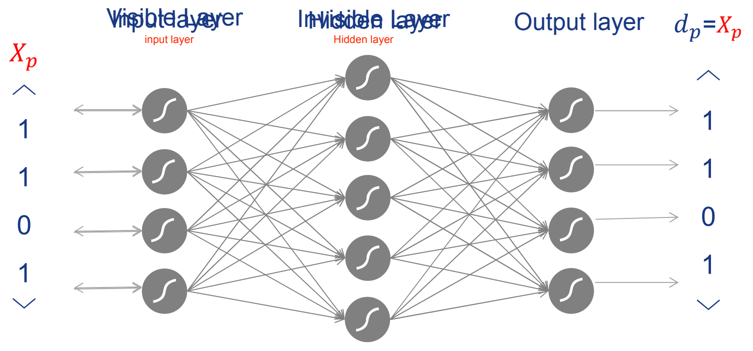
6. Deep Learning Model - Quick View

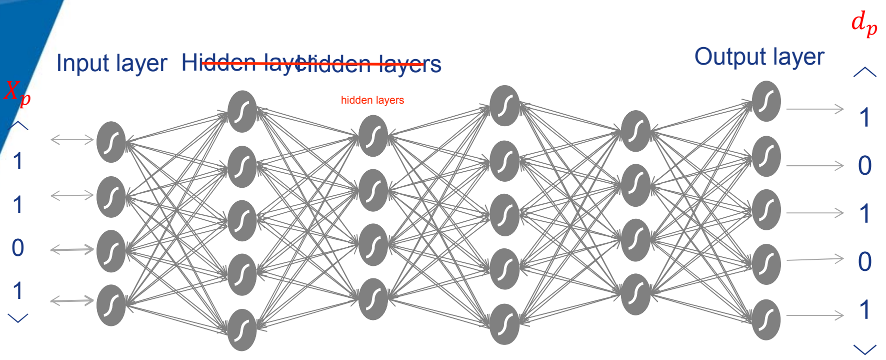
Restricted Voltzmann Machine

- **Learning? Backpropagation Algorithm**



Deep Neural Network

belief



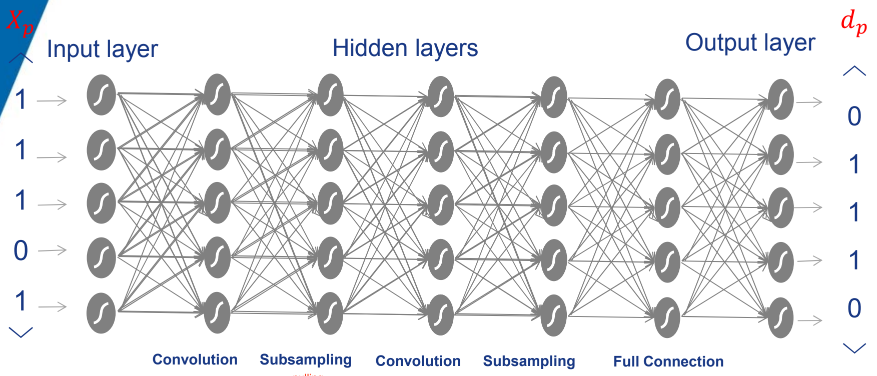
Multi-level Backpropagation Learning

PPT / capture
Revisited-Multi-Layered Backpropagation Neural Network 참고

인접한 두개 중에 가장 숫자값이 크게 나오는 것만 다음 레이어에 쓰겠다는 것. (weighted sum대신에 max함수를 쓴다)

Convolutional Neural Network

Revisited- mUlti-layered backpropagation Neural Network

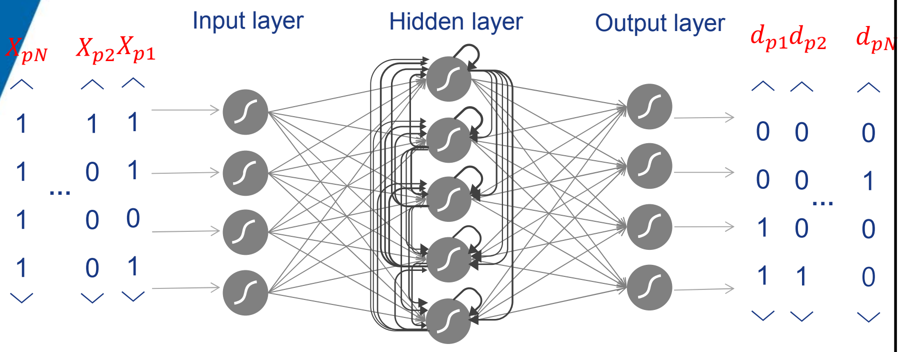


뒤의 2~4레이어
풀 커넥션돼있는쪽은
fully connected되어있는 것이다 (앞쪽은 partically connected)

Multi-level Backpropagation Learning

Recurrent Neural Network

- **Backpropagation Learning Algorithm**



Time-series Training Data

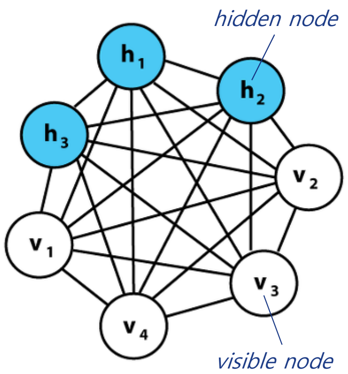
Restricted Boltzmann Machine (RBM)

Introduction

- **Boltzmann Machine**

- 모든 node (visible node, hidden node) 가 서로 연결되어 있는 구조
- 비지도 학습 모델
- 순회판매원 문제, 최적화 문제의 근사해 구하는 경우에 적합

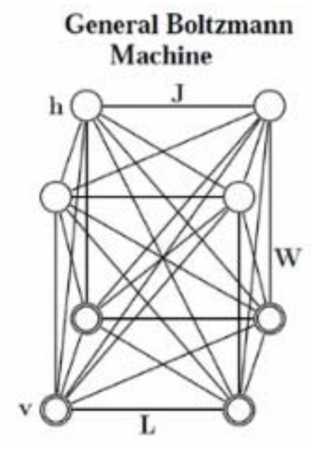
난해해지는 문제가 있다



Introduction

- **Boltzmann Machine**

- 모든 node (visible node, hidden node) 가 서로 연결되어 있는 구조
- 비지도 학습 모델
- 순회판매원 문제, 최적화 문제의 근사해 구하는 경우에 적합
- **하지만 복잡한 구조로 인해 실제적인 문제 해결에 유용하지 않음**

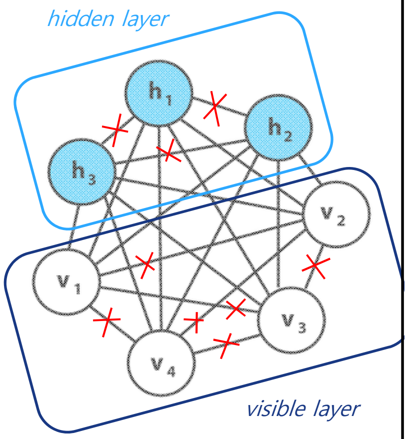


Concept

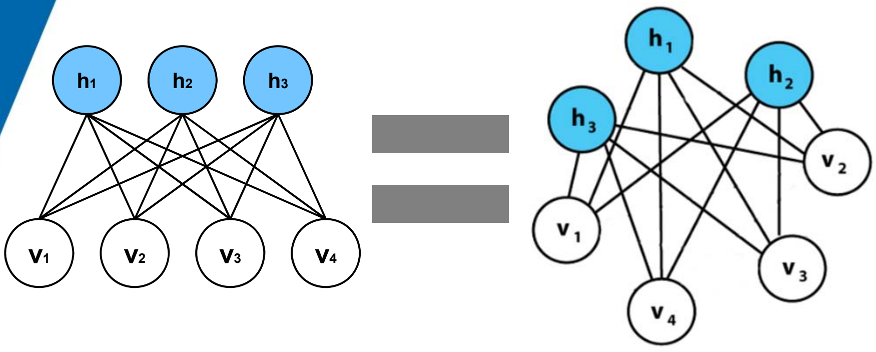
- **“Restricted”**

- 같은 layer 내부의 연결을 <u>제한함</u>을 의미

- 구조를 단순화하여 계산을 간단히 함으로써 학습을 빠르게 할 수 있음



Introduction



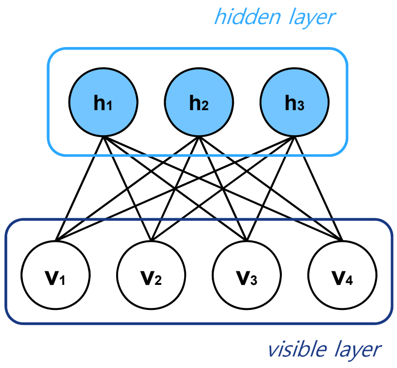
Introduction

- **Restricted Boltzmann Machine**

특징

- Visible layer, Hidden layer 두 개의 계층으로 구성
- 각 노드는 다른 레이어의 모든 노드와 연결
- 각 노드는 Binary 값만 가짐

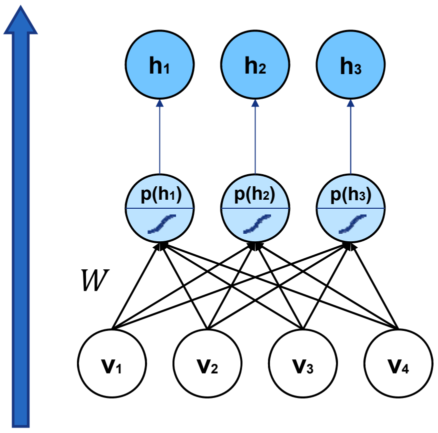
- 비지도 학습
- 관찰된 결과(visible layer)만으로 hidden layer의 분포를 예측하도록 학습



Learning

- **Feed Forward**

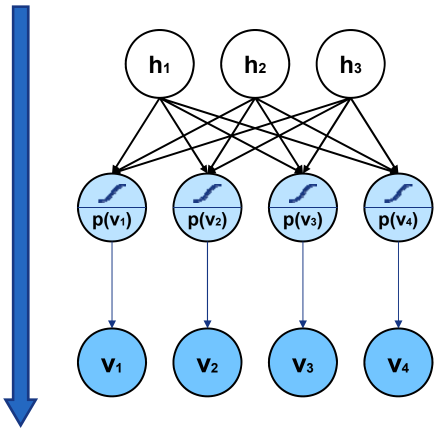
- 주어진 **v**에 대해 hidden node의 값을 구하는 과정
- $p(h_i|v) = sigm\ (vW_i + b_i)$
  v, 등과 같이 인풋 집어넣었을 때 결과값이 1이 될 확률?
- 각 노드에서 구해진 확률에 따라 hidden node는 최종적으로 0 또는 1의 값을 갖는다. 확률로 생각.
- W는 Generalized random value



Learning

- **Reconstruction (재구성)**

- 앞서 구한 **h** 값을 이용하여 visual node의 값을 구하는 과정
- $p(v_j|h) = sigm\left(hW_j^T + b_j\right)$
- 마찬가지로 각각의 visual node는 확률에 따라 0 또는 1의 값을 갖는다.

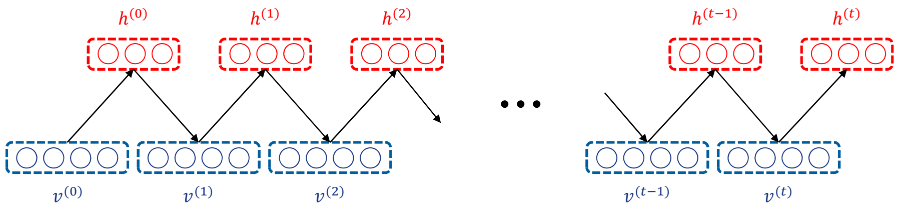


Learning

- **Weight update**

E(x,h) : energy function
필기 확인하기!

- $\frac{\partial \log p(v^{(0)})}{\partial w} = \langle h^0(v^0 - v^1)\rangle + \langle v^1(h^0 - h^1)\rangle + \langle h^1(v^1 - v^2)\rangle + \cdots$
- 수렴할 때까지 Feed Forward, Reconstruction 반복 ($v^{(t-1)} \approx v^{(t)}$)
- $\Delta \boldsymbol{W} = \gamma \cdot \left(v^{(0)^T} \cdot h^{(0)} - v^{(t)^T} \cdot h^{(t)}\right)$
  중간은 계속 상쇄가 된다
  - γ은 학습률
  - $v^T \cdot h \ni a_{ij} = count(v_i = 1, h_j = 1)$ Vi=1, Hj=1인 갯수를 Aij에 넣고 학습시킨다

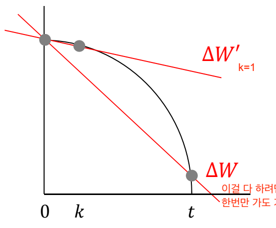


Learning

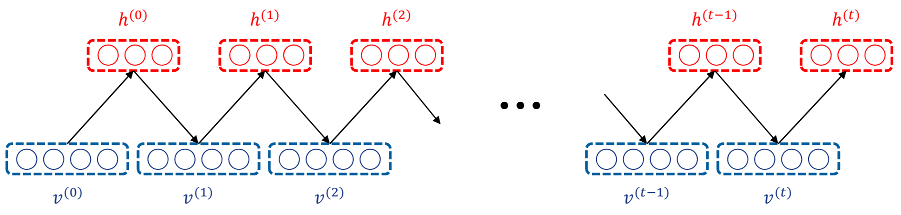
- **Gibbs Sampling**

- Weight update 한 번에 투자하는 연산의 양이 너무 많음
- 수렴할 때까지 반복시키지 않고 k번의 시행만으로 ΔW 계산
  → 경향성은 비슷할 것이므로
- RBM에서 주로 k = 1
- $\Delta \boldsymbol{W} = \gamma \cdot \left({v^{(0)}}^T \cdot h^{(0)} - {v^{(k)}}^T \cdot h^{(k)} \right)$

어차피 iteration반복이기때문에
길게할 필요 없고 k=1만 해줘도 된다



이걸 다 하려면 엄청난 샘플이 필요하지만
한번만 가도 기울기는 유지된다



AI Lab, Hanyang University

Learning

- **예제**

- Input data는 학생들이 좋아하는 과목의 분포
- 계산의 편의를 위하여 bias 생략
- 학습률 = 0.3
- Gibbs sampling의 step k = 1

	국어	수학	영어	사회	과학	한국사
지훈	1	0	1	0	0	0
영수	0	1	1	0	1	0
태희	1	0	1	1	0	1
동건	0	1	0	1	0	1
소영	0	1	0	0	1	0

AI Lab, Hanyang University

Learning

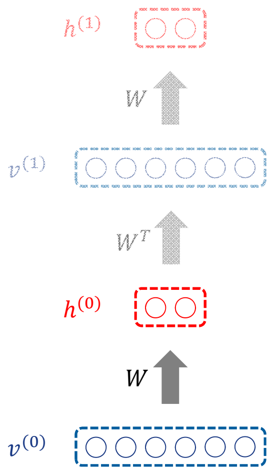
- **예제**

1. $p(\boldsymbol{h}|v) = sigm\ (v \cdot W)$

$$v = \begin{pmatrix} 1\,0\,1\,0\,0\,0 \\ 0\,1\,1\,0\,1\,0 \\ 1\,0\,1\,1\,0\,1 \\ 1\,1\,0\,1\,0\,1 \\ 0\,1\,0\,0\,1\,0 \end{pmatrix} \quad W = \begin{pmatrix} 0.2 & -0.2 \\ -0.7 & 0.1 \\ 0.8 & -0.5 \\ -0.4 & -0.2 \\ 0.2 & 0.4 \\ -0.1 & 0.1 \end{pmatrix}$$

$$p(h|v) = \begin{pmatrix} 0.73 & 0.33 \\ 0.57 & 0.50 \\ 0.62 & 0.31 \\ 0.27 & 0.45 \\ 0.38 & 0.62 \end{pmatrix}$$

~ 퍼센트로 1이 나온다



AI Lab, Hanyang University

Learning

- **예제**

2. $\boldsymbol{h} = p(h|v) > random\ value\ (0\sim1)$

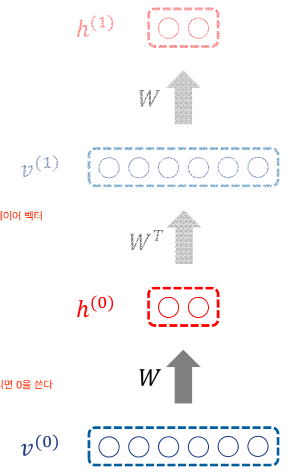
(각 행렬 값에 대해 서로 다른 random value와 비교)

$$v = \begin{pmatrix} 1\,0\,1\,0\,0\,0 \\ 0\,1\,1\,0\,1\,0 \\ 1\,0\,1\,1\,0\,1 \\ 1\,1\,0\,1\,0\,1 \\ 0\,1\,0\,0\,1\,0 \end{pmatrix} \quad W = \begin{pmatrix} 0.2 & -0.2 \\ -0.7 & 0.1 \\ 0.8 & -0.5 \\ -0.4 & -0.2 \\ 0.2 & 0.4 \\ -0.1 & 0.1 \end{pmatrix} \quad h = \begin{pmatrix} 1\,0 \\ 1\,1 \\ 1\,0 \\ 0\,0 \\ 0\,1 \end{pmatrix}$$

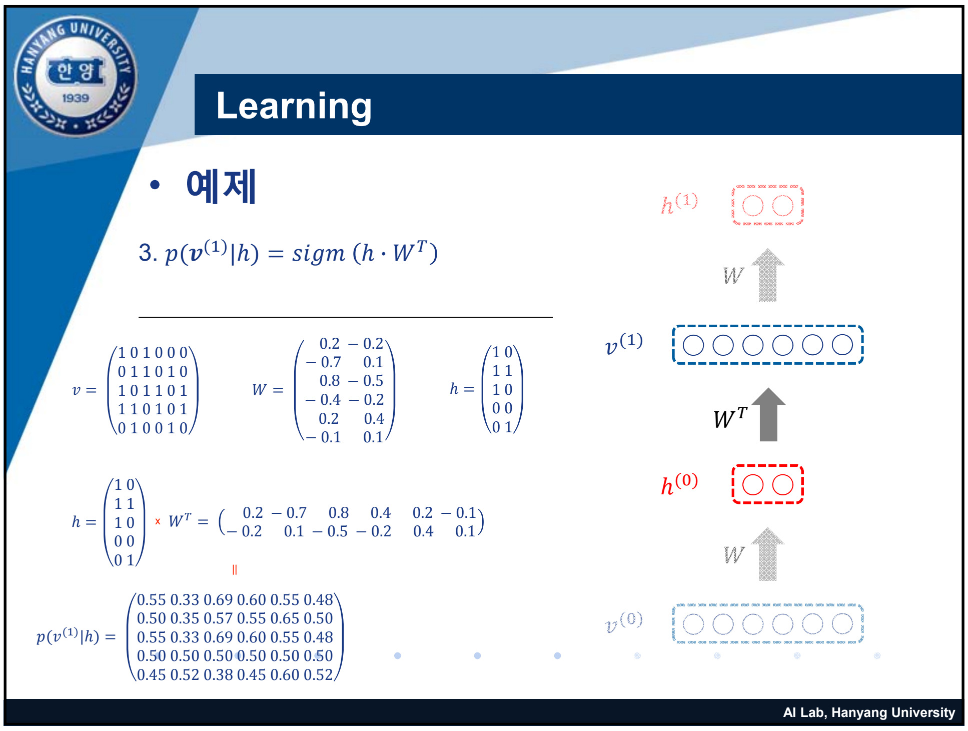
이게 히든 레이어 벡터

$$p(h|v) = \begin{pmatrix} 0.73 & 0.33 \\ 0.57 & 0.50 \\ 0.62 & 0.31 \\ 0.27 & 0.45 \\ 0.38 & 0.62 \end{pmatrix}$$

그리고 이 행렬에 따라 생성을 시킨다
1이 나올 확률이 랜덤 값보다 크면 1쓰고 아니면 0을 쓴다



AI Lab, Hanyang University



Learning

- **예제**

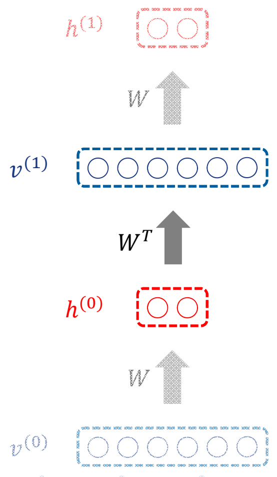
3. $p(\boldsymbol{v}^{(1)}|h) = sigm\ (h \cdot W^T)$

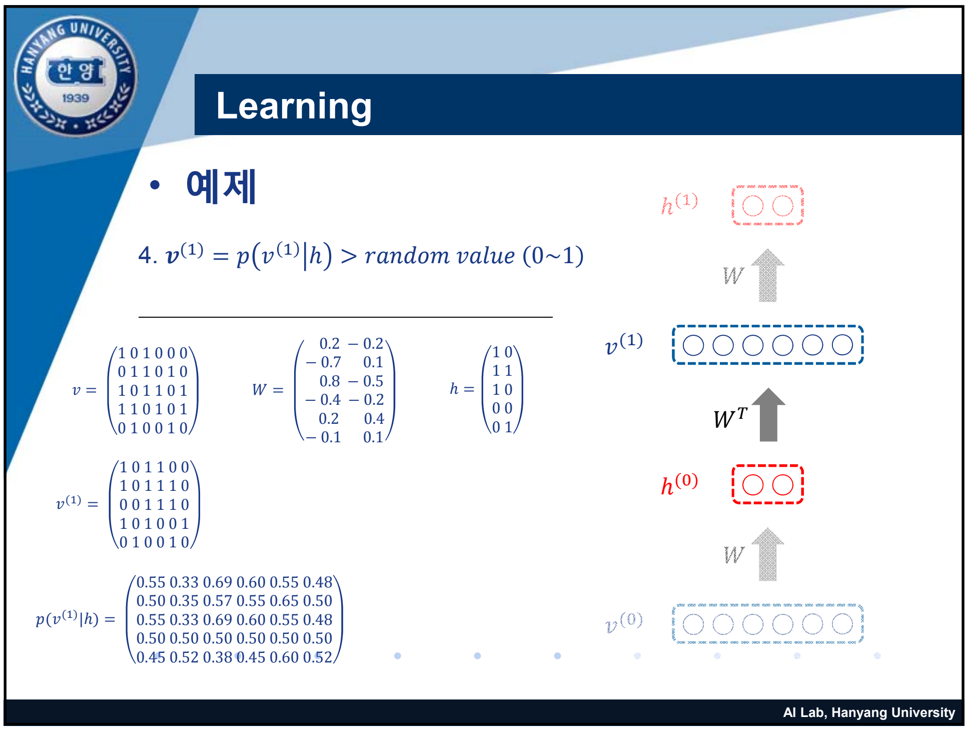
$$v = \begin{pmatrix} 1&0&1&0&0&0 \\ 0&1&1&0&1&0 \\ 1&0&1&1&0&1 \\ 1&1&0&1&0&1 \\ 0&1&0&0&1&0 \end{pmatrix} \qquad W = \begin{pmatrix} 0.2 & -0.2 \\ -0.7 & 0.1 \\ 0.8 & -0.5 \\ -0.4 & -0.2 \\ 0.2 & 0.4 \\ -0.1 & 0.1 \end{pmatrix} \qquad h = \begin{pmatrix} 1&0 \\ 1&1 \\ 1&0 \\ 0&0 \\ 0&1 \end{pmatrix}$$

$$h = \begin{pmatrix} 1&0 \\ 1&1 \\ 1&0 \\ 0&0 \\ 0&1 \end{pmatrix} \times W^T = \begin{pmatrix} 0.2 & -0.7 & 0.8 & 0.4 & 0.2 & -0.1 \\ -0.2 & 0.1 & -0.5 & -0.2 & 0.4 & 0.1 \end{pmatrix}$$

$$\parallel$$

$$p(v^{(1)}|h) = \begin{pmatrix} 0.55 & 0.33 & 0.69 & 0.60 & 0.55 & 0.48 \\ 0.50 & 0.35 & 0.57 & 0.55 & 0.65 & 0.50 \\ 0.55 & 0.33 & 0.69 & 0.60 & 0.55 & 0.48 \\ 0.50 & 0.50 & 0.50 & 0.50 & 0.50 & 0.50 \\ 0.45 & 0.52 & 0.38 & 0.45 & 0.60 & 0.52 \end{pmatrix}$$





Learning

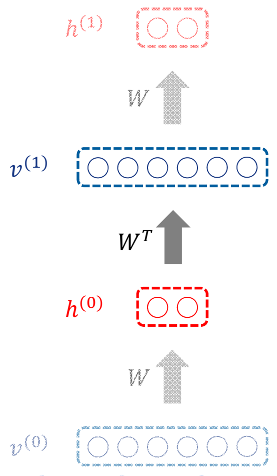
- **예제**

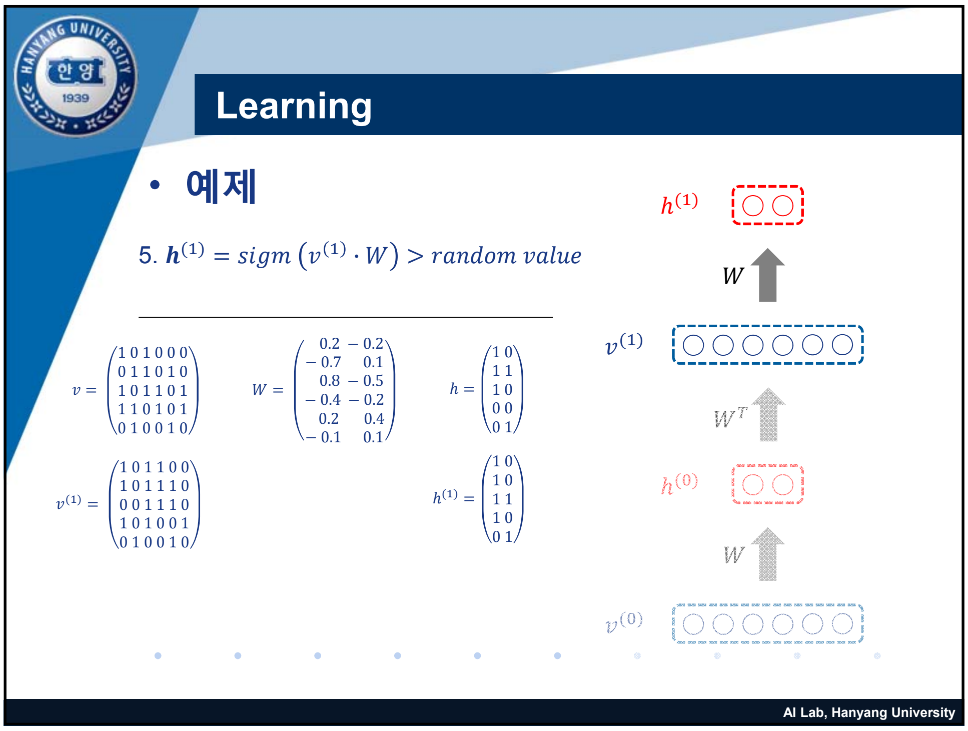
4. $\boldsymbol{v}^{(1)} = p\left(v^{(1)}\middle|h\right) > random\ value\ (0{\sim}1)$

$$v = \begin{pmatrix} 1&0&1&0&0&0 \\ 0&1&1&0&1&0 \\ 1&0&1&1&0&1 \\ 1&1&0&1&0&1 \\ 0&1&0&0&1&0 \end{pmatrix} \qquad W = \begin{pmatrix} 0.2 & -0.2 \\ -0.7 & 0.1 \\ 0.8 & -0.5 \\ -0.4 & -0.2 \\ 0.2 & 0.4 \\ -0.1 & 0.1 \end{pmatrix} \qquad h = \begin{pmatrix} 1&0 \\ 1&1 \\ 1&0 \\ 0&0 \\ 0&1 \end{pmatrix}$$

$$v^{(1)} = \begin{pmatrix} 1&0&1&1&0&0 \\ 1&0&1&1&1&0 \\ 0&0&1&1&1&0 \\ 1&0&1&0&0&1 \\ 0&1&0&0&1&0 \end{pmatrix}$$

$$p(v^{(1)}|h) = \begin{pmatrix} 0.55 & 0.33 & 0.69 & 0.60 & 0.55 & 0.48 \\ 0.50 & 0.35 & 0.57 & 0.55 & 0.65 & 0.50 \\ 0.55 & 0.33 & 0.69 & 0.60 & 0.55 & 0.48 \\ 0.50 & 0.50 & 0.50 & 0.50 & 0.50 & 0.50 \\ 0.45 & 0.52 & 0.38 & 0.45 & 0.60 & 0.52 \end{pmatrix}$$





Learning

- **예제**

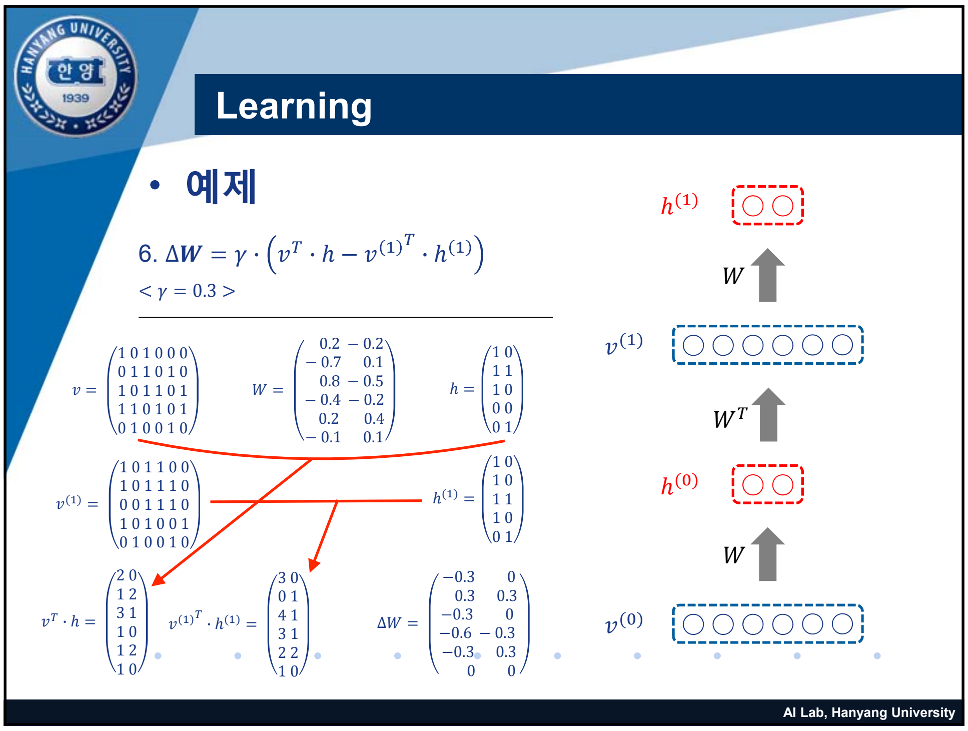
5. $\boldsymbol{h}^{(1)} = sigm\left(v^{(1)} \cdot W\right) > random\ value$

$$v = \begin{pmatrix} 1&0&1&0&0&0 \\ 0&1&1&0&1&0 \\ 1&0&1&1&0&1 \\ 1&1&0&1&0&1 \\ 0&1&0&0&1&0 \end{pmatrix} \quad W = \begin{pmatrix} 0.2 & -0.2 \\ -0.7 & 0.1 \\ 0.8 & -0.5 \\ -0.4 & -0.2 \\ 0.2 & 0.4 \\ -0.1 & 0.1 \end{pmatrix} \quad h = \begin{pmatrix} 1&0 \\ 1&1 \\ 1&0 \\ 0&0 \\ 0&1 \end{pmatrix}$$

$$v^{(1)} = \begin{pmatrix} 1&0&1&1&0&0 \\ 1&0&1&1&1&0 \\ 0&0&1&1&1&0 \\ 1&0&1&0&0&1 \\ 0&1&0&0&1&0 \end{pmatrix} \quad h^{(1)} = \begin{pmatrix} 1&0 \\ 1&0 \\ 1&1 \\ 1&0 \\ 0&1 \end{pmatrix}$$

$h^{(1)}$
W
$v^{(1)}$
W^T
$h^{(0)}$
W
$v^{(0)}$

AI Lab, Hanyang University



Learning

- **예제**

6. $\Delta \boldsymbol{W} = \gamma \cdot \left(v^T \cdot h - {v^{(1)}}^T \cdot h^{(1)}\right)$

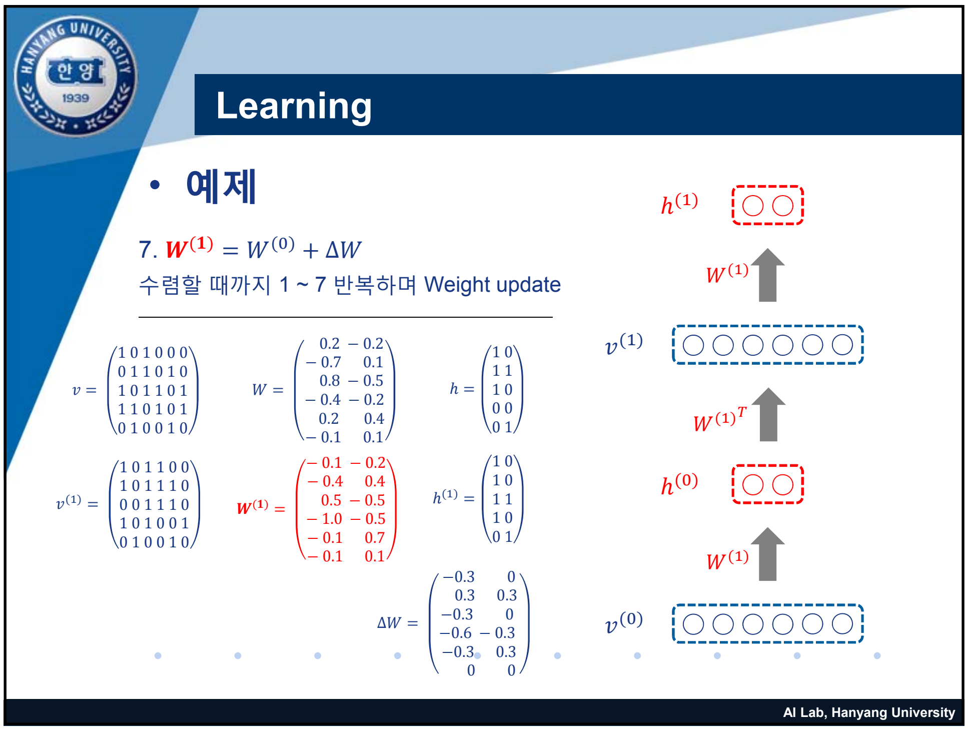
$< \gamma = 0.3 >$

$$v = \begin{pmatrix} 1&0&1&0&0&0 \\ 0&1&1&0&1&0 \\ 1&0&1&1&0&1 \\ 1&1&0&1&0&1 \\ 0&1&0&0&1&0 \end{pmatrix} \quad W = \begin{pmatrix} 0.2 & -0.2 \\ -0.7 & 0.1 \\ 0.8 & -0.5 \\ -0.4 & -0.2 \\ 0.2 & 0.4 \\ -0.1 & 0.1 \end{pmatrix} \quad h = \begin{pmatrix} 1&0 \\ 1&1 \\ 1&0 \\ 0&0 \\ 0&1 \end{pmatrix}$$

$$v^{(1)} = \begin{pmatrix} 1&0&1&1&0&0 \\ 1&0&1&1&1&0 \\ 0&0&1&1&1&0 \\ 1&0&1&0&0&1 \\ 0&1&0&0&1&0 \end{pmatrix} \quad h^{(1)} = \begin{pmatrix} 1&0 \\ 1&0 \\ 1&1 \\ 1&0 \\ 0&1 \end{pmatrix}$$

$$v^T \cdot h = \begin{pmatrix} 2&0 \\ 1&2 \\ 3&1 \\ 1&0 \\ 1&2 \\ 1&0 \end{pmatrix} \quad {v^{(1)}}^T \cdot h^{(1)} = \begin{pmatrix} 3&0 \\ 0&1 \\ 4&1 \\ 3&1 \\ 2&2 \\ 1&0 \end{pmatrix} \quad \Delta W = \begin{pmatrix} -0.3 & 0 \\ 0.3 & 0.3 \\ -0.3 & 0 \\ -0.6 & -0.3 \\ -0.3 & 0.3 \\ 0 & 0 \end{pmatrix}$$

AI Lab, Hanyang University



Learning

- **예제**

7. $\mathbf{W^{(1)}} = W^{(0)} + \Delta W$

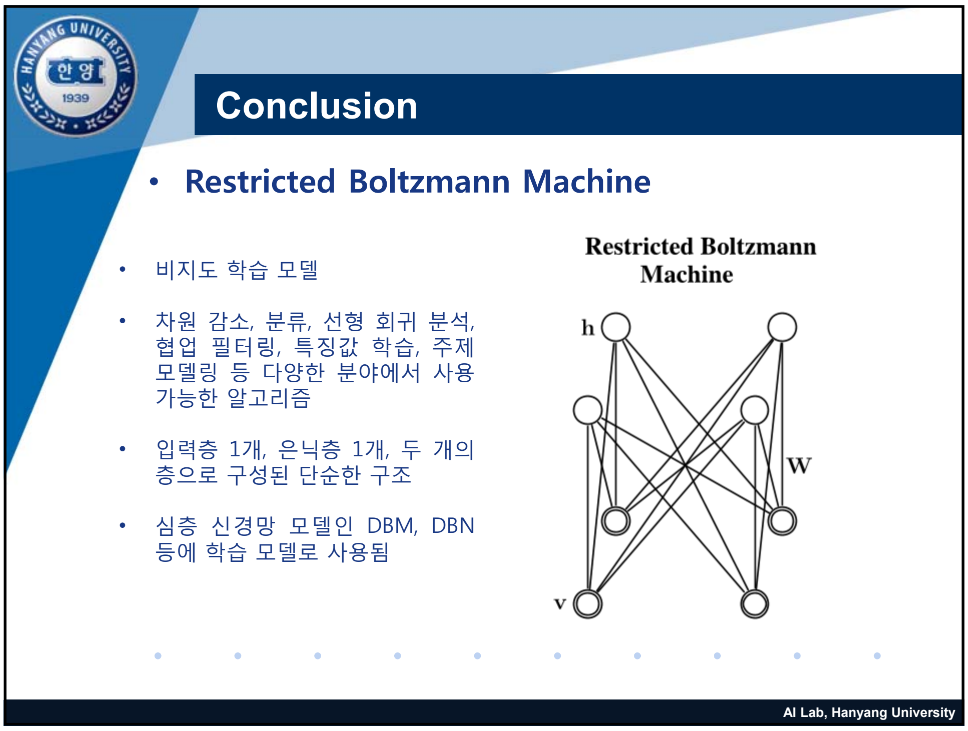
수렴할 때까지 1 ~ 7 반복하며 Weight update

$$v = \begin{pmatrix} 1\,0\,1\,0\,0\,0 \\ 0\,1\,1\,0\,1\,0 \\ 1\,0\,1\,1\,0\,1 \\ 1\,1\,0\,1\,0\,1 \\ 0\,1\,0\,0\,1\,0 \end{pmatrix} \quad W = \begin{pmatrix} 0.2 & -0.2 \\ -0.7 & 0.1 \\ 0.8 & -0.5 \\ -0.4 & -0.2 \\ 0.2 & 0.4 \\ -0.1 & 0.1 \end{pmatrix} \quad h = \begin{pmatrix} 1\,0 \\ 1\,1 \\ 1\,0 \\ 0\,0 \\ 0\,1 \end{pmatrix}$$

$$v^{(1)} = \begin{pmatrix} 1\,0\,1\,1\,0\,0 \\ 1\,0\,1\,1\,1\,0 \\ 0\,0\,1\,1\,1\,0 \\ 1\,0\,1\,0\,0\,1 \\ 0\,1\,0\,0\,1\,0 \end{pmatrix} \quad \mathbf{W^{(1)}} = \begin{pmatrix} -0.1 & -0.2 \\ -0.4 & 0.4 \\ 0.5 & -0.5 \\ -1.0 & -0.5 \\ -0.1 & 0.7 \\ -0.1 & 0.1 \end{pmatrix} \quad h^{(1)} = \begin{pmatrix} 1\,0 \\ 1\,0 \\ 1\,1 \\ 1\,0 \\ 0\,1 \end{pmatrix}$$

$$\Delta W = \begin{pmatrix} -0.3 & 0 \\ 0.3 & 0.3 \\ -0.3 & 0 \\ -0.6 & -0.3 \\ -0.3 & 0.3 \\ 0 & 0 \end{pmatrix}$$

AI Lab, Hanyang University



Conclusion

- **Restricted Boltzmann Machine**

- 비지도 학습 모델
- 차원 감소, 분류, 선형 회귀 분석, 협업 필터링, 특징값 학습, 주제 모델링 등 다양한 분야에서 사용 가능한 알고리즘
- 입력층 1개, 은닉층 1개, 두 개의 층으로 구성된 단순한 구조
- 심층 신경망 모델인 DBM, DBN 등에 학습 모델로 사용됨

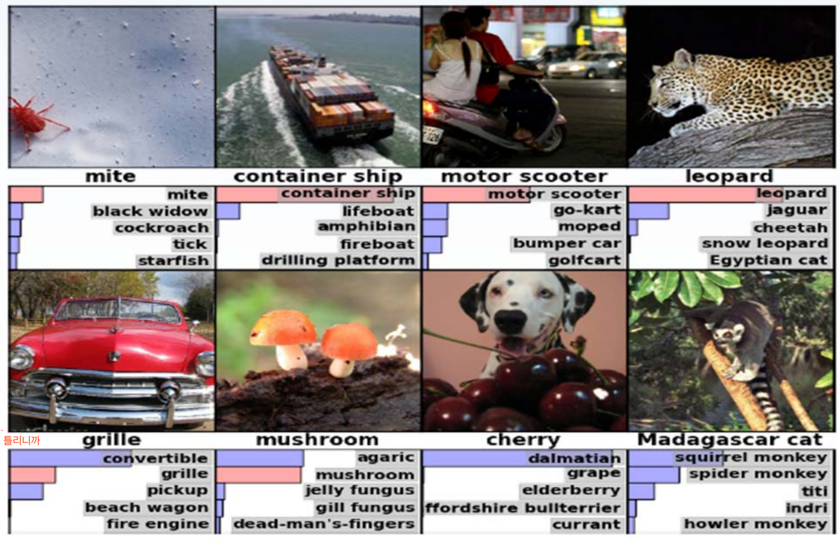
Restricted Boltzmann Machine
h
W
v

AI Lab, Hanyang University

Convolutional Neural Network (CNN)

AI Lab, Hanyang University

CNN(Convolutional Neural Net) 영상 분류 (1/3)

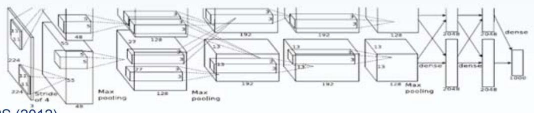


틀렸지만..
뭐 사람도 틀리니까

Krizhevsky et al., NIPS (2012)

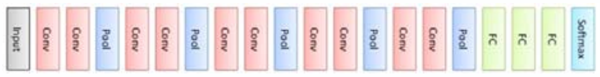
CNN 영상 분류 (2/3)

- 딥러닝 기반의 이미지 분류 기법: AlexNet(Univ. of Toronto) - ILSVRC'12
  - 크고 깊은 컨볼루션 신경망(CNN)을 학습하여 영상을 분류
  - 1000개의 클래스로 이루어진 1,200,000개의 영상을 분류하는데 있어서 16.4%의 에러율



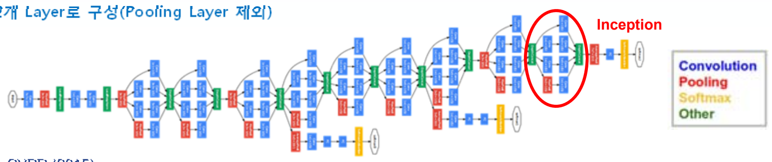
Krizhevsky et al., NIPS (2012)

- 최적의 네트워크 디자인: VGG(Univ. of Oxford) - ILSVRC'14
  - 네트워크의 깊이가 어떠한 영향을 주는지 연구 하기 위하여 더 깊은 네트워크를 구성
  - Convolutional Layer와 Pooling Layer를 중첩하여 총 19개의 Layer를 쌓음



Simonyan and Zisserman, arXiv (2014)

- 더 깊은 컨볼루션 네트워크: GoogLeNet(Google) - ILSVRC'14
  - Inception 모듈을 반복해서 쌓는 형태로 네트워크가 발전
  - 총 22개 Layer로 구성(Pooling Layer 제외)

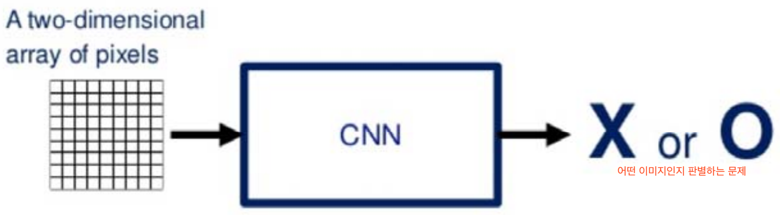


Szegedy et al., CVPR (2015)

27

How ConvNet Works

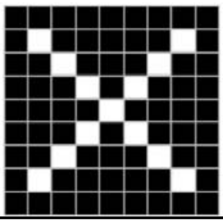
- For example, a ConvNet takes the input as an image which can be classified as 'X' or 'O'



어떤 이미지인지 판별하는 문제

- In a simple case, 'X' would look like:

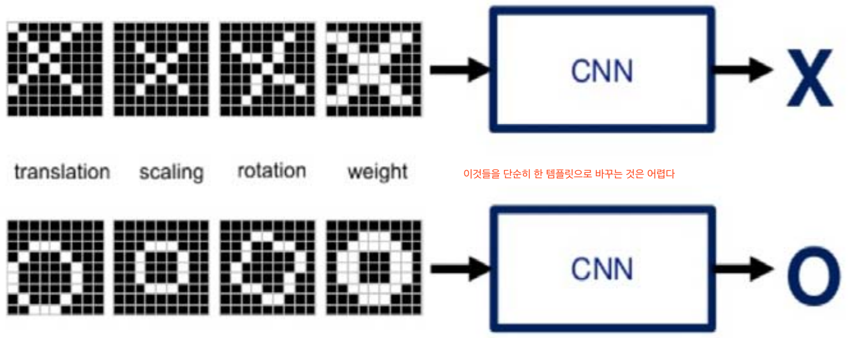
타겟과 템플릿의 내적 값을 구하면..
부합도가 높다



새로운 이미지와 이 이미지의 내적을 계산하고
내적이 높을 수록 새로운 이미지와 이 이미지는 부합하는 것이다

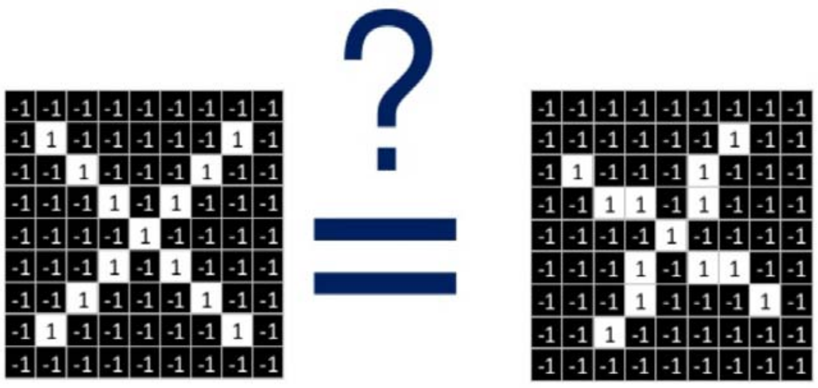
How ConvNet Works

- What about trickier cases?



이것들을 단순히 한 템플릿으로 바꾸는 것은 어렵다

How ConvNet Works – What Computer Sees



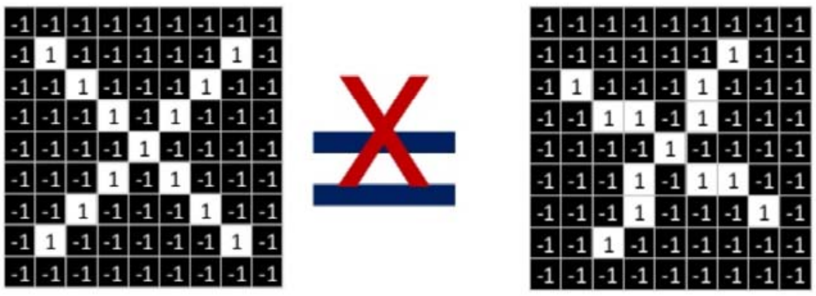
-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	1	-1	-1	-1	-1	-1	1	-1
-1	-1	1	-1	-1	-1	1	-1	-1
-1	-1	-1	1	-1	1	-1	-1	-1
-1	-1	-1	-1	1	-1	-1	-1	-1
-1	-1	-1	1	-1	1	-1	-1	-1
-1	-1	1	-1	-1	-1	1	-1	-1
-1	1	-1	-1	-1	-1	-1	1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1

?

=

-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	1	-1	-1
-1	1	-1	-1	-1	1	-1	-1	-1
-1	-1	1	1	-1	1	-1	-1	-1
-1	-1	-1	-1	1	-1	-1	-1	-1
-1	-1	-1	1	-1	1	1	-1	-1
-1	-1	-1	1	-1	-1	-1	1	-1
-1	-1	1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1

How ConvNet Works



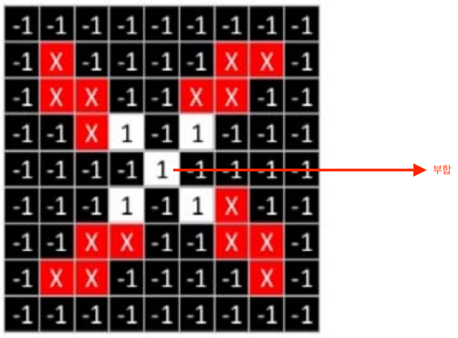
-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	1	-1	-1	-1	-1	-1	1	-1
-1	-1	1	-1	-1	-1	1	-1	-1
-1	-1	-1	1	-1	1	-1	-1	-1
-1	-1	-1	-1	1	-1	-1	-1	-1
-1	-1	-1	1	-1	1	-1	-1	-1
-1	-1	1	-1	-1	-1	1	-1	-1
-1	1	-1	-1	-1	-1	-1	1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1

$\neq$

-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	1	-1	-1
-1	1	-1	-1	-1	1	-1	-1	-1
-1	-1	1	1	-1	1	-1	-1	-1
-1	-1	-1	-1	1	-1	-1	-1	-1
-1	-1	-1	1	-1	1	1	-1	-1
-1	-1	-1	1	-1	-1	-1	1	-1
-1	-1	1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1

How ConvNet Works – What Computer Sees

- Since the pattern doesnot match exactly, the computer will not be able to classify this as 'X'



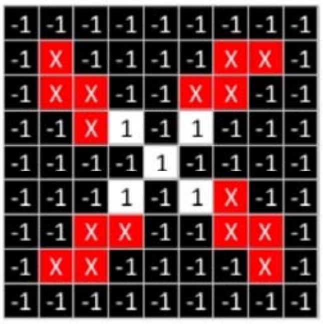
-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	X	-1	-1	-1	-1	X	X	-1
-1	X	X	-1	-1	X	X	-1	-1
-1	-1	X	1	-1	1	-1	-1	-1
-1	-1	-1	-1	1	-1	-1	-1	-1
-1	-1	-1	1	-1	1	X	-1	-1
-1	-1	X	X	-1	-1	X	X	-1
-1	X	X	-1	-1	-1	-1	X	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1

ConvNet Layers (At a Glance)

- CONV layer will compute the output of neurons that are connected to local regions in the input, each computing a dot product between their weights and a small region they are connected to in the input volume.
- RELU layer will apply an elementwise activation function, such as the max(0,x) thresholding at zero. This leaves the size of the volume unchanged.
- POOL layer will perform a downsampling operation along the spatial dimensions (width, height).
- FC (i.e. fully-connected) layer will compute the class scores, resulting in volume of size [1x1xN], where each of the N numbers correspond to a class score, such as among the N categories.

Recall – What Computer Sees

- Since the pattern doesnot match exactly, the computer will not be able to classify this as 'X'

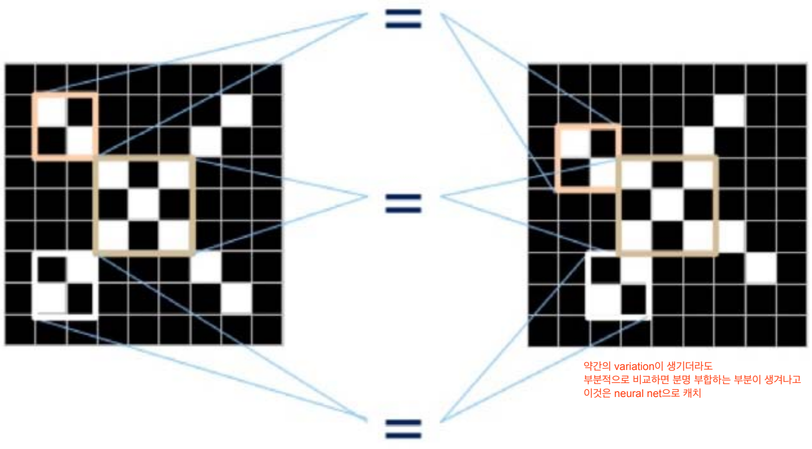


-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	X	-1	-1	-1	-1	X	X	-1
-1	X	X	-1	-1	X	X	-1	-1
-1	-1	X	1	-1	1	-1	-1	-1
-1	-1	-1	-1	1	-1	-1	-1	-1
-1	-1	-1	1	-1	1	X	-1	-1
-1	-1	X	X	-1	-1	X	X	-1
-1	X	X	-1	-1	-1	-1	X	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1

- What got changed?

Convolutional Layer

- Convolution layer will work to identify patterns (features) instead of individual pixels



약간의 variation이 생기더라도
부분적으로 비교하면 분명 부합하는 부분이 생겨나고
이것은 neural net으로 캐치

Convolutional Layer - Filters

- The CONV layer's parameters consist of a set of learnable filters.
- Every filter is small spatially (along width and height), but extends through the full depth of the input volume.
- During the forward pass, we slide (more precisely, convolve) each filter across the width and height of the input volume and compute dot products between the entries of the filter and the input at any position.

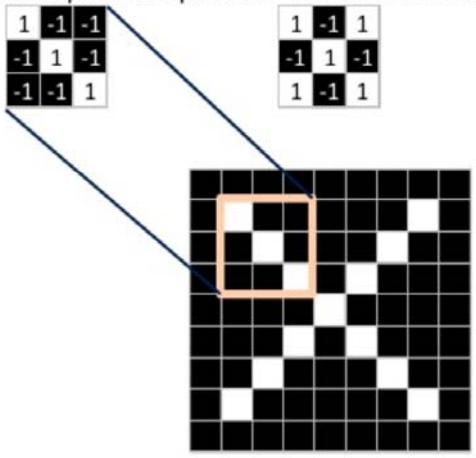
1	-1	-1
-1	1	-1
-1	-1	1

1	-1	1
-1	1	-1
1	-1	1

-1	-1	1
-1	1	-1
1	-1	-1

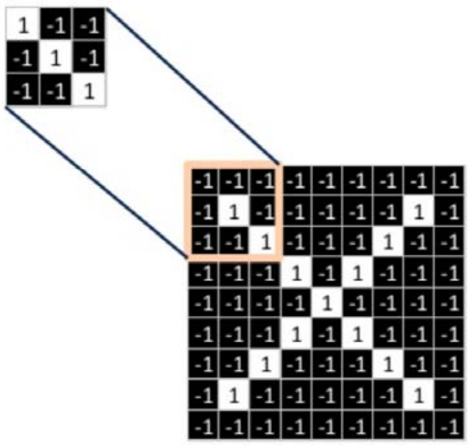
Convolutional Layer - Filters

- Sliding the filter over the width and height of the input gives 2-dimensional activation map that responds to that filter at every spatial position.



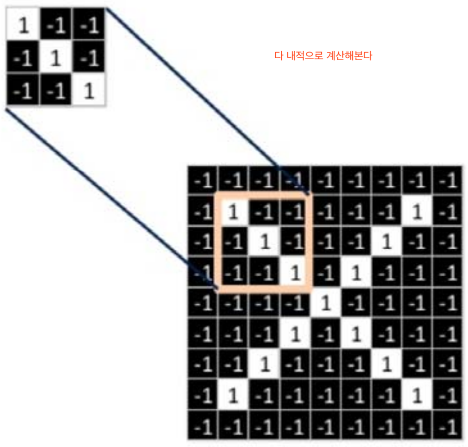
Convolutional Layer – Filters – Navigation Example

- Strides = 1, Filter Size = 3 X 3 X 1, Padding = 0

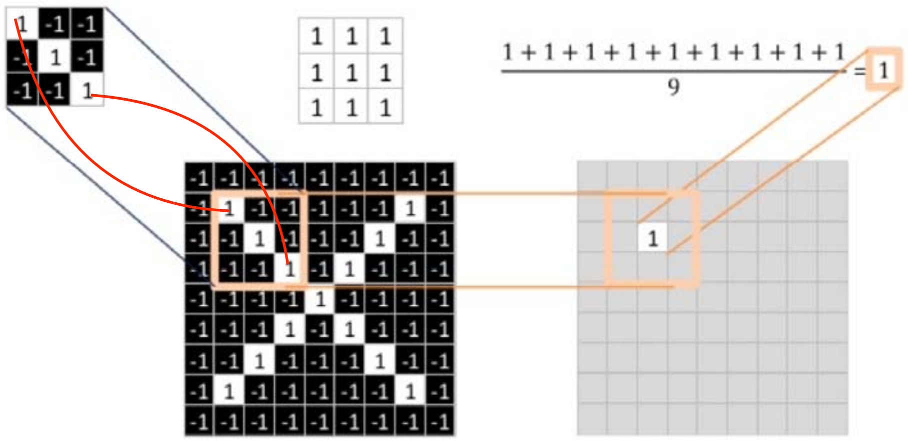


Convolutional Layer – Filters – Navigation Example

다 내적으로 계산해본다

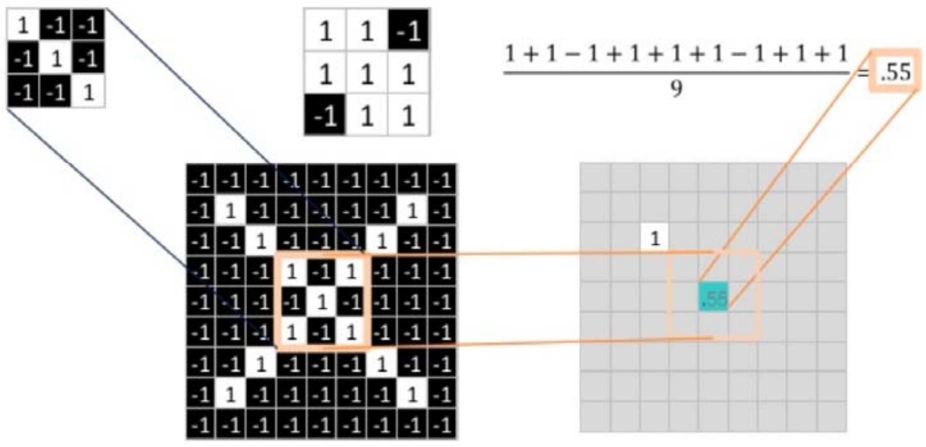


Convolutional Layer – Filters – Computation Example



$$\frac{1+1+1+1+1+1+1+1+1}{9} = 1$$

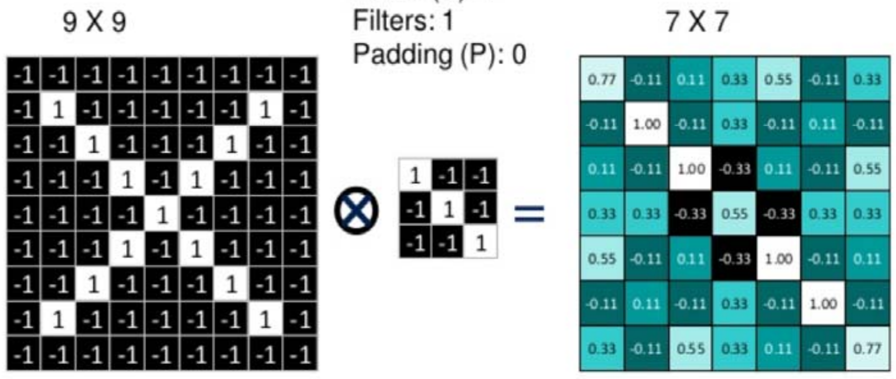
Convolutional Layer – Filters – Computation Example



$$\frac{1+1-1+1+1+1-1+1+1}{9} = .55$$

Convolutional Layer – Filters – Computation Example

Input Size (W): 9
Filter Size (F): 3 X 3
Stride (S): 1
Filters: 1
Padding (P): 0

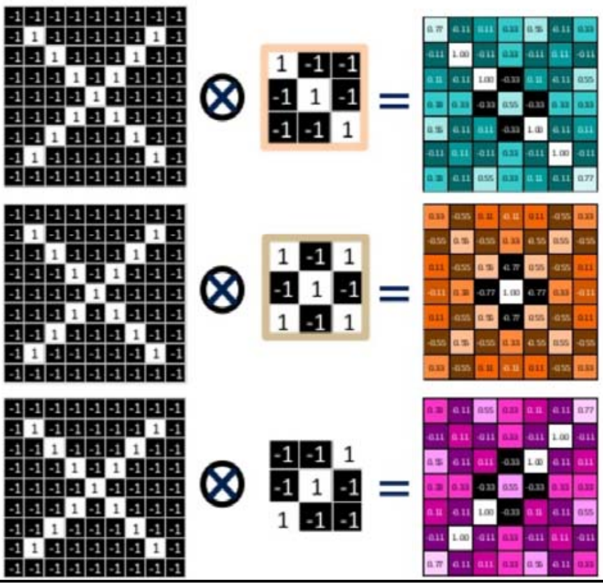


0.77	-0.11	0.11	0.33	0.55	-0.11	0.33
-0.11	1.00	-0.11	0.33	-0.11	0.11	-0.11
0.11	-0.11	1.00	-0.33	0.11	-0.11	0.55
0.33	0.33	-0.33	0.55	-0.33	0.33	0.33
0.55	-0.11	0.11	-0.33	1.00	-0.11	0.11
-0.11	0.11	-0.11	0.33	-0.11	1.00	-0.11
0.33	-0.11	0.55	0.33	0.11	-0.11	0.77

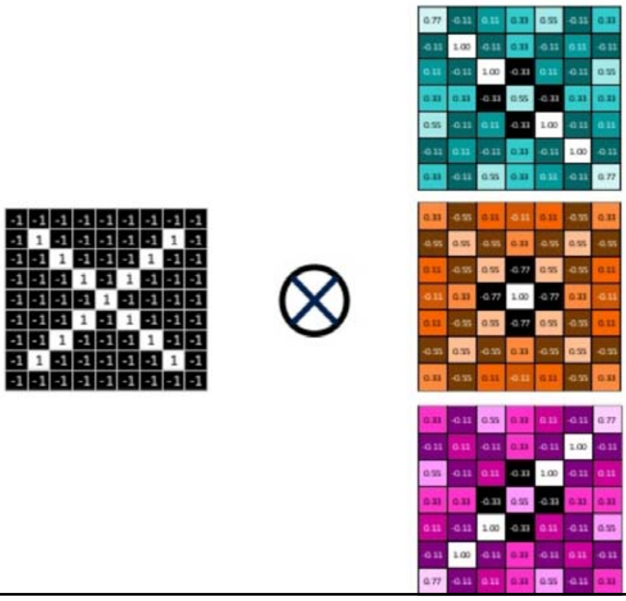
Feature Map Size = 1+ (W – F + 2P)/S
= 1+ (9 – 3 + 2 X 0)/1 = 7

Convolutional Layer – Filters – Output Feature Map

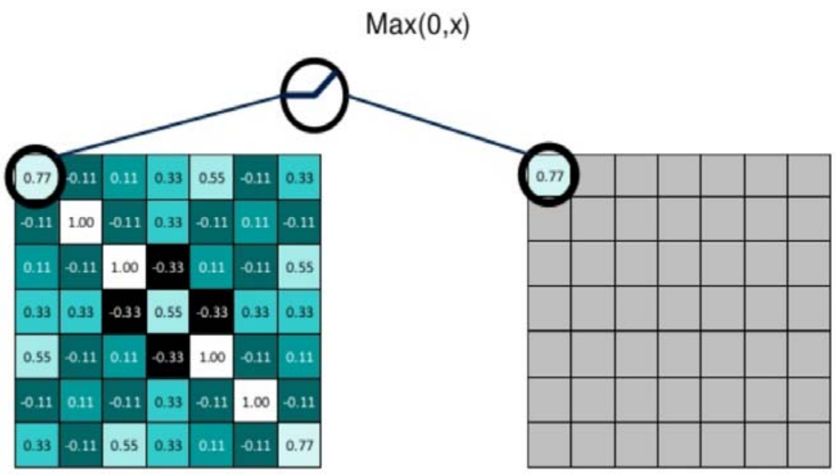
- Output Feature Map of One complete convolution:
  - Filters: 3
  - Filter Size: 3 X 3
  - Stride: 1
- Conclusion:
  - Input Image:
    9 X 9
  - Output of Convolution:
    7 X 7 X 3



Convolutional Layer – Output



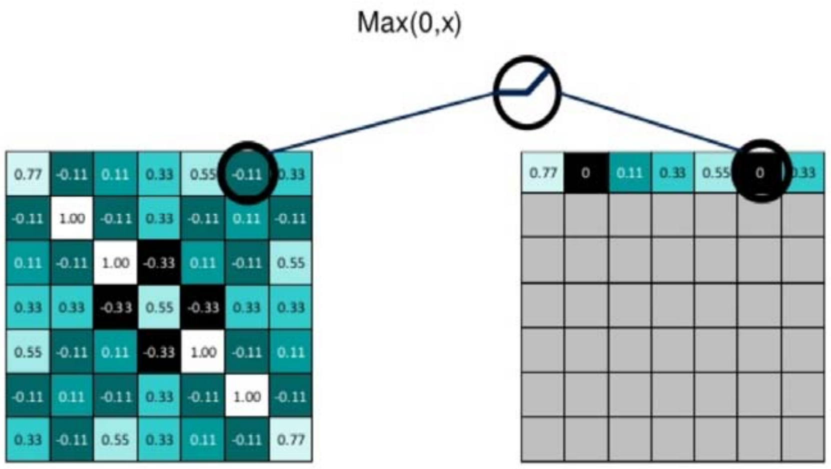
Rectified Linear Units (ReLUs)



0.77	-0.11	0.11	0.33	0.55	-0.11	0.33
-0.11	1.00	-0.11	0.33	-0.11	0.11	-0.11
0.11	-0.11	1.00	-0.33	0.11	-0.11	0.55
0.33	0.33	-0.33	0.55	-0.33	0.33	0.33
0.55	-0.11	0.11	-0.33	1.00	-0.11	0.11
-0.11	0.11	-0.11	0.33	-0.11	1.00	-0.11
0.33	-0.11	0.55	0.33	0.11	-0.11	0.77

0.77						

Rectified Linear Units (ReLUs)



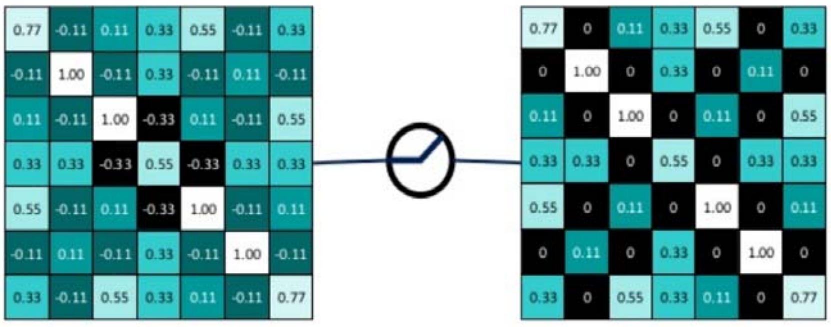
0.77	-0.11	0.11	0.33	0.55	-0.11	0.33
-0.11	1.00	-0.11	0.33	-0.11	0.11	-0.11
0.11	-0.11	1.00	-0.33	0.11	-0.11	0.55
0.33	0.33	-0.33	0.55	-0.33	0.33	0.33
0.55	-0.11	0.11	-0.33	1.00	-0.11	0.11
-0.11	0.11	-0.11	0.33	-0.11	1.00	-0.11
0.33	-0.11	0.55	0.33	0.11	-0.11	0.77

0.77	0	0.11	0.33	0.55	0	0.33

Rectified Linear Units (ReLUs)

Max(0,x)

좀 더 명확
ReLUs를 통해 쓸데없는 정보를 없애버림 (마이너스 없애기)

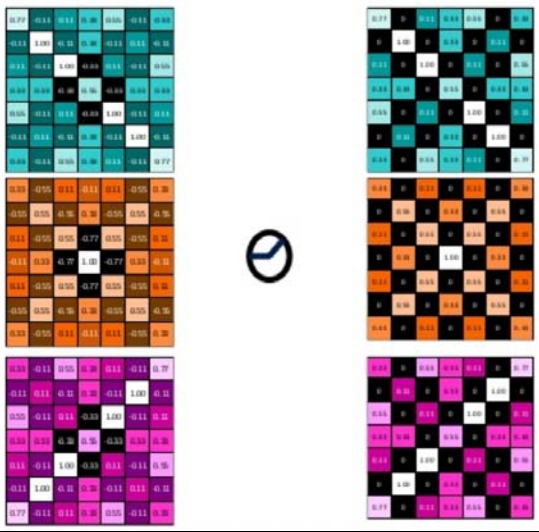


0.77	-0.11	0.11	0.33	0.55	-0.11	0.33
-0.11	1.00	-0.11	0.33	-0.11	0.11	-0.11
0.11	-0.11	1.00	-0.33	0.11	-0.11	0.55
0.33	0.33	-0.33	0.55	-0.33	0.33	0.33
0.55	-0.11	0.11	-0.33	1.00	-0.11	0.11
-0.11	0.11	-0.11	0.33	-0.11	1.00	-0.11
0.33	-0.11	0.55	0.33	0.11	-0.11	0.77

0.77	0	0.11	0.33	0.55	0	0.33
0	1.00	0	0.33	0	0.11	0
0.11	0	1.00	0	0.11	0	0.55
0.33	0.33	0	0.55	0	0.33	0.33
0.55	0	0.11	0	1.00	0	0.11
0	0.11	0	0.33	0	1.00	0
0.33	0	0.55	0.33	0.11	0	0.77

ReLU layer

A stack of images becomes a stack of images with no negative values.

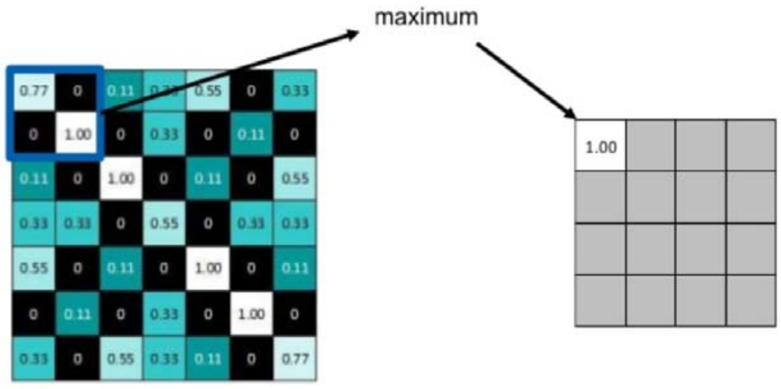


Pooling Layer

- The pooling layers down-sample the previous layers feature map.
- Its function is to progressively reduce the spatial size of the representation to reduce the amount of parameters and computation in the network
- The pooling layer often uses the Max operation to perform the downsampling process

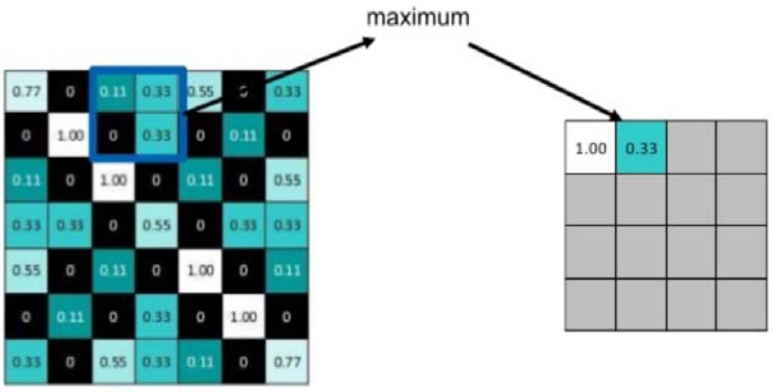
Pooling

- Pooling Filter Size = 2 X 2, Stride = 2



Pooling

- Pooling Filter Size = 2 X 2, Stride = 2

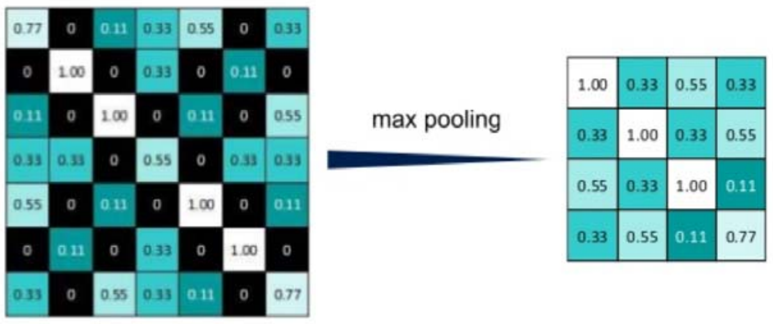


0.77	0	0.11	0.33	0.55	0	0.33
0	1.00	0	0.33	0	0.11	0
0.11	0	1.00	0	0.11	0	0.55
0.33	0.33	0	0.55	0	0.33	0.33
0.55	0	0.11	0	1.00	0	0.11
0	0.11	0	0.33	0	1.00	0
0.33	0	0.55	0.33	0.11	0	0.77

1.00	0.33		

Pooling

- Pooling Filter Size = 2 X 2, Stride = 2

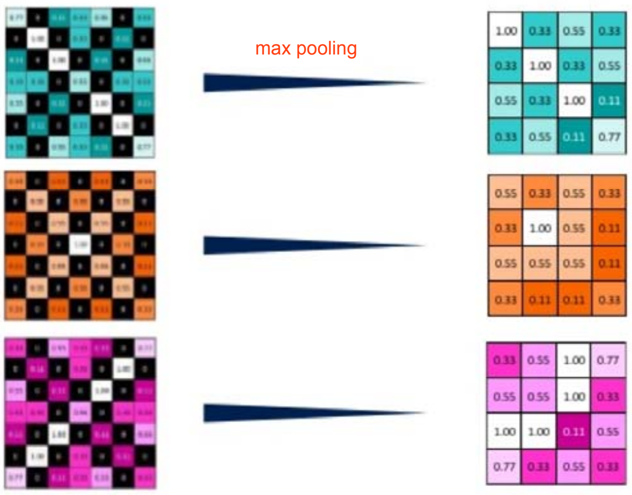


0.77	0	0.11	0.33	0.55	0	0.33
0	1.00	0	0.33	0	0.11	0
0.11	0	1.00	0	0.11	0	0.55
0.33	0.33	0	0.55	0	0.33	0.33
0.55	0	0.11	0	1.00	0	0.11
0	0.11	0	0.33	0	1.00	0
0.33	0	0.55	0.33	0.11	0	0.77

1.00	0.33	0.55	0.33
0.33	1.00	0.33	0.55
0.55	0.33	1.00	0.11
0.33	0.55	0.11	0.77

Pooling

A stack of 3 larger images becomes a stack of smaller images.



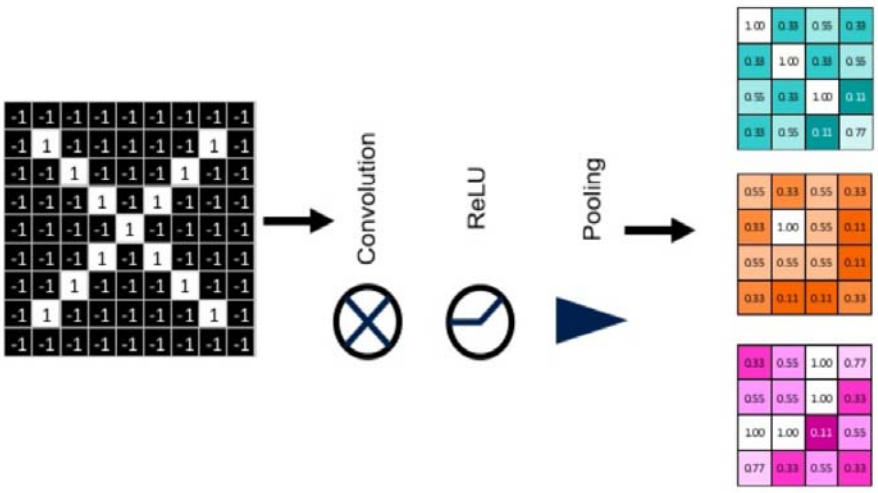
1.00	0.33	0.55	0.33
0.33	1.00	0.33	0.55
0.55	0.33	1.00	0.11
0.33	0.55	0.11	0.77

0.55	0.33	0.55	0.33
0.33	1.00	0.55	0.11
0.55	0.55	0.55	0.11
0.33	0.11	0.11	0.33

0.33	0.55	1.00	0.77
0.55	0.55	1.00	0.33
1.00	1.00	0.11	0.55
0.77	0.33	0.55	0.33

Layers get stacked

The output of one becomes the input of the next.



-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	1	-1	-1	-1	-1	-1	1	-1
-1	-1	1	-1	-1	-1	1	-1	-1
-1	-1	-1	1	-1	1	-1	-1	-1
-1	-1	-1	-1	1	-1	-1	-1	-1
-1	-1	-1	1	-1	1	-1	-1	-1
-1	-1	1	-1	-1	-1	1	-1	-1
-1	1	-1	-1	-1	-1	-1	1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1

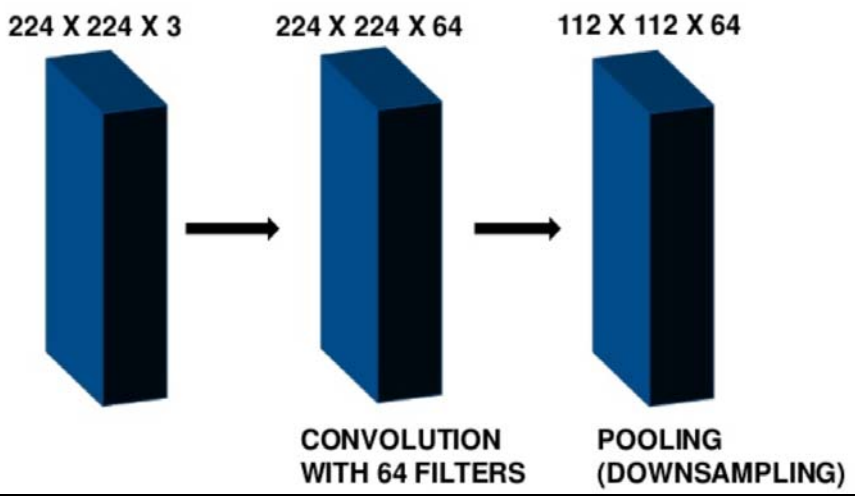
1.00	0.33	0.55	0.33
0.33	1.00	0.33	0.55
0.55	0.33	1.00	0.11
0.33	0.55	0.11	0.77

0.55	0.33	0.55	0.33
0.33	1.00	0.55	0.11
0.55	0.55	0.55	0.11
0.33	0.11	0.11	0.33

0.33	0.55	1.00	0.77
0.55	0.55	1.00	0.33
1.00	1.00	0.11	0.55
0.77	0.33	0.55	0.33

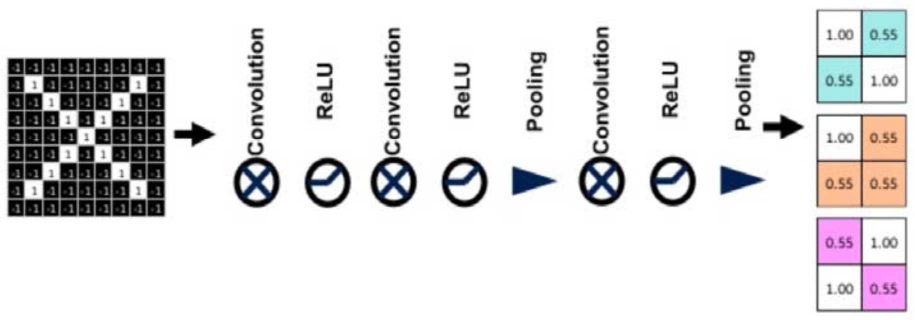
Layers Get Stacked - Example

이걸 계속 반복해주는 것이
convolution pooling
(deep learning architecture)



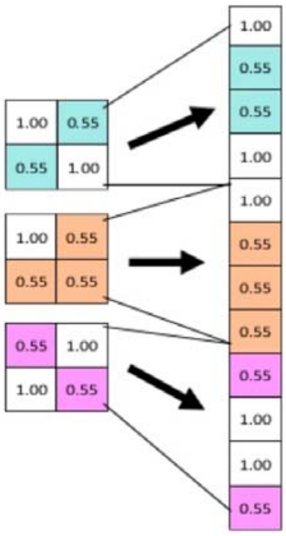
Deep stacking

Layers can be repeated several (or many) times.



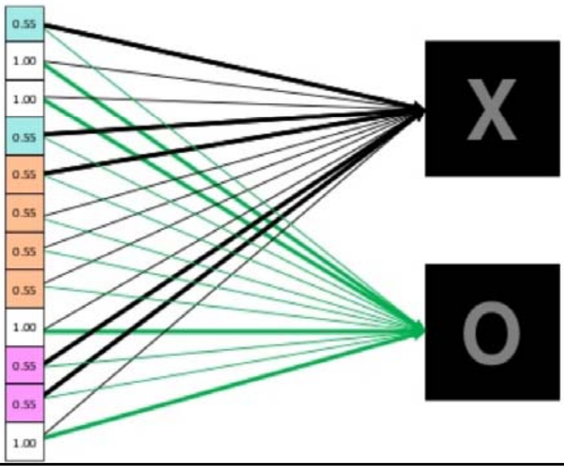
Fully connected layer

- Fully connected layers are the normal flat feed-forward neural network layers.
- These layers may have a non-linear activation function or a softmax activation in order to predict classes.
- To compute our output, we simply re-arrange the output matrices as a 1-D array.

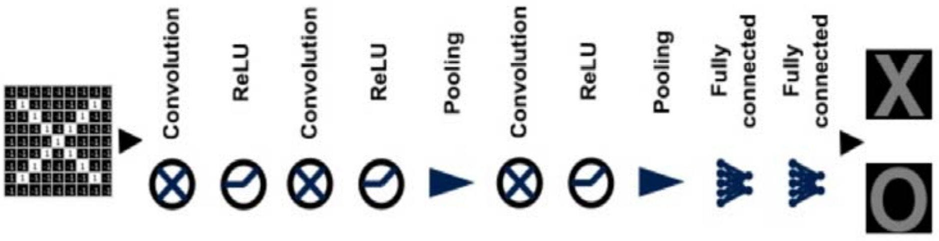


Fully connected layer

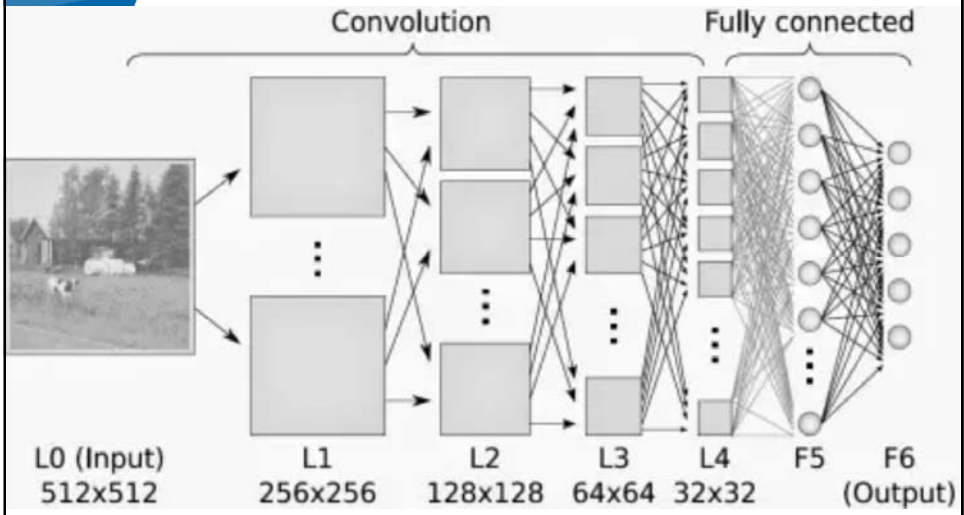
- A summation of product of inputs and weights at each output node determines the final prediction

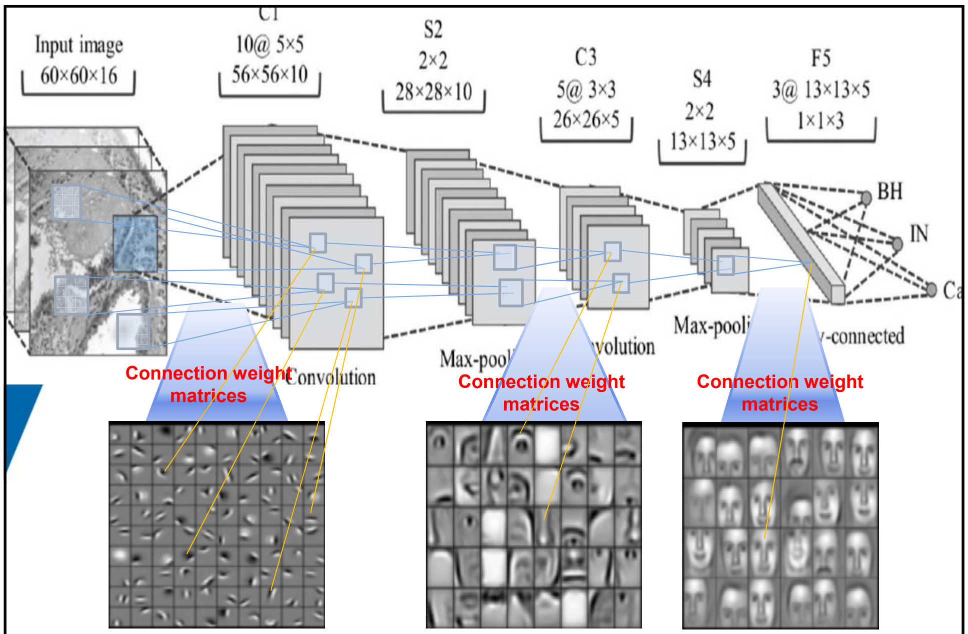


Putting it all together



Overall CNN Architecture – image Recognition



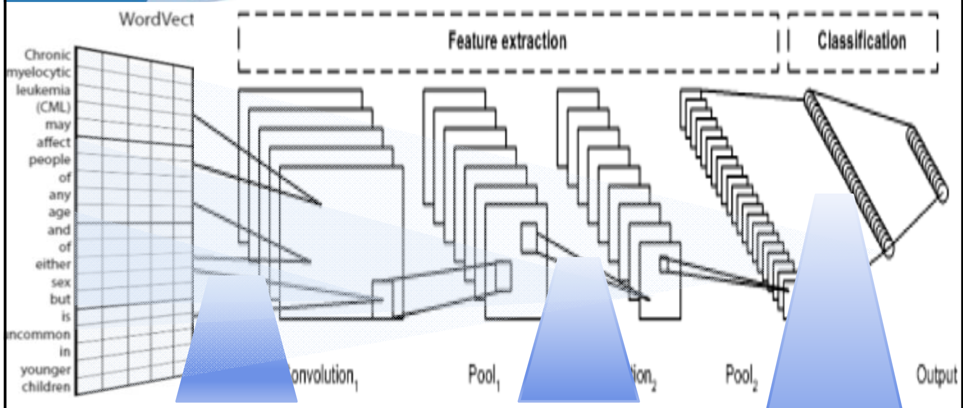


Low level Convolution filter

Middle level Convolution filter

High level Convolution filter

Overall CNN Architecture – Text Recognition



Phrase Filter
e.g. "people of any age"

Clause Filter
e.g. "affect people of any age"

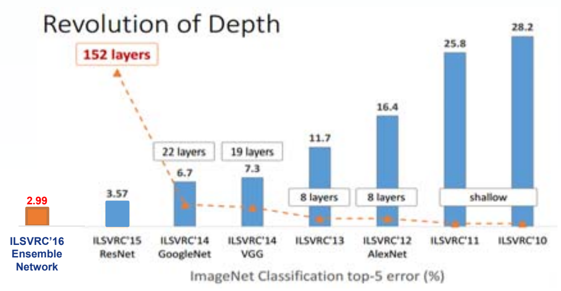
Sentence Filter
e.g. "CML may affect people of any

CNN 영상 분류 (3/3)

- 깊은 나머지 네트워크: ResNet(Microsoft) - ILSVRC'15
  - 152층의 layer를 쌓아서 매우 깊은 구조의 네트워크를 학습 가능하도록 구성
    - 19개 층이 넘어 갈 경우 Gradient값이 포화되어 학습의 효과가 저하되는 현상 해결

He et al., CVPR (2016)

- 다양한 네트워크의 조합: Ensemble Network(Trimps-Soushen) - ILSVRC'16
  - ILSVRC'16 에서는 Top-5 오류율이 2.99%까지 감소
    - The Third Research Institute of the Ministry of Public Security, China(공안부 산하 연구소)
  - ResNet(Microsoft), GoogleNet(Google) 등을 조합하여 사용



영상의 모호함으로 인한 인간의 오류율은 5.1%

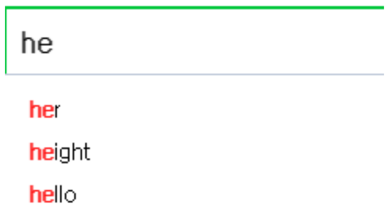
http://image-net.org/challenges/LSVRC/2016/results

Recurrent Neural Network (RNN)

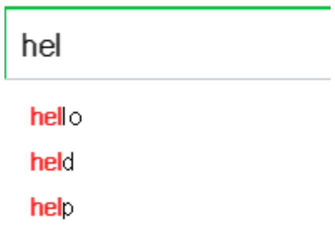
Introduction

- **기존의 FFNN으로는 불가능한 것들**

- 입력에 따른 빈출단어 예측 (검색어 자동완성)
- 단어들의 길이가 다르기 때문에 학습이 힘듦


he

her
height
hello


hel

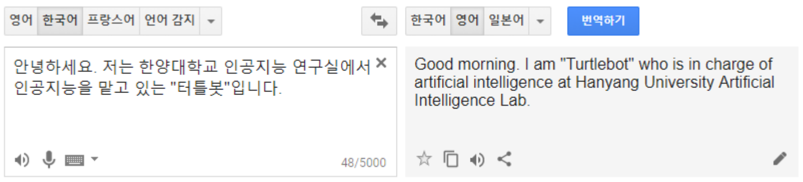
hello
held
help

AI Lab, Hanyang University

Introduction

- **기존의 FFNN으로는 불가능한 것들**

- 기계 번역
- 어순, 문맥 등을 파악하기 쉽지 않다.



안녕하세요. 저는 한양대학교 인공지능 연구실에서 인공지능을 맡고 있는 "터틀봇"입니다.

Good morning. I am "Turtlebot" who is in charge of artificial intelligence at Hanyang University Artificial Intelligence Lab.

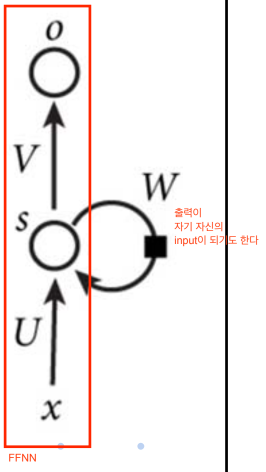
- 기존의 인공신경망은 순차적인 정보가 담긴 데이터를 처리하는 것이 어려움

AI Lab, Hanyang University

Concept

- **Recursive Neural Network**

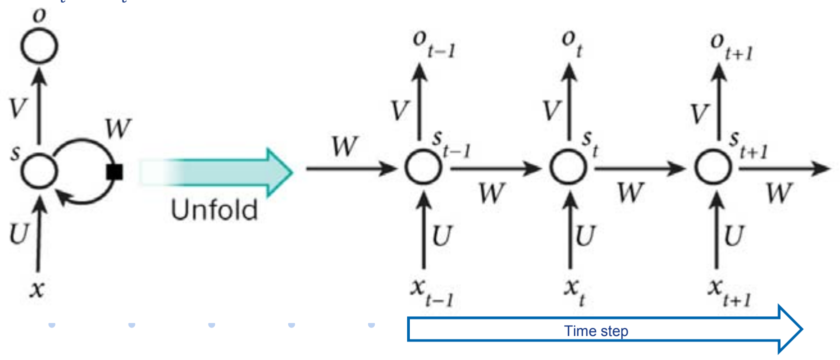
- 순차적인 정보가 담긴 데이터(Sequence Data)를 처리하는데 용이
- hidden layers에서의 결과가 다음 순서의 hidden layers의 입력으로 들어가도록 설계
- 사람이 대화를 할 때 이전 대화 내용을 기억해 현재 대화 내용을 이해하는 것처럼, 순차적인 데이터 입력에 따라 문맥 정보 파악
- 이론적으로 긴 Sequence Data를 처리할 수 있지만, 실제로는 비교적 짧은 Sequence Data만 효과적으로 처리 – 장기 의존성 문제



Concept

- **Recursive Neural Network**

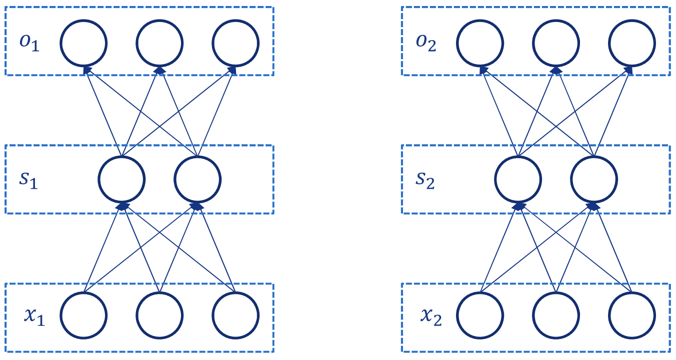
- x_t, s_t, o_t는 각각 *step* t에서의 입력층 값, 은닉층 값, 출력층 값
- $s_t = f(Ux_t + Ws_{t-1})$ → 이전 상태 은닉층의 영향을 받음을 의미
- activation 함수 f는 보통 tanh나 ReLU가 사용된다.
- $o_t = Vs_t$



Concept

- **Recursive Neural Network**

- $step$이 진행됨에 따라 input state, hidden state, output state의 값은 변한다.



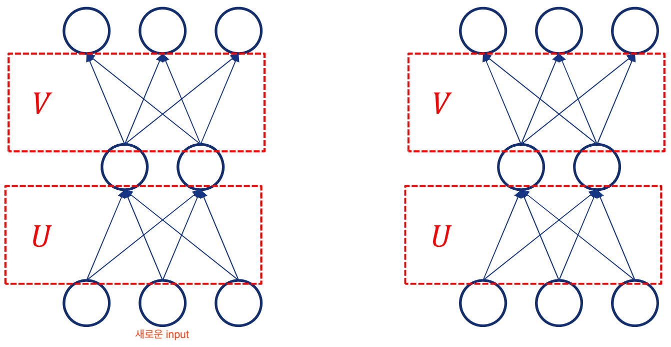
Step 1 Step 2

AI Lab, Hanyang University

Concept

- **Recursive Neural Network**

- 반면 U, V, W는 step이 진행되어도 변하지 않는다.
- 전체 Sequence를 하나의 학습 데이터로 보기 때문



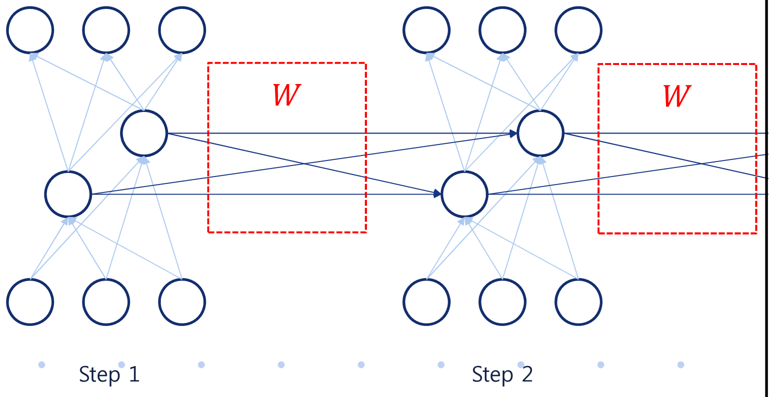
Step 1 Step 2

AI Lab, Hanyang University

Concept

- **Recursive Neural Network**

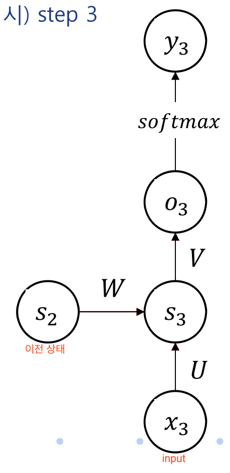
- 반면 U, V, W는 step이 진행되어도 변하지 않는다.
- 전체 Sequence를 하나의 학습 데이터로 보기 때문



Learning

- **Forward**

- 예시) step 3



$$s_3 = tanh(\ Ux_3 + Ws_2\ )$$

$$o_3 = Vs_3$$

구글링하면 자세히 나온다 (별건 아니라고 하시는데..)

$$y_3 = softmax(o_3)$$

뉴런 / 전체 출력값의 합

Learning

시간에 따라 state가 변한다

- **Backpropagation Through Time**

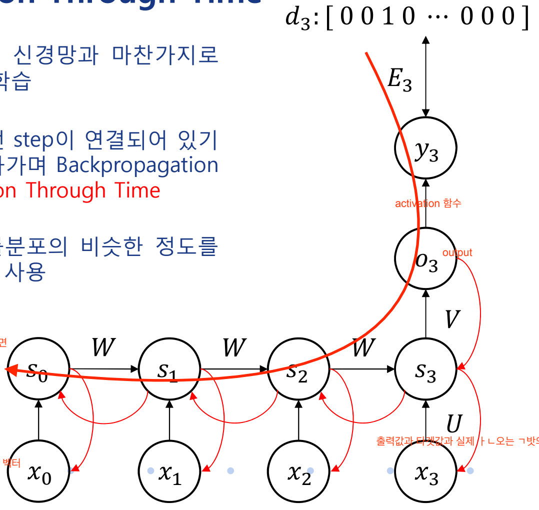
- RNN의 학습은 다른 인공 신경망과 마찬가지로 Backpropagation을 통해 학습

- RNN에선 현재 step과 이전 step이 연결되어 있기 때문에 시간을 거슬러 올라가며 Backpropagation을 진행 → Backpropagation Through Time

- 오차 함수는 두 개의 확률분포의 비슷한 정도를 나타내는 Cross-Entropy를 사용

target 값과 actual 값을 x,y 로 해서
x와 logx를 해서.......... (근데 자세히는 안함. 개념만 간단하게)
에러를 계산하는 방법 중 하나임
여기서는 예를들어 x =x프라임일 때 최소가 된다

이러한 계산법을
bptt라고 한다



출력값과 타겟값과 실제 ㅏ ㄴ오는 ㄱ밧의 차이를 제곱한 것을 최소화 시키는

Learning

- **Softmax**

- 값의 분포에 따른 출현 확률 (비율)
- $Let\ y = softmax(o)$
- $Then\ y_j = \frac{e^{o_j}}{\sum_k e^{o_k}}$ 출력값을 지수함수화 해서 더 과장 / 전체 출력 값 과장

$example)$

$o = (1, 2, 3)$

$$y = \left(\frac{e^1}{e^1+e^2+e^3}, \frac{e^2}{e^1+e^2+e^3}, \frac{e^3}{e^1+e^2+e^3}\right)$$

$$= (0.090, 0.245, 0.665)$$

큰거는 더 커지고 작은건 더 작게 나타난다

합하면 1!

- **Cross-Entropy**

- 두 개의 확률 분포의 비슷한 정도를 나타내는 지표.
- 0에 가까울수록 확률 분포가 비슷함
- $softmax$를 통해 나오는 예측값이 확률 분포이므로, 오차함수는 Cross-Entropy를 사용

$example)$

$$error\ = -d \cdot \ln(y) = -\sum_i d_i \cdot \ln(y_i)$$

1 1 이면 0
0 0 이어도 0

서로 다를수록 커지고
같을수록 작아진다

$$d = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}, \qquad y = \begin{pmatrix} 0.3 \\ 0.5 \\ 0.8 \end{pmatrix}$$

$$error\ = -(1\ *\ln(0.3) + 0\ *\ln(0.5) + 0\ *\ln(0.8))$$

서로 떨어질 수록 엔트로피가 커진다
엔트로피를 적용한 error function

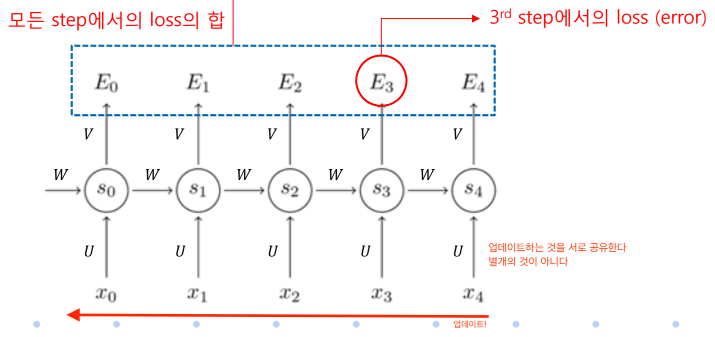
$$error \cong 1.204$$

Learning

- **Backpropagation Through Time**

- 학습 목표: Loss function E를 최소화하는 W, U, V를 찾는 것

  → Find $\frac{\partial E}{\partial W}, \frac{\partial E}{\partial U}, \frac{\partial E}{\partial V}$!


모든 step에서의 loss의 합
3rd step에서의 loss (error)
E_0 E_1 E_2 E_3 E_4
V
W
s_0 s_1 s_2 s_3 s_4
U
x_0 x_1 x_2 x_3 x_4
업데이트하는 것을 서로 공유한다
별개의 것이 아니다
업데이트!

Learning

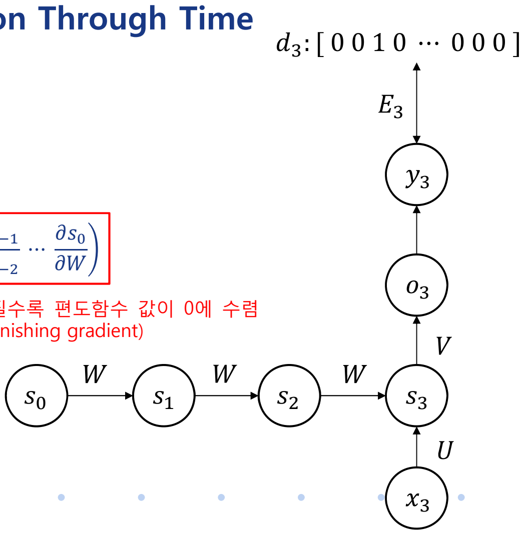
- **Backpropagation Through Time**

$$\frac{\partial E}{\partial W} = \sum_t \frac{\partial E_t}{\partial W}$$

$$\frac{\partial E_t}{\partial W} = \frac{\partial E_t}{\partial y_t}\frac{\partial y_t}{\partial o_t}\frac{\partial o_t}{\partial s_t}\frac{\partial s_t}{\partial W}$$

$$= \frac{\partial E_t}{\partial y_t}\frac{\partial y_t}{\partial o_t}\frac{\partial o_t}{\partial s_t}\boxed{\left(\frac{\partial s_t}{\partial s_{t-1}}\frac{\partial s_{t-1}}{\partial s_{t-2}}\cdots\frac{\partial s_0}{\partial W}\right)}$$

길이가 길어질수록 편도함수 값이 0에 수렴하는 문제 (Vanishing gradient)

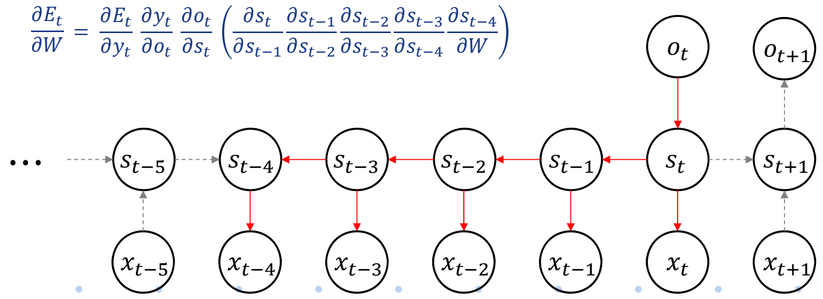


Learning

- **Vanishing gradient 해결 방법**

- **Truncated BPTT (단기 BPTT)**
- 모든 시간에 대한 은닉층의 값을 저장하는 것은 현실적으로 불가능하므로, 기준 길이보다 오래된 값은 반영하지 않도록 학습하는 방식

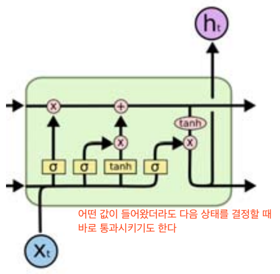
$$\frac{\partial E_t}{\partial W} = \frac{\partial E_t}{\partial y_t}\frac{\partial y_t}{\partial o_t}\frac{\partial o_t}{\partial s_t}\left(\frac{\partial s_t}{\partial s_{t-1}}\frac{\partial s_{t-1}}{\partial s_{t-2}}\frac{\partial s_{t-2}}{\partial s_{t-3}}\frac{\partial s_{t-3}}{\partial s_{t-4}}\frac{\partial s_{t-4}}{\partial W}\right)$$



Learning

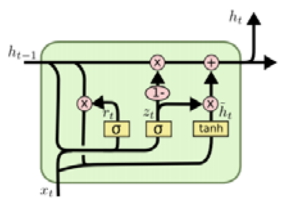
- **Vanishing gradient 해결 방법**

- **LSTM, GRU 등**
- RNN의 구조를 변형하여 vanishing gradient 문제를 해결



h_t
tanh
σ
σ
tanh
σ
어떤 값이 들어왔더라도 다음 상태를 결정할 때 바로 통과시키기도 한다
X_t

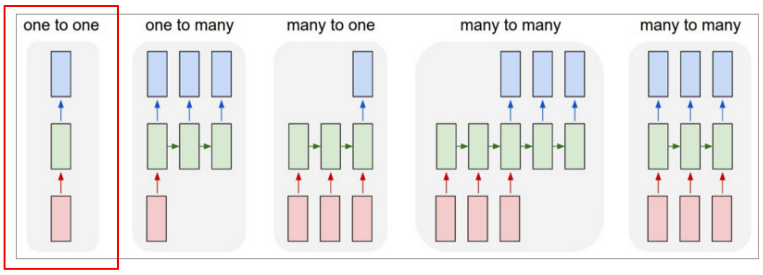
< LSTM의 간략 구조>



< GRU의 간략 구조>

Application

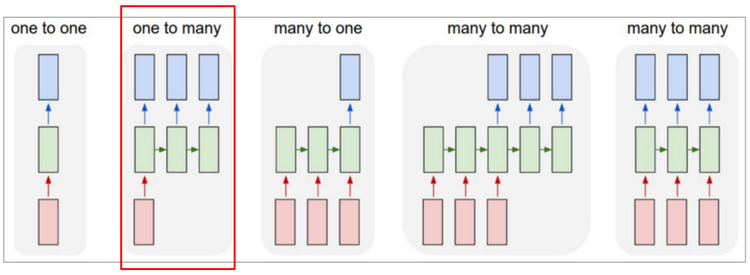
- **RNN을 이용한 다양한 입출력**
  - 무엇을 입력하고 출력하는지에 따라 유연하게 활용 가능



* 신경망의 기본적인 구조

Application

- **RNN을 이용한 다양한 입출력**



e.g. Image captioning

Image -> sequence of words

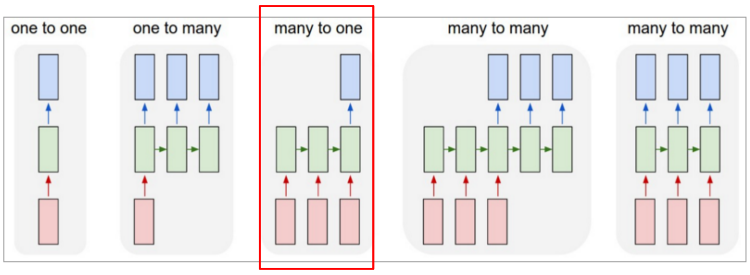
CNN 뉴랄넷을 통과하면
feature이 상대적으로…….

들판이라는 단어가 나타나고 그 다음 상태가 바뀌고 구름, 하늘, ...

들판 위에 구름이
낀 하늘이 있다.

Application

- **RNN을 이용한 다양한 입출력**



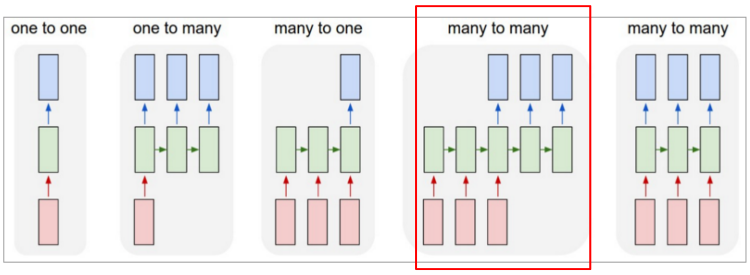
e.g. Sentiment analysis

sequence of words -> sentiment
주어진 문장의 감정 상태를 분석

너가 잘도 그러겠다.

Application

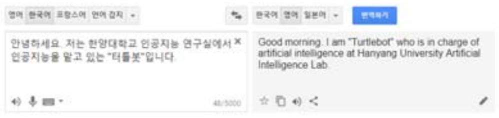
- **RNN을 이용한 다양한 입출력**



e.g. Machine Translation

sequence of words
-> sequence of words

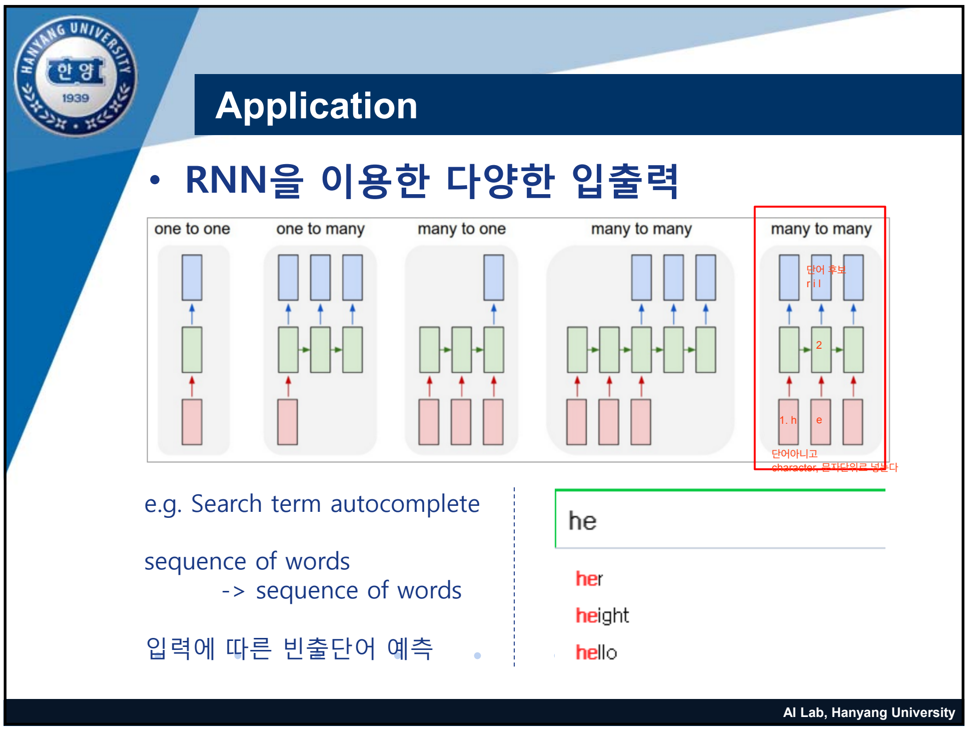
한국어 -> 영어 자동 번역



안녕하세요. 저는 한양대학교 인공지능 연구실에서 인공지능을 맡고 있는 "터틀봇"입니다.

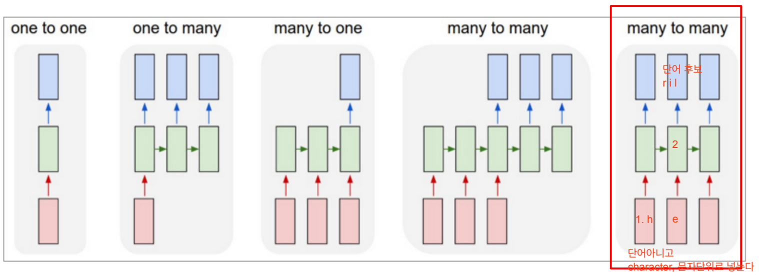
Good morning. I am "Turtlebot" who is in charge of artificial intelligence at Hanyang University Artificial Intelligence Lab.

문장을 준다는 것은 많은 시퀀스를 준다는 것
한글로 단어 3개를 입력하면 단어 3개가 영어로 나오는 것



Application

- **RNN을 이용한 다양한 입출력**



e.g. Search term autocomplete

sequence of words
-> sequence of words

입력에 따른 빈출단어 예측

he

her
height
hello

인공지능이란~