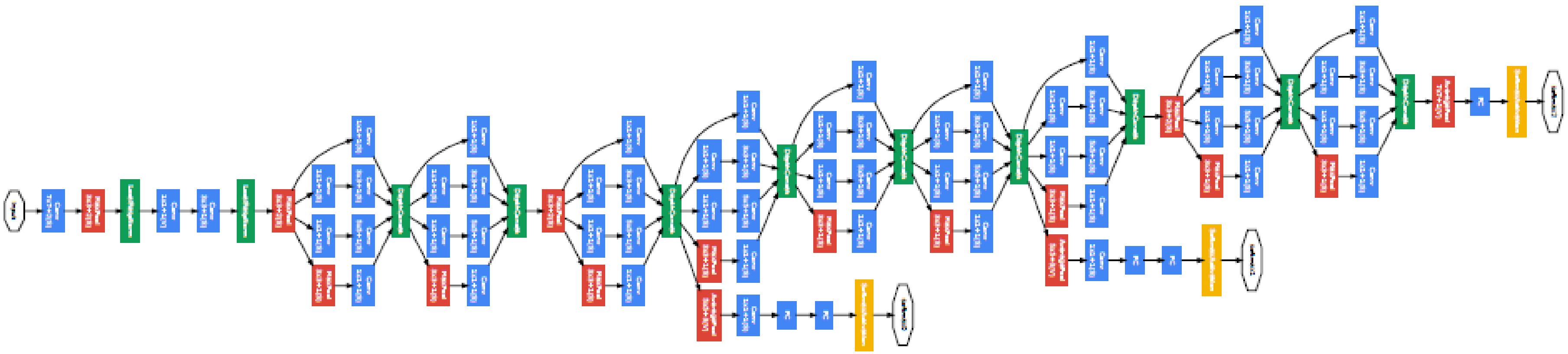
VGG Net

- Simonyan and Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognition" 2015

Method	top-1 val. error (%)	top-5 val. error (%)	top-5 test error (%)
VGG (2 nets, multi-crop & dense eval.)	**23.7**	**6.8**	**6.8**
VGG (1 net, multi-crop & dense eval.)	24.4	7.1	7.0
VGG (ILSVRC submission, 7 nets, dense eval.)	24.7	7.5	7.3
GoogLeNet (Szegedy et al., 2014) (1 net)	-	7.9	
GoogLeNet (Szegedy et al., 2014) (7 nets)	-	**6.7**	

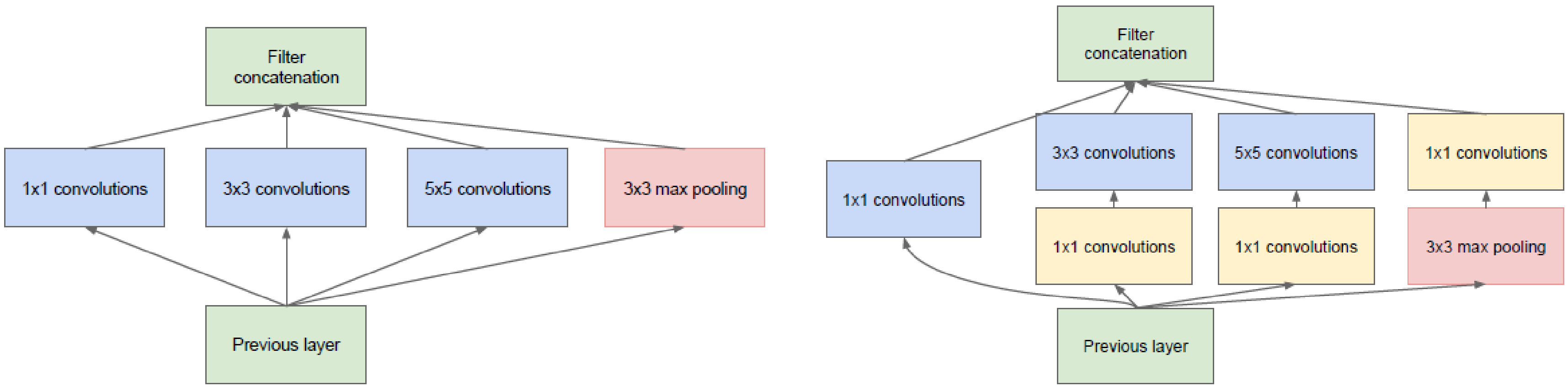
Inception (GoogLeNet)

- Szegedy, et al. “Going deeper with convolutions”, 2015
  - Very deep network with 22 layers
  - Inception module



Inception (GoogLeNet)

- Inception module
  - Multi-scale convolution
  - Dimensionality reduction by 1x1 convolution



(a) Inception module, naïve version

(b) Inception module with dimension reductions

Figure 2: Inception module

ResNet

- He, et.al, “Deep Residual Learning for Image Recognition”, Dec. 2015
  - Target function $H(x) = F(x) + x$
  - Residual Function $F(x) = H(x) - x$
  - Deep residual network contains 152 layers

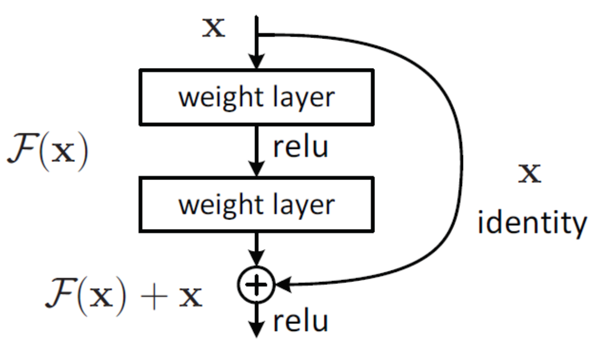
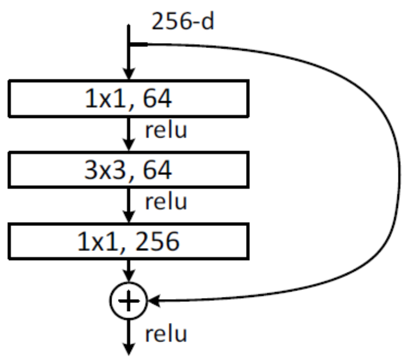


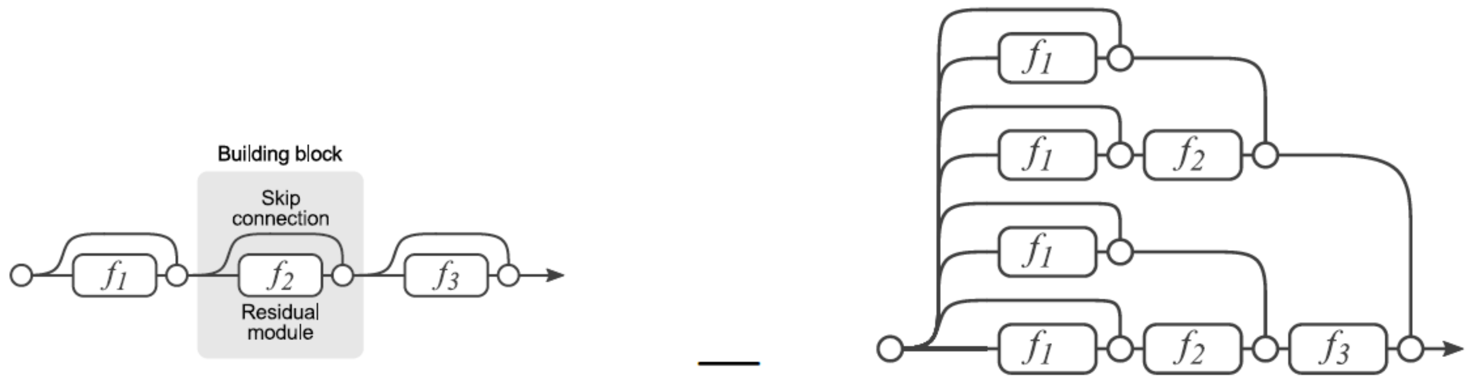
Figure 2. Residual learning: a building block.



Bottleneck building block

ResNet

- Veit, et.al., “Residual Networks Behave Like Ensembles of Relatively Shallow Networks”, 2016
  - During training, gradients are mainly from relatively shallow paths



(a) Conventional 3-block residual network

(b) Unraveled view of (a)

Figure 1: Residual Networks are conventionally shown as (a), which is a natural representation of Equation (1). When we expand this formulation to Equation (6), we obtain an *unraveled view* of a 3-block residual network (b). Circular nodes represent additions. From this view, it is apparent that residual networks have $O(2^n)$ implicit paths connecting input and output and that adding a block doubles the number of paths.

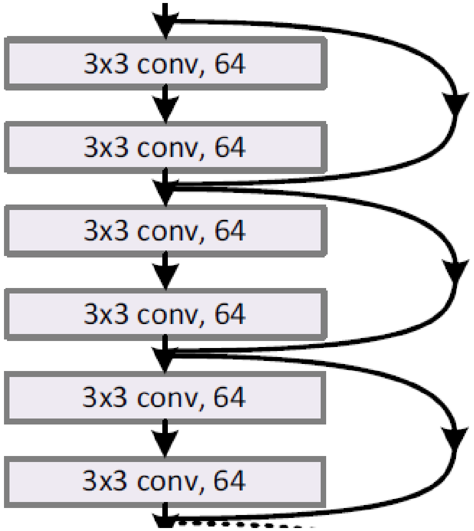
ResNet.v2

- He, et al., "Identity Mappings in Deep Residual Networks," Jul. 2016.
  - Analysis of ResNet

$$x_{l+1} = x_l + \mathcal{F}(x_l, \mathcal{W}_l)$$

$$x_L = x_l + \sum_{i=l}^{L-1} \mathcal{F}(x_i, \mathcal{W}_i)$$

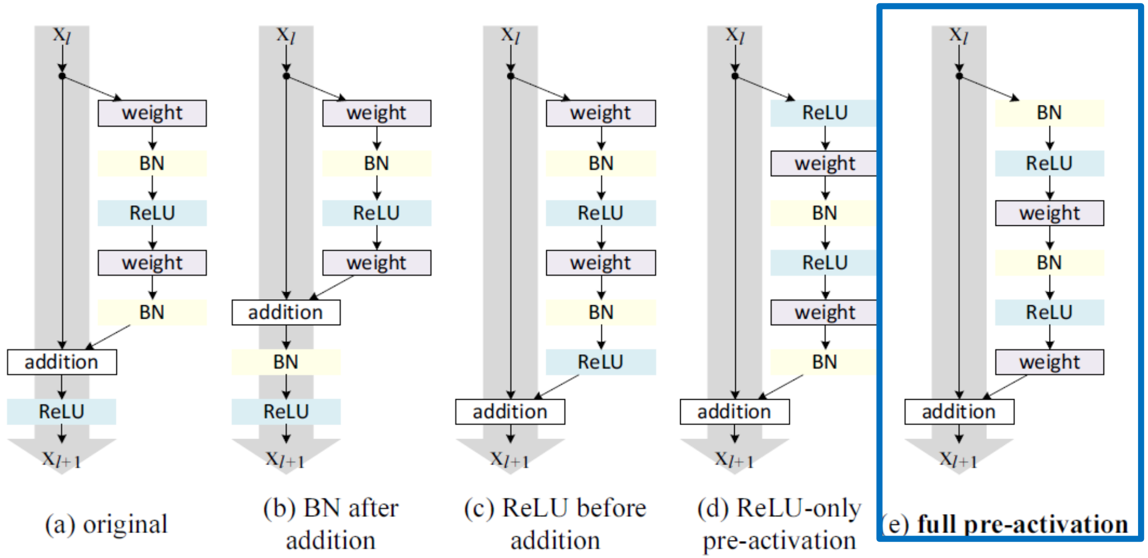
$$\frac{\partial \mathcal{E}}{\partial x_l} = \frac{\partial \mathcal{E}}{\partial x_L}\frac{\partial x_L}{\partial x_l} = \frac{\partial \mathcal{E}}{\partial x_L}\left(1 + \frac{\partial}{\partial x_l}\sum_{i=l}^{L-1} \mathcal{F}(x_i, \mathcal{W}_i)\right)$$



- Gradient of top layer ($\frac{\partial \mathcal{E}}{\partial x_L}$) directly propagates to lower layers

ResNet.v2

- He, et al., “Identity Mappings in Deep Residual Networks,” Jul. 2016.
  - Typical blocks: Weight–BN–ReLU – Weight–BN–ReLU – …
  - Pre-activation is better than post-activation
    - BN+ReLU+Weight > Weight+BN+ReLU
  - Identity mapping is the best among skip connection



Agenda

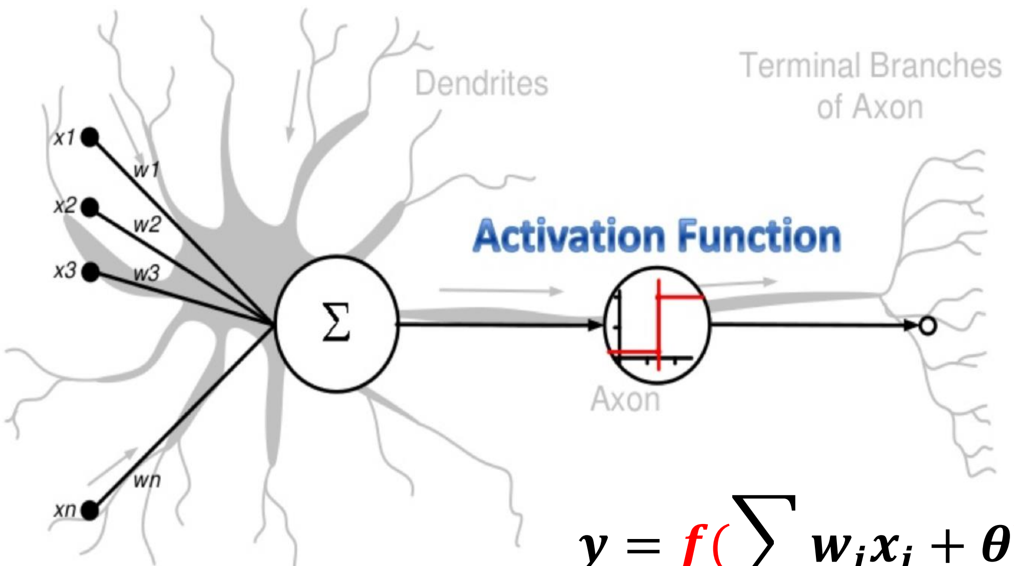
- Quick Review

- Advanced CNN Models
  - Maxout net
  - Network in networks
  - Spatial pyramid pooling
  - VGG net
  - Inception
  - ResNet

- <u>Nonlinearity Functions</u>

Nonlinearity Functions

- Activation function
  - Non-linearity
  - Restrict outputs in a specific range
  - Measurement → probability or decision



$$y = f\left(\sum_i w_i x_i + \theta\right)$$

Output Units

- Activation functions for output units
  - Identity function: unbounded value (for regression)
  - Sigmoid: Bernoulli distribution, values in range (0,1)
  - Tanh: Sigmoid scaled to range (−1,1)
  - Softmax: probabilities of categories (for classification)

Linear Output Units

- No activation function

$$\hat{y} = \boldsymbol{W}^\top \boldsymbol{h} + \boldsymbol{b}$$

  - Regression
  - The mean of conditional Gaussian distribution

$$p(\boldsymbol{y} \mid \boldsymbol{x}) = \mathcal{N}(\boldsymbol{y}; \hat{\boldsymbol{y}}, \boldsymbol{I})$$

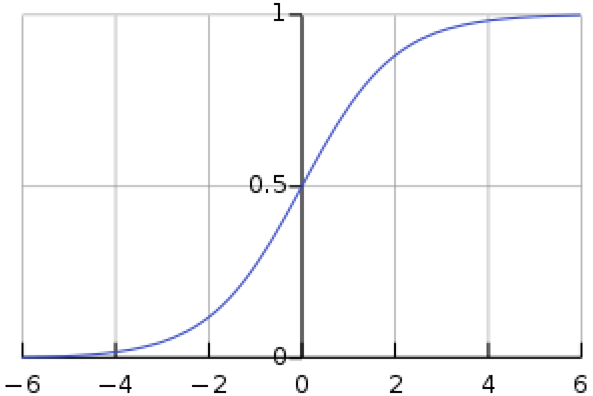
Ex) encoder of VAE outputs $\mu_{z|x}, \log \sigma^2_{z|x}$

Sigmoid Output Units

- Often used to model Bernoulli distribution

$$\hat{y} = \sigma\left(\boldsymbol{w}^\top \boldsymbol{h} + b\right)$$

$$\sigma(x) = \frac{1}{1 + \exp(-x)}$$



  - Neural net predicts $p(y = 1|x)$
    - Predicting a binary variable y
    - Classification with two classes

    Learns $z = \log \tilde{P}(y = 1 \mid \boldsymbol{x})$ by $\boldsymbol{z} = \boldsymbol{W}^\top \boldsymbol{h} + \boldsymbol{b}$

$$\log \tilde{P}(y) = yz$$

$$\tilde{P}(y) = \exp(yz)$$

$$p(y = 1) = \frac{\exp(z)}{\sum_{y \in \{0,1\}} \exp(yz)}$$

$$= \frac{\exp(z)}{\exp(z) + \exp(0)} = \frac{1}{1 + \exp(-z)}$$

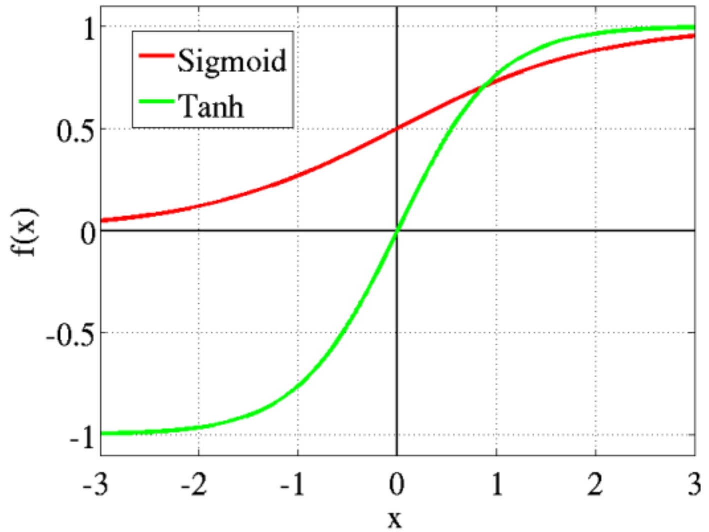
Hyperbolic Tangent Unit

- Tanh: scaled Sigmoid

$$Sigmoid(x) = \frac{1}{1+e^{-x}}$$

$$\tanh(x) = \frac{e^{x}-e^{-x}}{e^{x}+e^{-x}} = \frac{1-e^{-2x}}{1+e^{-2x}} = \frac{2-(1+e^{-2x})}{1+e^{-2x}}$$

$$= \frac{2}{1+e^{-2x}} - 1 = \mathbf{2 \cdot Sigmoid(2x) - 1}$$



Derivatives of Sigmoid/Tanh

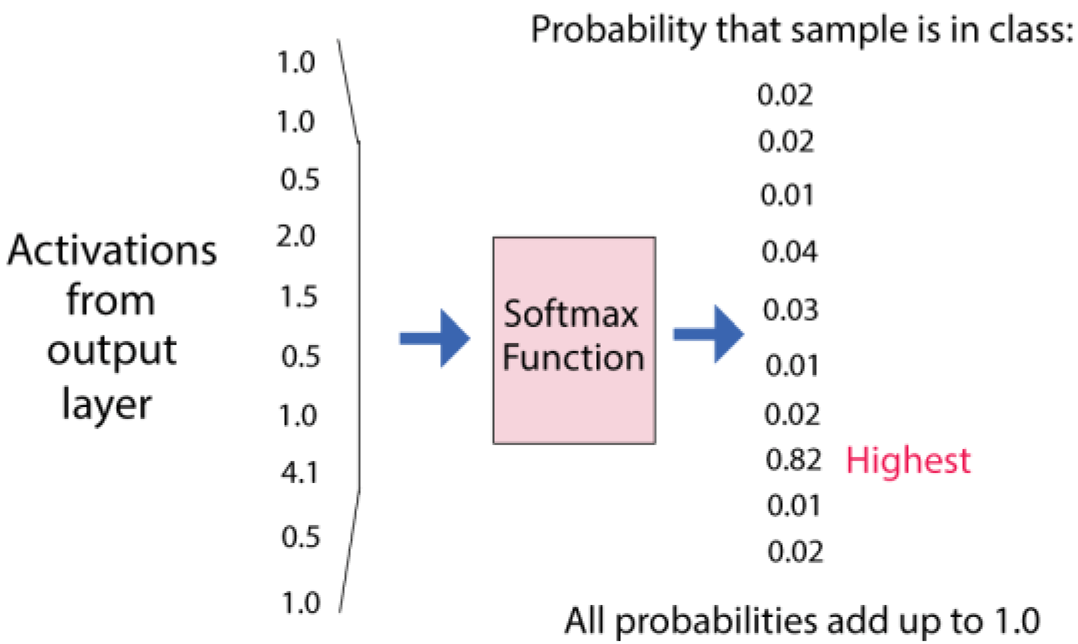
- Derivative of softmax
  - $y = softmax(z)$
  - $\frac{\partial y}{\partial z} = y(1 - y)$

- Derivative of tanh
  - $y = tanh(z)$
  - $\frac{\partial y}{\partial z} = 1 - y^2$

Softmax Output Units

- Probability distribution over n different classes

$$\mathrm{softmax}(\boldsymbol{z})_i = \frac{\exp(z_i)}{\sum_j \exp(z_j)}$$



Softmax Output Units

- Probability distribution over n different classes
  - Approximate $z_i = \log \tilde{P}(y = i \mid \boldsymbol{x})$
    by $\boldsymbol{z} = \boldsymbol{W}^\top \boldsymbol{h} + \boldsymbol{b}$
  - Take exponent and normalize

$$\text{softmax}(\boldsymbol{z})_i = \frac{\exp(z_i)}{\sum_j \exp(z_j)}$$

$$\log \text{softmax}(\boldsymbol{z})_i = z_i - \log \sum_j \exp(z_j)$$

Derivative of Softmax

- Softmax

$$\hat{y}_j = \frac{e^{o_j}}{\sum_k e^{o_k}}$$

- Derivative of fraction function

$$\left(\frac{f(x)}{g(x)}\right)' = \frac{f'(x)\,g(x) - f(x)g'(x)}{g(x)^2}$$

- Derivative of Softmax

$$\boldsymbol{\frac{\partial \hat{y}_j}{\partial o_i} = \hat{y}_j\left(1_{i,j} - \hat{y}_i\right)}$$

  - If $i = j$, $\frac{\partial \hat{y}_j}{\partial o_i} = \frac{e^{o_j}(\sum_k e^{o_k}) - e^{o_j}e^{o_i}}{(\sum_k e^{o_k})^2} = \frac{e^{o_j}(\sum_k e^{o_k})}{(\sum_k e^{o_k})^2} - \frac{e^{o_j}e^{o_i}}{\left(\sum_k e^{o_k}\right)^2}$

$$= \frac{e^{o_j}}{\sum_k e^{o_k}} - \frac{e^{o_j}}{\sum_k e^{o_k}}\frac{e^{o_i}}{\sum_k e^{o_k}} = \hat{y}_j - \hat{y}_j\hat{y}_i = \boldsymbol{\hat{y}_j(1 - \hat{y}_i)}$$

  - If $i \neq j$, $\frac{\partial \hat{y}_j}{\partial o_i} = \frac{-e^{o_j}e^{o_i}}{(\sum_k e^{o_k})^2} = -\frac{e^{o_j}}{\sum_k e^{o_k}}\frac{e^{o_i}}{\sum_k e^{o_k}} = \boldsymbol{-\hat{y}_j\hat{y}_i = \hat{y}_j(-\hat{y}_i)}$

Derivative of Softmax Cross Entropy

- Softmax

$$\hat{y}_i = \frac{\exp(z_i)}{\sum_i \exp(z_i)}$$

- Cross entropy

$$L = -\sum_i y_i \log(\hat{y}_i)$$

  - $y_i \in \{0,1\}$: label (only $y_{true} = 1$)

- Gradient w.r.t. logit z_t

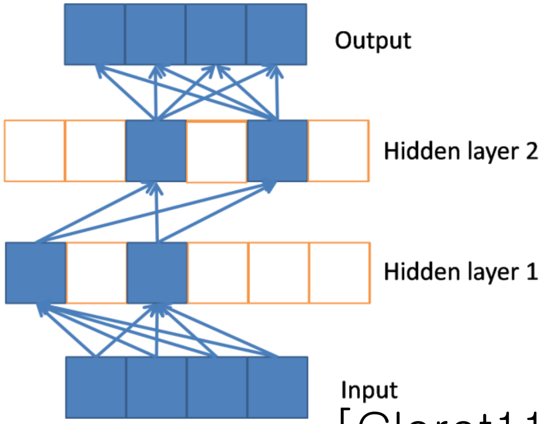
$$\frac{\partial L}{\partial z_i} = \sum_j \frac{\partial L}{\partial \hat{y}_j} \frac{\partial \hat{y}_j}{\partial z_i} = -\frac{1}{\hat{y}_{true}} \hat{y}_{true}\left(1_{i,true} - \hat{y}_i\right) = \boldsymbol{\hat{y}_i - 1_{i,true}}$$

Hidden Units

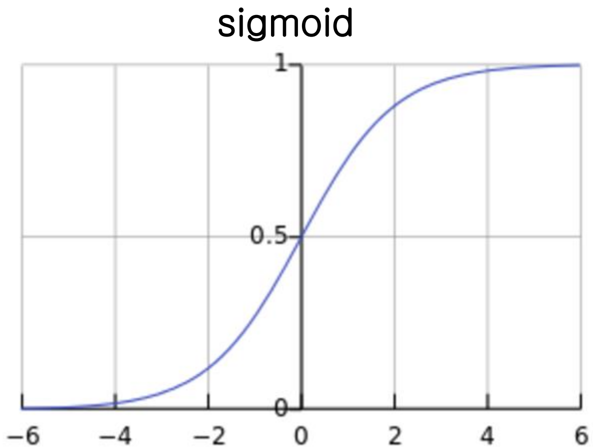
- Activation functions for hidden units
  - Traditional units
    - Sigmoid, Tanh
    - RNN, gates, regression
  - Piece-wise linear units
    - ReLU, LReLU, PReLU, RReLU, ELU
  - Gated units
    - GTU, GLU

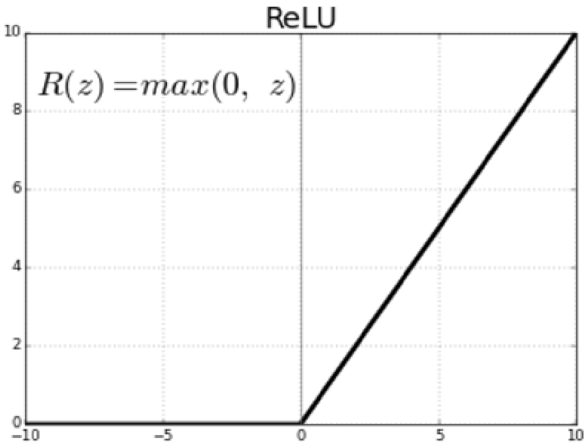
ReLU Activation Function [Hinton10]

- ReLU (Rectified Linear Unit)
  - Faster than Sigmoid or Tanh
  - No ‘saturated regime’
  - Makes network activation sparse
- Problems
  - No gradient for negative input
  - Unbounded in positive direction



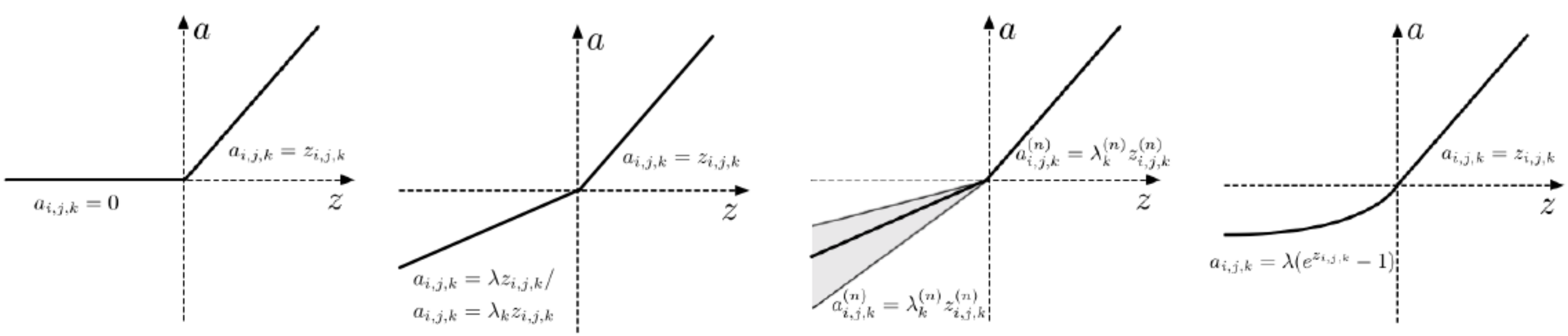
[Glorot11]





Variations of ReLU

- Leaky ReLU (LReLU) $$a_{i,j,k} = \max(z_{i,j,k}, 0) + \lambda_k \min(z_{i,j,k}, 0)$$
  - λ is fixed
- Parametric ReLU (PReLU)
  - λ is learned
- Randomized ReLU (RReLU)
  - λ is randomly sampled
- Exponential LU (ELU) $$a_{i,j,k} = \max(z_{i,j,k}, 0) + \min(\lambda(e^{z_{i,j,k}} - 1), 0)$$



(a) ReLU (b) LReLU/PReLU (c) RReLU (d) ELU

Q&A

Thank you
for your attention!