# Neural Networks II

Il-Chul Moon
Dept. of Industrial and Systems Engineering
KAIST
icmoon@kaist.ac.kr

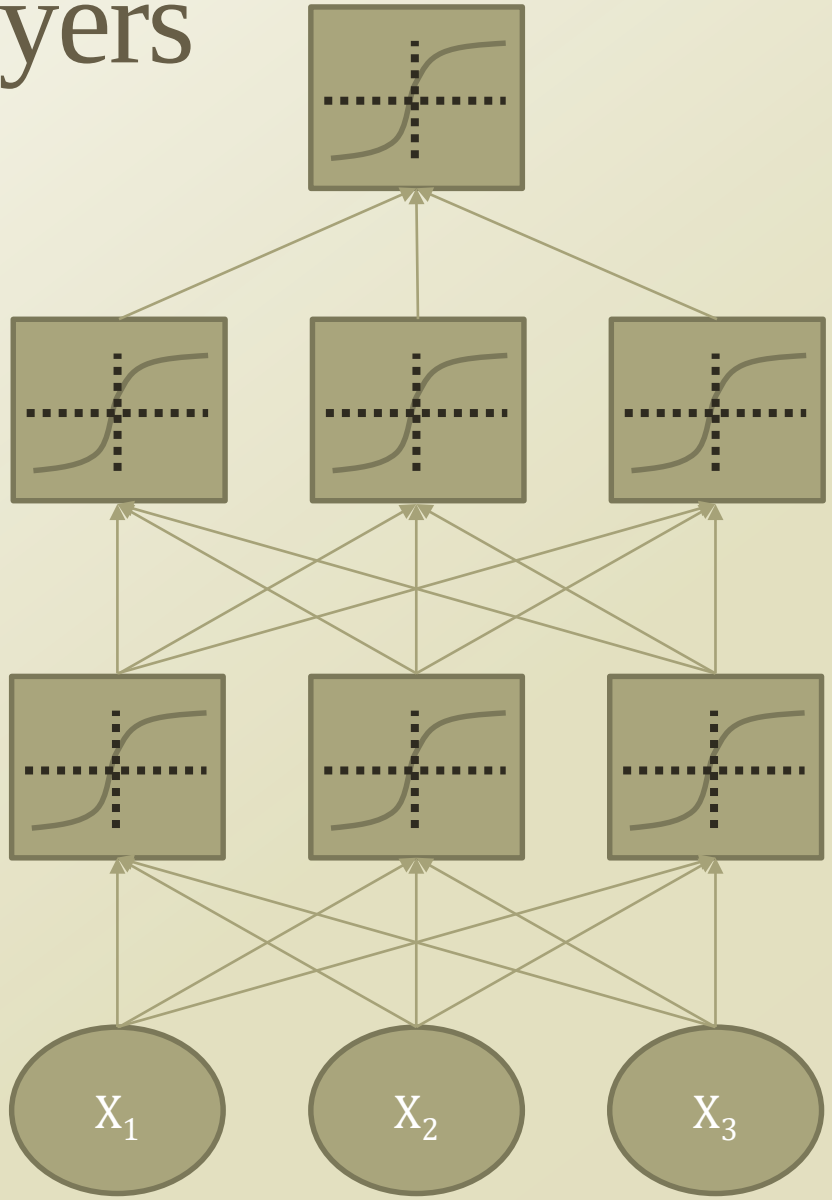
Copyright © 2012 by Il-Chul Moon, Dept. of Industrial and Systems Engineering, KAIST

# STRUCTURES OF NEURAL NETWORKS

Copyright © 2012 by Il-Chul Moon, Dept. of Industrial and Systems Engineering, KAIST

# *Detour:* Multiple Layers of Neural Network

- Considering ***L*** hidden layers
  - Neuron input activation function for the ***k-th*** layer
    - $a^k(x) = b^k + W^k h^{(k-1)}(x)$
  - Neuron output activation function for the ***k-th*** layer
    - $h^k(x) = g\left(a^k(x)\right)$
  - Output layer activation
    - $h^{L+1}(x) = o\left(a^{L+1}(x)\right) = f(x)$
- Each function is available by choices
  - Sigmoid functions
  - Softmax functions




Copyright © 2012 by Il-Chul Moon, Dept. of Industrial and Systems Engineering, KAIST

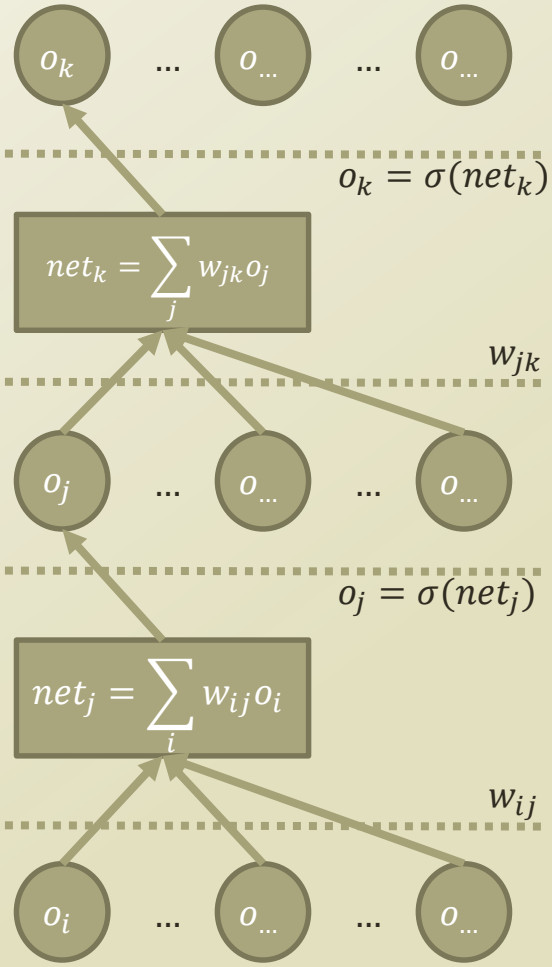
# *Detour:* Backpropagation

- Backpropagation
  - Do
    - For training examples, x

      // forward pass of the neural net.
      - $o_k = f(x; w)$
      - $E = \frac{1}{2}(\sum_k (t_k - o_k)^2)$

      // backward pass of the neural net.
      - Calculate $\delta_k = (t_k - o_k) o_k (1 - o_k)$
      - For the backward-pass from the top to the bottom
        - Calculate $\delta_j = o_j(1 - o_j)\sum_k \delta_k w_{jk}$

      // weight update
      - Update $w_{jk}$ with $w_{jk}^{t+1} \leftarrow w_{jk}^t + \eta \delta_k o_j$
      - For the backward-pass from the top to the bottom
        - Update $w_{ij}^{t+1} \leftarrow w_{ij}^t + \eta o_i \delta_j$
  - Until converges




Copyright © 2012 by Il-Chul Moon, Dept. of Industrial and Systems Engineering, KAIST

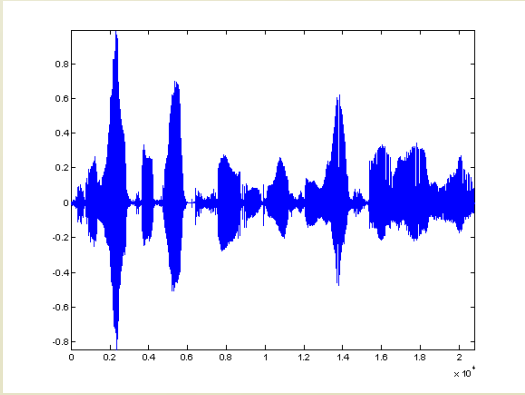
# *Detour:* Stochastic Gradient Descent

- Algorithm that performs updates after each example
  - Initialize $\boldsymbol{w}$
  - For N iterations
    - For each training example $\left(\mathbf{x}^{(n)}, t^{(n)}\right)$
      - $\Delta = \nabla_{\boldsymbol{w}} E\left(f\left(\mathbf{x}^{(n)}; \boldsymbol{w}\right), t^{(n)}\right)$
      - $\boldsymbol{w} \leftarrow \boldsymbol{w} - \eta\Delta$
- Gradient descent vs. Stochastic gradient descent
  - Gradient descent
    - $w \leftarrow w - \eta\nabla_{\boldsymbol{w}} E(f(\boldsymbol{x}; \boldsymbol{w}), \boldsymbol{t}) = w - \eta\frac{1}{N}\sum_{i=1}^{N} \nabla_{\boldsymbol{w}} E\left(f\left(\mathbf{x}^{(n)}; \boldsymbol{w}\right), t^{(n)}\right)$
    - Compute the gradient with all instances
  - Stochastic gradient descent
    - $w \leftarrow w - \eta\nabla_{\boldsymbol{w}} E\left(f\left(\mathbf{x}^{(n)}; \boldsymbol{w}\right), t^{(n)}\right)$
    - Compute the gradient with each instance
    - Approximation of the true gradient with all instances
  - Mini-batch : using a part of instances

Copyright © 2012 by Il-Chul Moon, Dept. of Industrial and Systems Engineering, KAIST

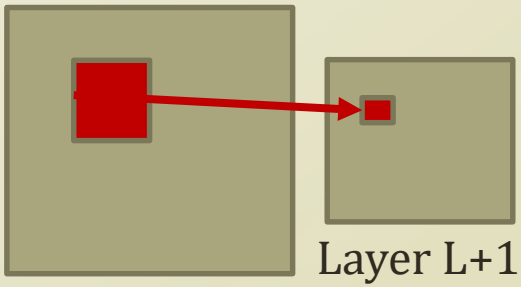
# Neural Networks for Various Domains

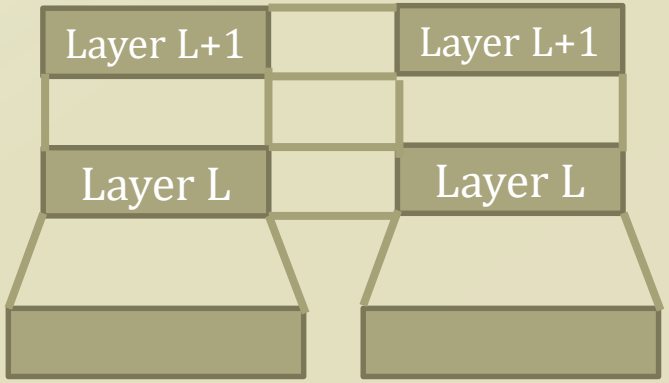
- Previous structure : Fully connected networks
- Recent advances in neural networks
  - Computer vision
  - Language models
- Application domain influences network structures
  - Convolutional structure utilizing localized connections
  - Recurrent structure utilizing chain connections



Speech Recognition


Image Classification
Object Recognition




Copyright © 2012 by Il-Chul Moon, Dept. of Industrial and Systems Engineering, KAIST

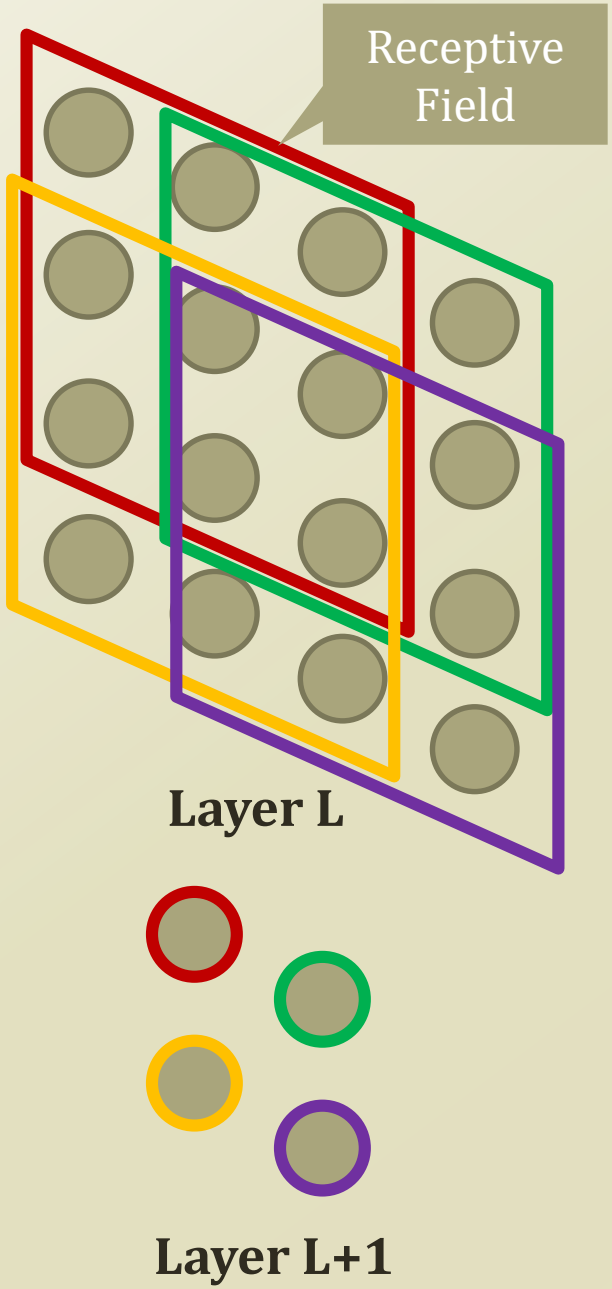
# Example : Computer Vision

- Vision problem characteristics
  - High dimensional inputs
    - 150 X 100 pixels = 15000 pixels
    - 15000 pixels X 3 RGB channels = 45000 real value inputs
  - 2D topology of pixels for images and 3D topology of pixels for video
  - Need to handle invariance
    - Rotation, Translation, Scale…
- Convolutional neural networks mitigates these characteristics
  - Local connectivity
  - Convolution
  - Pooling


Copyright © 2012 by Il-Chul Moon, Dept. of Industrial and Systems Engineering, KAIST

# Local Connectivity

- Use a local connectivity of hidden units
  - Each hidden unit is connected to only to a sub-region of the input images
  - Connected to all channels : 1 channel of grayscale and 3 RGB channel of color images
- Advantages of local connectivity
  - # of a fully connected layer >>> # of a sparsely connected Layer
    - More number of parameters to optimize
      - More data, more difficulties in learning
  - How to sparsely connect?
    - Utilizing the 2D topologies : Receptive field




Copyright © 2012 by Il-Chul Moon, Dept. of Industrial and Systems Engineering, KAIST

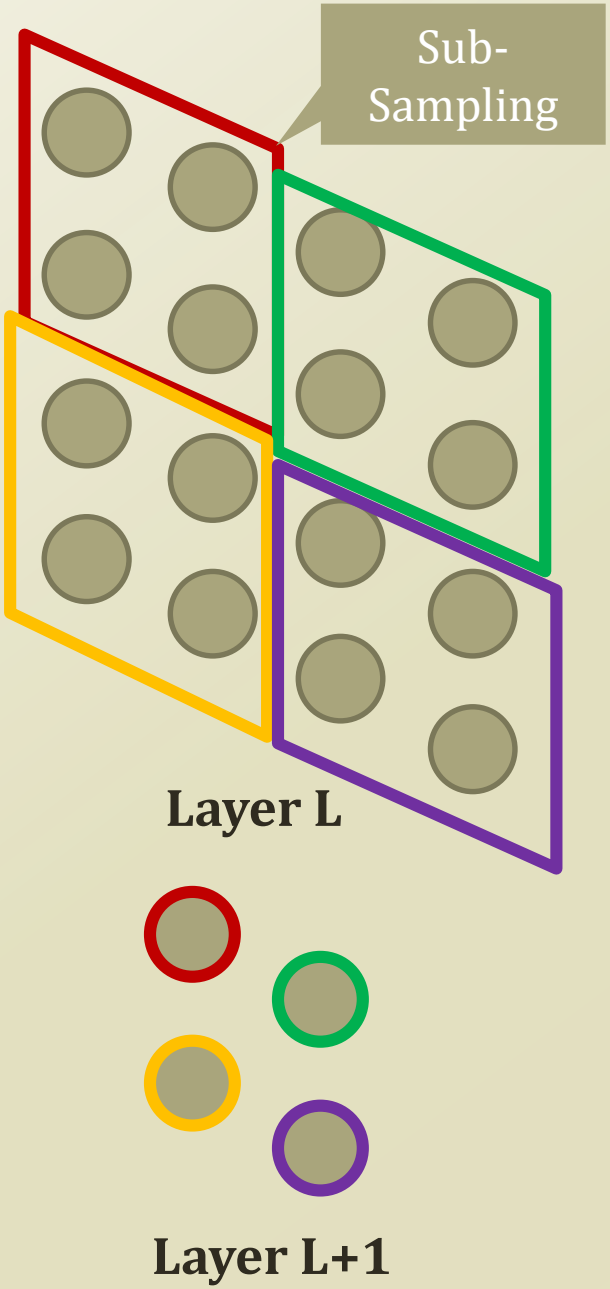
# Convolution Layer

- Each feature map forms a 2D grid of features
  - Can be computed with a discrete convolution (*) of a kernel matrix $k_{ij}$
    - Which is the hidden weights matrix $W_{ij}$ with its rows and columns flipped
    - The kernel can be matched to the receptive field dimension
- Mathematical definition on convolution
  - L-th layer has the neuron with $n_1 \times n_2$
  - Kernel matrix, $K \in R^{(2h_1+1)\times(2h_2+1)}$
  - $(I * K)_{r,s} := \sum_{u=-h_1}^{h_1} \sum_{v=-h_2}^{h_2} K_{u,v} I_{r+u,s+v}$
    - There are exception cases when we compute the convolution at the edge of the images with the negative indexes
      - Limit the convolution range to avoid the negative indexes
      - Padding values, usually zero (so called zero padding) to the edge of the layer
- Convolution layer
  - $(Y_i^{(l)})_{r,s} = \sigma\left(\left(B_i^{(l)}\right)_{r,s} + \sum_{j=1}^{m_1^{(l-1)}} \sum_{u=-h_1^{(l)}}^{h_1^{(l)}} \sum_{v=-h_2^{(l)}}^{h_2^{(l)}} \left(K_{i,j}^{(l)}\right)_{u,v} \left(Y_j^{(l-1)}\right)_{r+u,s+v}\right)$
    - $m_2^{(l)} = \frac{m_2^{(l-1)} - 2h_1^{(l)}}{s_1^{(l)}+1}$, $m_3^{(l)} = \frac{m_3^{(l-1)} - 2h_2^{(l)}}{s_2^{(l)}+1}$
  - $l$ : layer index
  - $r, s$ : 2D neuron index at layer $l$
  - $m_1^{(l-1)} \times m_2^{(l-1)} \times m_3^{(l-1)}$ : Dimension of layer (*l-1)* by Depth X Width X Height
  - $s_1^{(l)}, s_2^{(l)}$ *:* Stride (skipping factor) of the kernel matrix on the width and the height dim.

Copyright © 2012 by Il-Chul Moon, Dept. of Industrial and Systems Engineering, KAIST

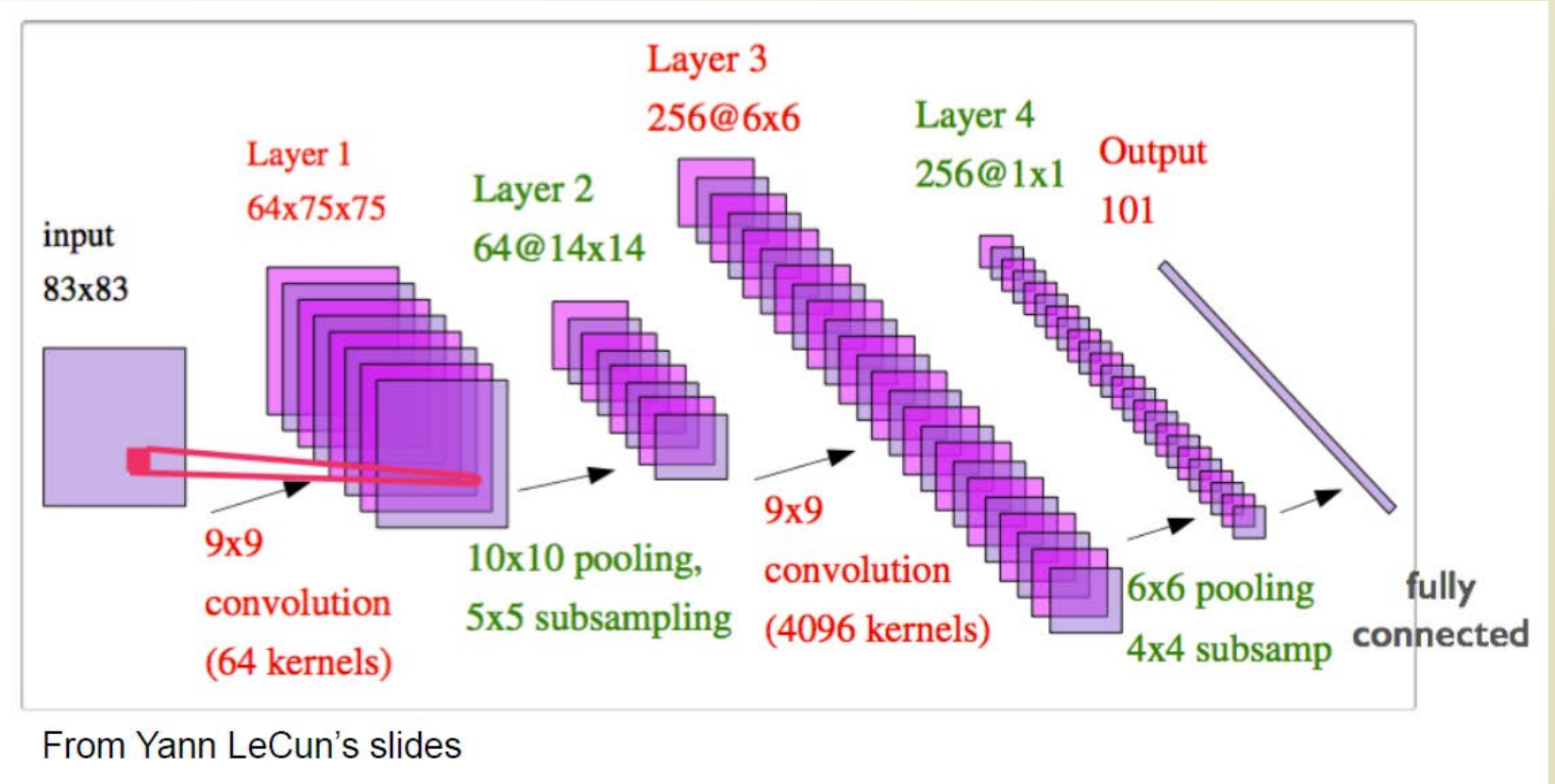
# Pooling Layer

- Pool hidden units in the same neighborhood
  - Introduce invariance to local translations
- Definition of various pooling
  - $(Y_i^{(l)})_{r,s} = f\left(\left(Y_i^{(l-1)}\right)_{r-h_1,s-h_2}, \ldots, \left(Y_i^{(l-1)}\right)_{r+h_1,s+h_2}\right)$
  - Max pooling : $f(Y_i) = \max(Y_i)$
  - Average pooling : $f(Y_i) = \frac{1}{h_1 \times h_2} \sum Y_i$




Copyright © 2012 by Il-Chul Moon, Dept. of Industrial and Systems Engineering, KAIST

# Example of Convolution Neural Network




From Yann LeCun's slides

- Should be able to read the structure presented as the above diagram
- Convolution layer, pooling layer, fully connected layer
- Require to use diverse kernels and carefully calibrated hyperparameters
- Still learnable with the back-propagation with some techniques

Copyright © 2012 by Il-Chul Moon, Dept. of Industrial and Systems Engineering, KAIST

# *Detour:* Handling Datasets

- Characteristics of Image datasets
  - Real valued : 0~255 → Often be normalized with mean and std.
  - Multi channeled : RGB 3 channel
  - Dense : no exact zero cases of the data
  - Topologically structured : 2D of image, 3D of video (width*height*time)
- Characteristics of Text datasets
  - Categorical value : “I”, “ate”, “ramen”
    - Need an encoding to represent
    - Typical encoding scheme : One-Hot encoding
      - 10 unique words in the entire corpus, $4^{th}$ word, $w$, appears
        - $e(w) = [0,0,0,1,0,0,0,0,0,0]$
        - No assumption on word similarity
        - Sparse when encoded
        - Becomes a very high dimension : # of unique word can easily go over 10,000
  - Single channel

Copyright © 2012 by Il-Chul Moon, Dept. of Industrial and Systems Engineering, KAIST

# *Detour:* Word2Vec

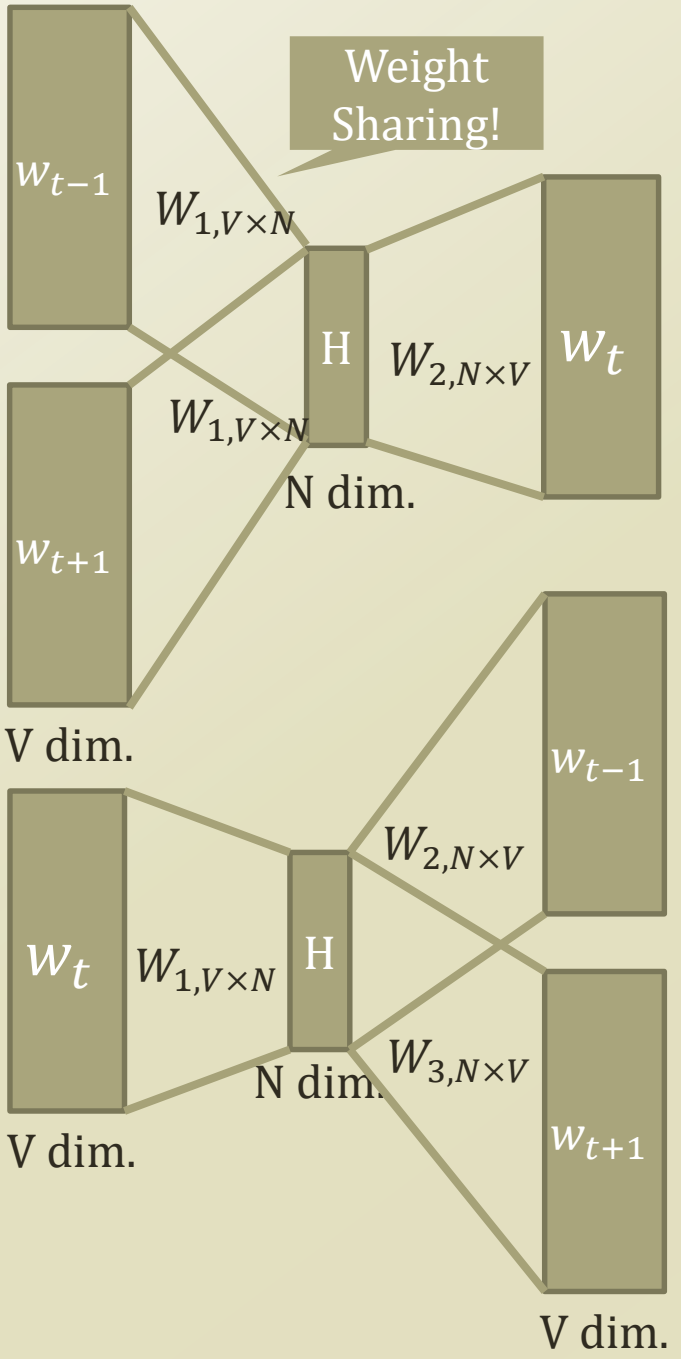
- Word2Vec
  - A continuous and low dimensional representation of words
- Modeling approaches
  - Continuous Bag-of-Words
    - Predict a word with surrounding words in a window of length $2m$
    - Learning objective function:

$$J(\theta) = \frac{1}{T}\sum_{t=1}^{T}\sum_{-m\le j\le m, j\neq 0} \log P(w_t|w_{t+j},\theta)$$

  - Continuous Skip gram
    - Predict surrounding words with a current word in a window of length $2m$
    - Learning objective function:

$$J(\theta) = \frac{1}{T}\sum_{t=1}^{T}\sum_{-m\le j\le m, j\neq 0} \log P(w_{t+j}|w_t,\theta)$$

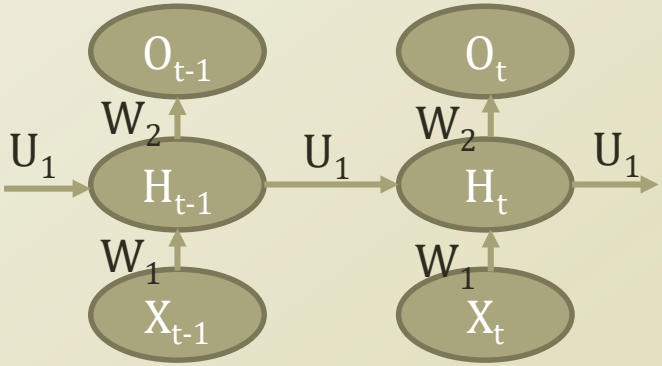
- $P(w_{t+j}|w_t,\theta)$ and $P(w_t|w_{t+j},\theta)$
  - Can be modeled as a neural network structure
    - With a single hidden layer
    - With a soft-max output function
  - Hidden layer becomes the low dimensional representation
- Introduction of negative sampling
  - $\log P\left(w_{t+j}\middle|w_t,\theta\right) + \sum_{i=1}^{k} E_{w_i\sim P_N(w)}\left(1 - \log P\left(w_{t+j}\middle|w_i,\theta\right)\right)$
  - Only positive example cannot set the decision boundary

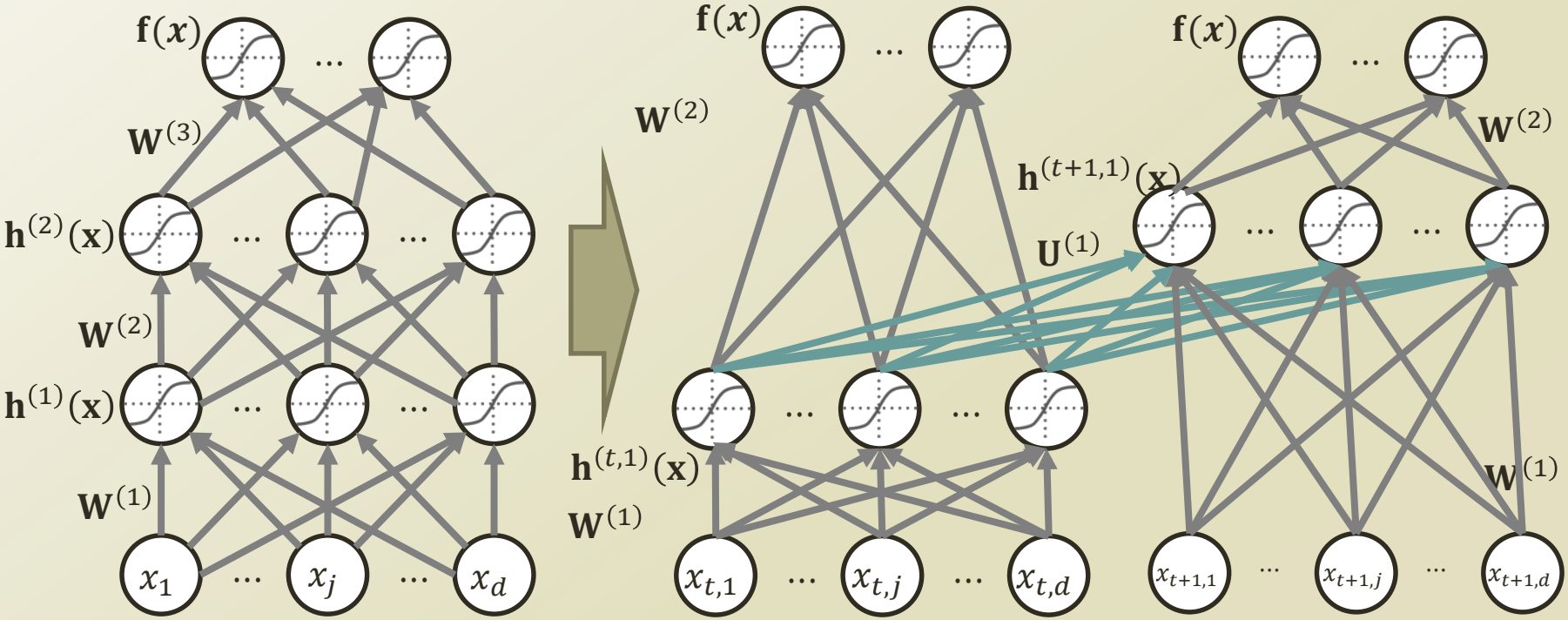



Copyright © 2012 by Il-Chul Moon, Dept. of Industrial and Systems Engineering, KAIST

# Modeling Temporal Data with NN

- Limitation of convolutional neural network
  - No temporal modeling in the input and the hidden information
  - Need to model the information flow from the previous hidden layer
- The definition of Recurrent Neural Network
  - $h_t = \sigma(Uh_{t-1} + Wx_t + b)$

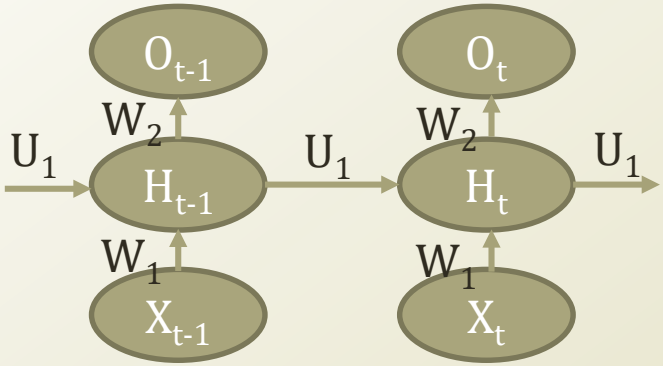


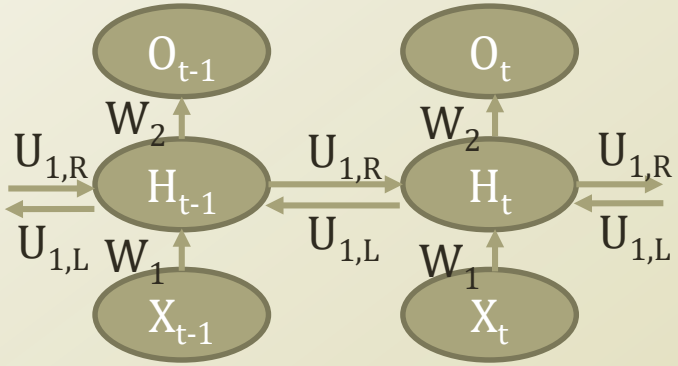

Copyright © 2012 by Il-Chul Moon, Dept. of Industrial and Systems Engineering, KAIST

# Variant of Recurrent Neural Network

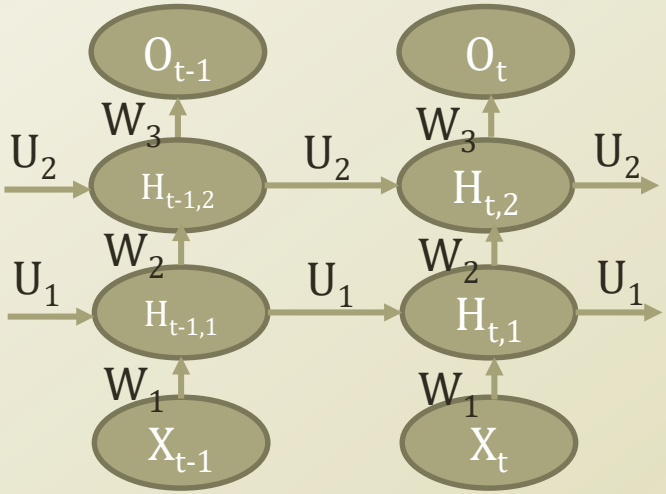
- Bidirectional RNN for influence from $X_{t+1}$ to $X_t$
- Deep RNN for further complex modeling on X and H
- Sequence-to-Sequence RNN when generate an output after the whole sequence input



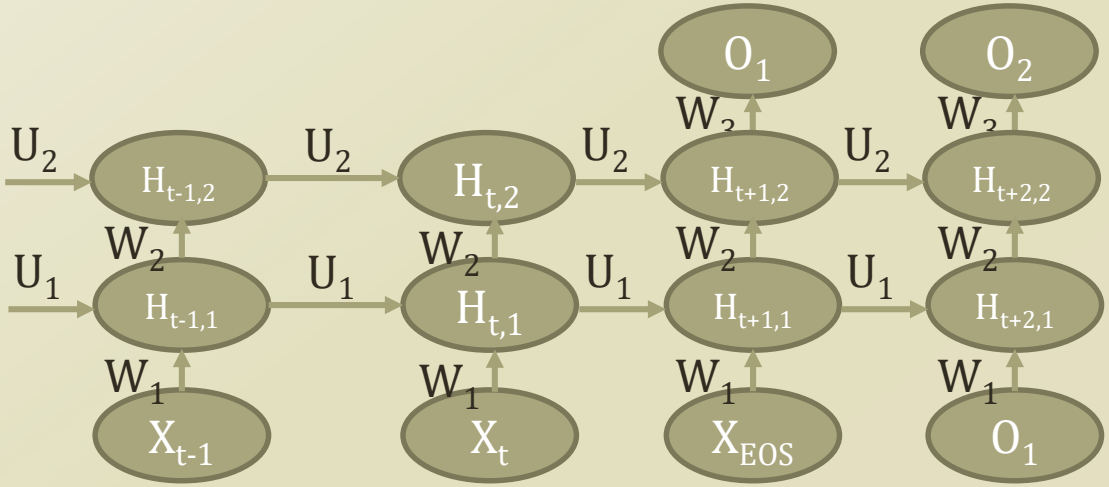

Recurrent Neural Network




Bidirectional Recurrent Neural Network




Deep Recurrent Neural Network




Deep Sequence-to-Sequence Recurrent Neural Network

Copyright © 2012 by Il-Chul Moon, Dept. of Industrial and Systems Engineering, KAIST

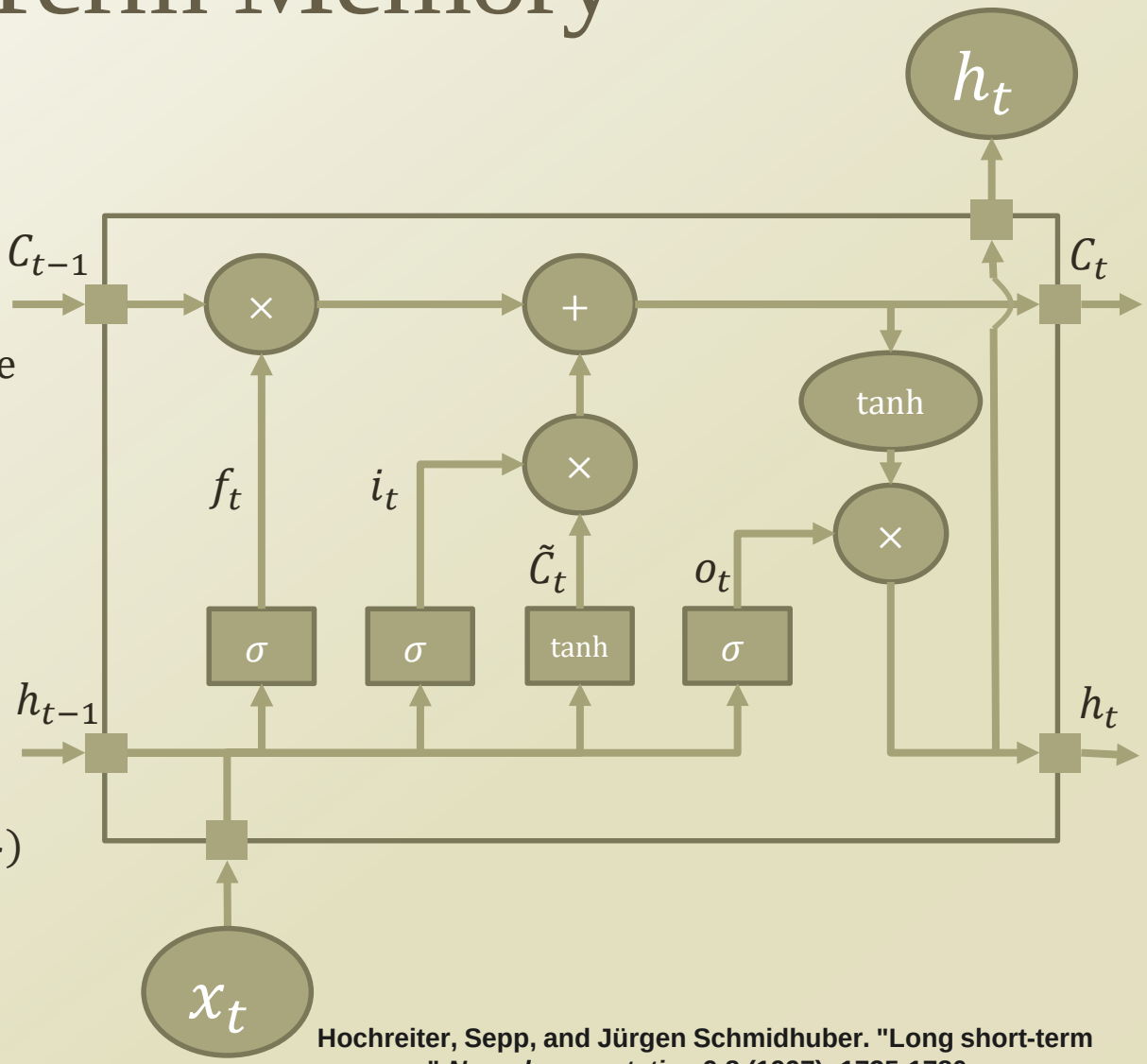
# Vanishing Gradient Problem

- Logistic function : $f(x) = \frac{1}{1+e^{-x}}$
  - $\frac{d}{dx} f(x) = \frac{d}{dx}(1+e^{-x})^{-1} = e^{-x}(1+e^{-x})^{-2} = f(x)(1-f(x))$
  - $0 < f(x) < 1$
  - $max \frac{d}{dx} f(x) = \frac{1}{4}$
- Multi-layered delta signal
  - $\delta_j = o_j(1-o_j)\sum_k \delta_k w_{jk}$
- Back-propagation algorithm
  - $w_{ij}^{t+1} \leftarrow w_{ij}^{t} + \eta o_i \delta_j$
- If a delta signal is weak, the weight update is impossible
  - Deep layered neural network
  - Single layered recurrent neural network

Copyright © 2012 by Il-Chul Moon, Dept. of Industrial and Systems Engineering, KAIST

# Long Short Term Memory

- LSTM cell
  - Introducing a state to the LSTM cell
  - Gate mechanism to add or remove information from the cell state
- Gate mechanism
  - Sigmoid activation
  - Pointwise multiplication
- Mathematical formulation
  - $f_t = \sigma\left(W_f[h_{t-1}, x_t] + b_f\right)$
  - $i_t = \sigma(W_i[h_{t-1}, x_t] + b_i)$
  - $\tilde{C}_t = tanh(W_C[h_{t-1}, x_t] + b_C)$
  - $C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$
  - $o_t = \sigma(W_o[h_{t-1}, x_t] + b_o)$
  - $h_t = o_t * \tanh(C_t)$

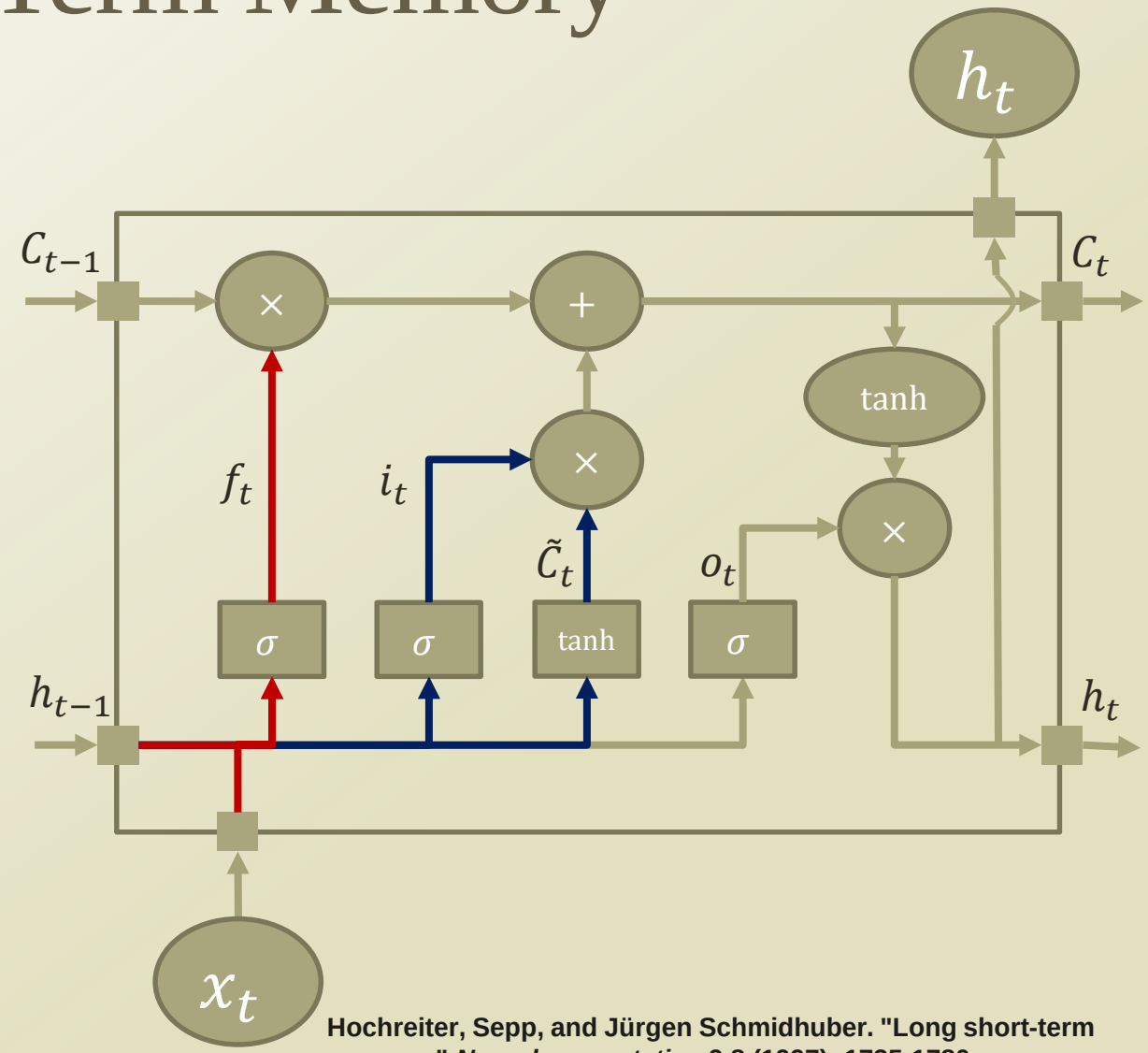



Hochreiter, Sepp, and Jürgen Schmidhuber. "Long short-term memory." *Neural computation* 9.8 (1997): 1735-1780.

Copyright © 2012 by Il-Chul Moon, Dept. of Industrial and Systems Engineering, KAIST

# Long Short Term Memory

- Forgetting process
  - $f_t = \sigma(W_f[h_{t-1}, x_t] + b_f)$
    - $f_t$ : Forget gate layer
    - Forgetting from the state of the last cell, $C_{t-1}$
    - Number of outputs equal to the cell state dimension
- Remembering process
  - $i_t = \sigma(W_i[h_{t-1}, x_t] + b_i)$
    - $i_t$ : Remember gate layer
    - Output dim.==Cell state dim.
  - $\tilde{C}_t = tanh(W_C[h_{t-1}, x_t] + b_C)$
    - $\tilde{C}_t$ : Information to be remembered
    - Output dim.==Cell state dim.

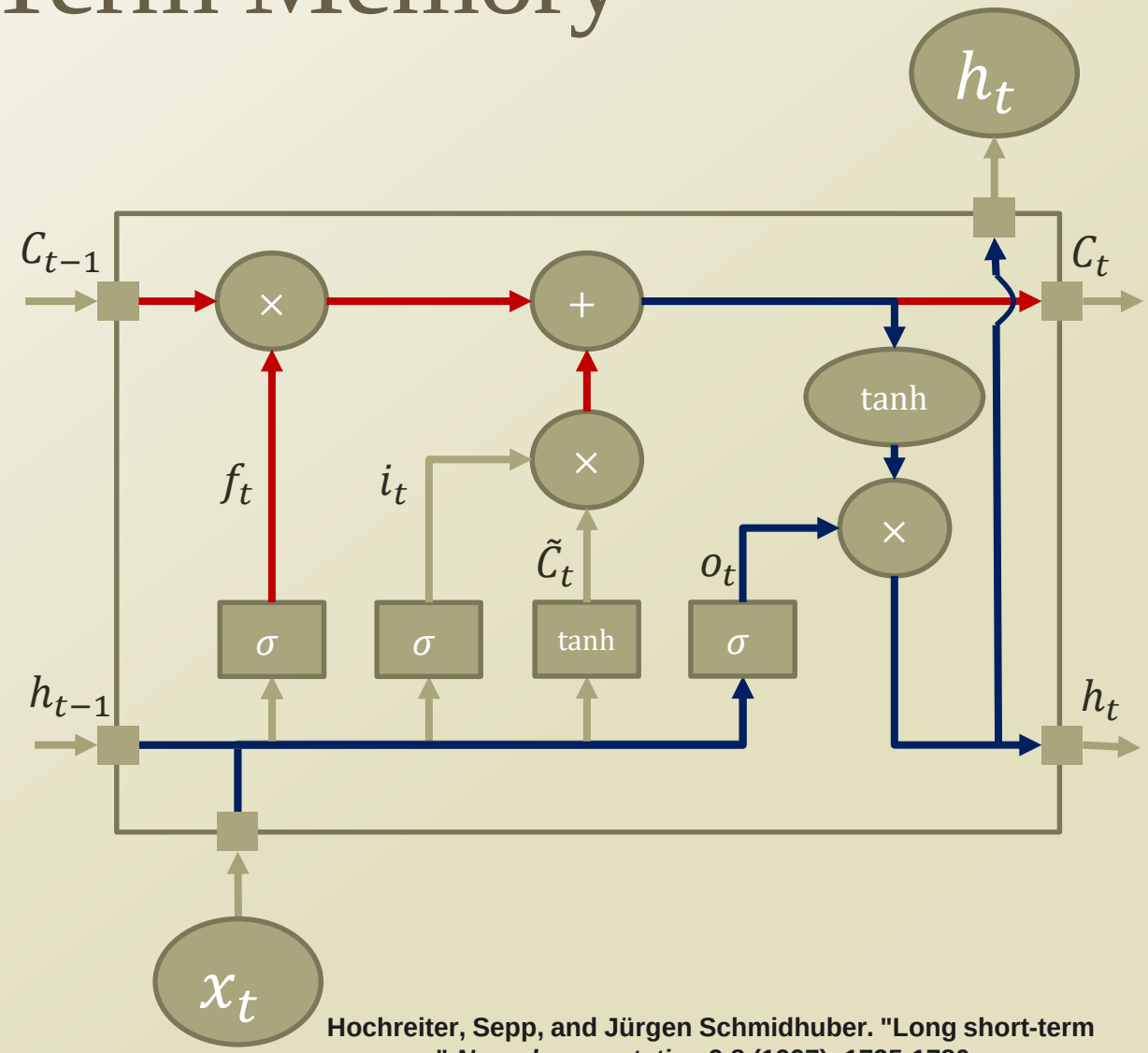



Hochreiter, Sepp, and Jürgen Schmidhuber. "Long short-term memory." *Neural computation* 9.8 (1997): 1735-1780.

Copyright © 2012 by Il-Chul Moon, Dept. of Industrial and Systems Engineering, KAIST

# Long Short Term Memory

- Application to the cell state
  - $C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$
    - Forgetting process
    - Remembering process
- Output process
  - $o_t = \sigma(W_o[h_{t-1}, x_t] + b_o)$
    - Output depends upon the state
    - Should be filtered by the past output and the current input
  - $h_t = o_t * \tanh(C_t)$
    - Squash the cell state to fit the range between [-1,1]




Hochreiter, Sepp, and Jürgen Schmidhuber. "Long short-term memory." *Neural computation* 9.8 (1997): 1735-1780.

Copyright © 2012 by Il-Chul Moon, Dept. of Industrial and Systems Engineering, KAIST

# Pros and Cons of LSTM

- LSTM enables
  - The long range information delivery through the cell state
  - Cell state does not require a layer propagation
- Problem of LSTM
  - Too many parameters to learn
    - $W_f, W_i, W_c, W_o$
    - Many new hazards to make the state propagation work
- Variants of LSTM
  - Gated recurrent unit
    - $z_t = \sigma(W_z x_t + U_z h_{t-1} + b_z)$
    - $r_t = \sigma(W_r x_t + U_r h_{t-1} + b_r)$
    - $h_t = z_t * h_{t-1} + (1 - z_t) * \tanh(W_h x_t + U_h(r_t * h_{t-1}) + b_h)$
    - No separate storage of information through the cell state
    - No concatenation of input and hidden information

Copyright © 2012 by Il-Chul Moon, Dept. of Industrial and Systems Engineering, KAIST

# TECHNIQUES IN NEURAL NETWORKS

Copyright © 2012 by Il-Chul Moon, Dept. of Industrial and Systems Engineering, KAIST

# Many Techniques in Neural Networks

- Neural network requires diverse techniques
  - Training process to facilitate the learning
    - Batch normalization
    - Early stopping
    - Adversarial training
    - Momentum in optimization, adaptive learning rate, gradient checking …
  - Structure to facilitate the learning
    - Attention network
    - Highway network
    - Dropout….
  - Some are originated from experiments, and others are motivated by theory
    - A few found to be useful by experiments, and later, they are revealed to be meaningful, theoretically
- Inference is limited to the stochastic gradient descent

Copyright © 2012 by Il-Chul Moon, Dept. of Industrial and Systems Engineering, KAIST

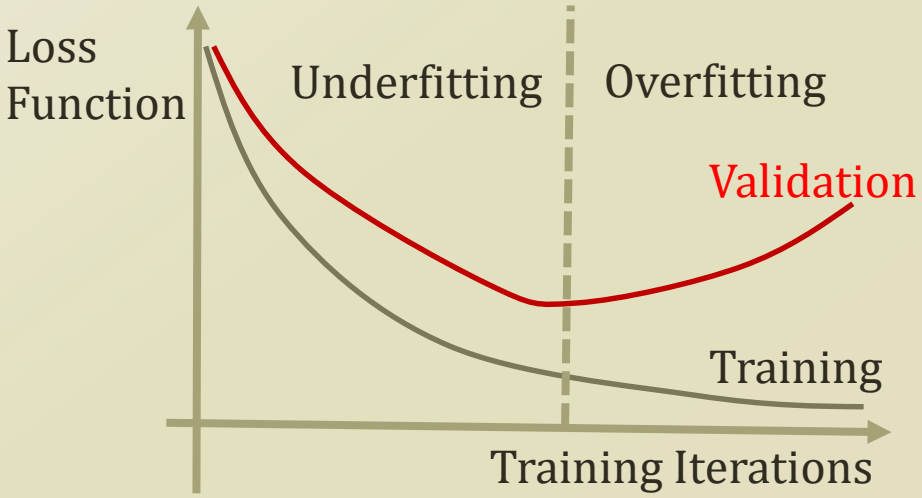
# Model Selection

- Training processes
  - Given a dataset, $D$
  - Divide $D$ into three datasets
    - Training set : $D^{\text{train}}$
      - To infer the parameter of the model to train
    - Validation set : $D^{\text{valid}}$
      - To search the hyper-parameter
        - Hidden layer size, learning rate, number of iterations
    - Test set : $D^{\text{test}}$
      - Actual testing
- In the real world,
  - Train the model with the training and the validation set
    - Deployment domain requires adaptations
      - Training : development phase
      - Validation : deployment phase
  - Test the model with the testing set
    - Evaluate the actual performance after the deployment

Copyright © 2012 by Il-Chul Moon, Dept. of Industrial and Systems Engineering, KAIST

# Early Stopping

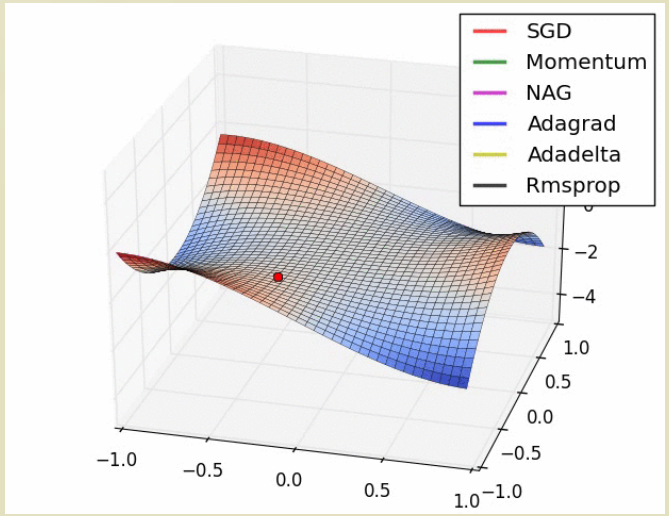
- Neural network model is naturally complex with many parameters
  - Very easy to overfit
  - Stop the fitting before the model overfits
- Training : the model parameter inference
- Validation : early stopping point check




Copyright © 2012 by Il-Chul Moon, Dept. of Industrial and Systems Engineering, KAIST

# Momentum and Adaptive Learning Rate

- Gradient Descent : $\theta_{t+1} = \theta_t - \eta \nabla_\theta l\left(f\left(x^{(t)}\right), y^{(t)}; \theta_t\right)$
- Momentum
  - Exponential average of the previous gradients
    - Move along the direction to overcome the plateau
    - $\beta$ becomes the decay factor
    - $\theta_{t+1} = \theta_t - \eta \hat{\nabla}_\theta^{(t)}$
    - $\hat{\nabla}_\theta^{(t)} = \nabla_\theta l\left(f\left(x^{(t)}\right), y^{(t)}; \theta_t\right) + \beta \hat{\nabla}_\theta^{(t-1)}$
- Adaptive Learning Rate
  - Adagrad : learning rate scaled by the square root of the cumulative sum of squared gradients
    - More movements on the smaller movements, less movements on the larger movements
    - $\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{G_t + \epsilon}} \nabla_\theta l\left(f\left(x^{(t)}\right), y^{(t)}; \theta_t\right)$
    - $G_t = G_{t-1} + \left(\nabla_\theta l\left(f\left(x^{(t)}\right), y^{(t)}; \theta_t\right)\right)^2$
  - RMSProp : Exponential moving average of the squared gradients
    - $\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{G_t + \epsilon}} \nabla_\theta l\left(f\left(x^{(t)}\right), y^{(t)}; \theta_t\right)$
    - $G_t = \gamma G_{t-1} + (1-\gamma)\left(\nabla_\theta l\left(f\left(x^{(t)}\right), y^{(t)}; \theta_t\right)\right)^2$
  - Adam : RMSProp + Momentum
    - $\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{G_t + \epsilon}} \hat{\nabla}_\theta^{(t)}$
    - $\hat{\nabla}_\theta^{(t)} = (1-\beta_1)\nabla_\theta l\left(f\left(x^{(t)}\right), y^{(t)}; \theta_t\right) + \beta_1 \hat{\nabla}_\theta^{(t-1)}$
    - $G_t = \gamma G_{t-1} + (1-\gamma)\left(\nabla_\theta l\left(f\left(x^{(t)}\right), y^{(t)}; \theta_t\right)\right)^2$




Copyright © 2012 by Il-Chul Moon, Dept. of Industrial and Systems Engineering, KAIST

# Mini-Batch and Gradient Checking

- Backpropagation algorithm
  - Single instance gradient calculation
    - $w \leftarrow w - \eta \nabla_{\boldsymbol{w}} E\left[l\left(f\left(\mathbf{x}^{(i)}; \boldsymbol{w}\right), y^{(i)}\right)\right]$
    - With a single instance, not much to calculate as expectation
  - Full instance gradient calculation
    - $w \leftarrow w - \eta \nabla_{\boldsymbol{w}} E\left[l\left(f\left(\boldsymbol{x}; \boldsymbol{w}\right), \boldsymbol{y}\right)\right]$
    - Normalize the summation with the size of the dataset
  - Should be mini-batch
    - $w \leftarrow w - \eta \nabla_{\boldsymbol{w}} E\left[l\left(f\left(\mathbf{x}^{(i;i+n)}; \boldsymbol{w}\right), y^{(i;i+n)}\right)\right]$
    - Normalize the summation with the size of the mini-batch
- Gradient checking
  - When you implement the propagation algorithm, compare the finite-difference approximation of the gradient
    - $\frac{\partial f(x)}{\partial x} \approx \frac{f(x+\epsilon)-f(x-\epsilon)}{2\epsilon}$

Copyright © 2012 by Il-Chul Moon, Dept. of Industrial and Systems Engineering, KAIST

# Batch Normalization

- Layer type output normalization with mini-batch data
  - BatchNormalization$(x_{1\ldots m}; \gamma, \beta)$
    - $\mu_B \leftarrow \frac{1}{m}\sum_{i=1}^{m} x_i$
    - $\sigma_B^2 \leftarrow \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu_B)^2$
    - $\hat{x}_i \leftarrow \frac{x_i - \mu_B}{\sqrt{\sigma_B^2 + \epsilon}}$
    - $y_i \leftarrow \gamma \hat{x}_i + \beta \equiv BN_{\gamma,\beta}(x_i)$
- Batch normalization parameter, $\gamma$ and $\beta$, is trainable through stochastic gradient descent, as well
  - Can use chain rule
- How to use it?
  - Training : After the regression, and before the activation
  - Testing : remember the mean of $\gamma$s and $\beta$s from mini-batches
- Why use it?
  - Scale and shift normalization

Copyright © 2012 by Il-Chul Moon, Dept. of Industrial and Systems Engineering, KAIST

# Adversarial Training

- Instances in datasets
  - Some instances work well with the trained model
  - Some instances do not work well with the trained model
    - Some instances should not be accommodated because it could be noise
    - Some instances should be accommodated because it is a meaningful case
- Find an (or artificially generate) instance that disrupt the current trained model
  - "Training on adversarial example"
  - $w^T\tilde{x} = w^T x + w^T\eta, ||\eta||_\infty < \epsilon$
    - $\tilde{x}$ : adversarial example, $w^T\eta$ : adversarial perturbation
    - $\eta^* = max_\eta w^T\eta = sign(w)$
    - Can make many infinitesimal changes, "Accidental steganography"
- Given $y \in \{-1,1\}, P(y=1) = \sigma(w^T x + b), g(z) = \log(1+\exp(z))$
  - Training objective
    - $min_w E_{x,y\sim P_{data}} g(-y(w^T x + b))$
  - Adding the adversarial training
    - $min_w E_{x,y\sim P_{data}} g(-y(w^T x + b - \epsilon||w||_1))$
      - $-\left|\left|w\right|\right|_1 = -w^T sign(w)$, because of the minimization objective
  - Similar to the L1 regularization, but it is different
    - Not directly applied as a penalty that does not go away.
    - Good fitting of *w* can eliminate the adversarial learning term
    - L1 regularization → controlling the sensitivity
    - Adversarial training → robust weight vector

Copyright © 2012 by Il-Chul Moon, Dept. of Industrial and Systems Engineering, KAIST

# Dropout

- Challenge of the neural network
  - Complex model to learn a simple principle
  - Limit the complexity, stochastically
    - Remove a set of randomly selected neurons for a certain mini-batch
- Rational
  - Hidden units cannot co-adapt to other units that were active in the mini-batch gradient descent
  - Hidden units must be more generally useful
- By defining a mask, $r^{(l)}, l = 1 \dots L$
  - Neural network without dropout
    - $z_i^{(l+1)} = w_i^{(l+1)} y^{(l)} + b_i^{(l+1)}, y_i^{(l+1)} = f(z_i^{(l+1)})$
  - Neural network with dropout
    - $r_j^{(l)} \sim Bernoulli(p)$
    - $\tilde{y}^{(l)} = r^{(l)} * y^{(l)}$
    - $z_i^{(l+1)} = w_i^{(l+1)} \tilde{y}^{(l)} + b_i^{(l+1)}, y_i^{(l+1)} = f(z_i^{(l+1)})$
- Marginalizing the dropout in the linear regression
  - $min_w E_{R\sim Bernoulli(p)} \left[ \left|\left| y - (R * X)w \right|\right|^2 \right] = min_w \left[ \left|\left| y - pXw \right|\right|^2 + p(1-p) \left|\left| \Gamma w \right|\right|^2 \right]$
    - $\Gamma = (diag(X^T X))^{1/2}$
  - $= min_w \left[ \left|\left| y - X\tilde{w} \right|\right|^2 + \frac{1-p}{p} \left|\left| \Gamma \tilde{w} \right|\right|^2 \right]$
    - $\tilde{w} = wp$

Copyright © 2012 by Il-Chul Moon, Dept. of Industrial and Systems Engineering, KAIST

# Acknowledgements


- This lecture is influenced and adopted materials from 10807 Topics in Deep Learning by Prof. Russ Salakhutdinov, Carnegie Mellon University.


Copyright © 2012 by Il-Chul Moon, Dept. of Industrial and Systems Engineering, KAIST