CNN and RNN

부산대학교
Information Security & IoT Lab
조동근
2017.11.29

목 차

1. Convolutional Neural Network

1-1 CNN 소개
1-2 Convolution, Pooling 소개

2. Recurrent Neural Network

2-1 RNN 소개
2-2 RNN 한계
2-3 LSTM(Long Short-Term Memory)

1. CNN

1-1 CNN 소개
1-2 Convolution, Pooling 소개

CNN(Convolutional Neural Networks)

- CNN은 인간의 시신경 구조를 모방한 기술로 이미지 프로세싱에서 시작해 글자인식, 이미지 인식, 사물인식에 이르기까지 인식에 필요한 특징을 자동으로 학습
- 또한, 형태 변이를 효과적으로 흡수할 수 있는 알고리즘
- 1998년 LeCun이 그의 논문 “Gradient-based learning applied to document recognition”에서 CNN을 이용하여 필기체를 성공적으로 인식함
- 2012년 Image Net 대회에서 CNN은 기존의 모든 알고리즘을 압도하는 성능으로 1등을 차지해 강력한 성능을 검증

ImageNet Classification with Deep Convolutional Neural Networks

Alex Krizhevsky University of Toronto kriz@cs.utoronto.ca
Ilya Sutskever University of Toronto ilya@cs.utoronto.ca
Geoffrey E. Hinton University of Toronto hinton@cs.utoronto.ca

Abstract

We trained a large, deep convolutional neural network to classify the 1.2 million high-resolution images in the ImageNet LSVRC-2010 contest into the 1000 different classes. On the test data, we achieved top-1 and top-5 error rates of 37.5% and 17.0% which is considerably better than the previous state-of-the-art. The

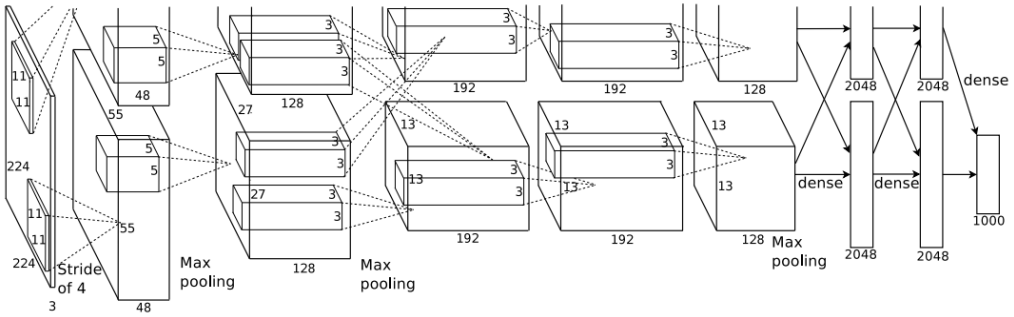


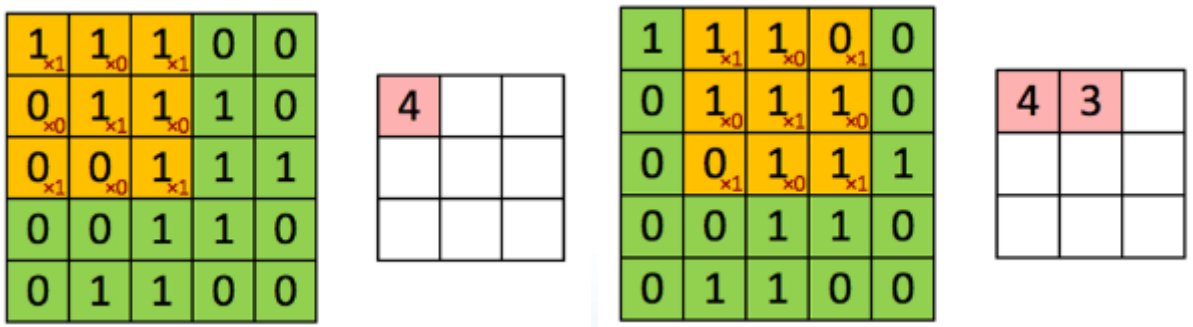
Figure 2: An illustration of the architecture of our CNN, explicitly showing the delineation of responsibilities between the two GPUs. One GPU runs the layer-parts at the top of the figure while the other runs the layer-parts at the bottom. The GPUs communicate only at certain layers. The network’s input is 150,528-dimensional, and the number of neurons in the network’s remaining layers is given by 253,440–186,624–64,896–64,896–43,264–4096–4096–1000.

CNN 역사

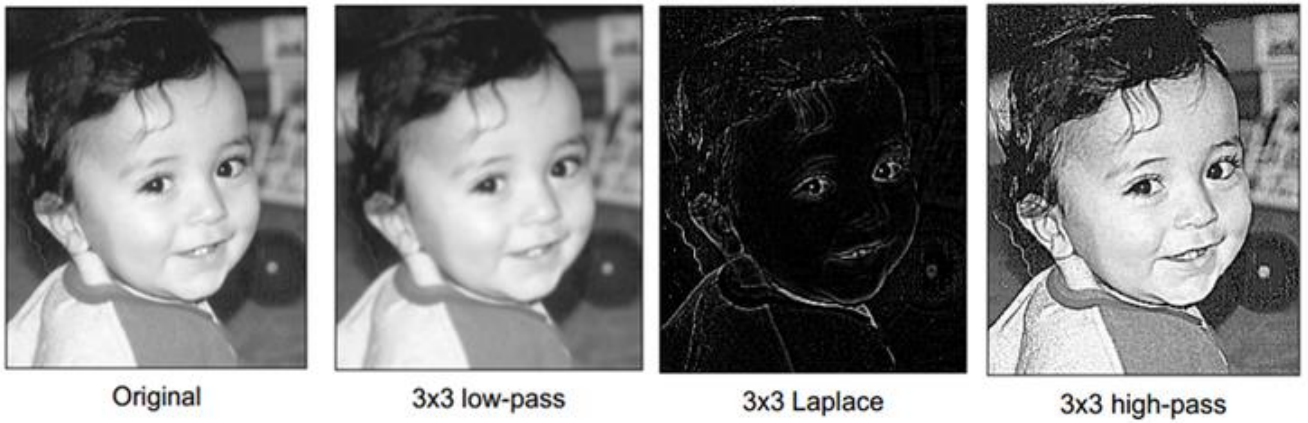
- 기존의 multi-layered neural network는 글자의 topology는 고려하지 않으므로 pixel단위의 row 데이터를 학습하기 때문에 데이터가 조금만 달라져도 좋은 결과를 기대하기 어려움
- 학습시간(training time), 망의 크기(network size), 변수의 개수(parameter) 문제점을 해결하기 위해 CNN이 제시

CNN 개념

- 영상에서의 특정 위치에 있는 픽셀들은 그 주변에 있는 일부 픽셀과 correlation 높고, 거리가 멀어지면 멀어질수록 그 영향은 감소함
- 영상 전체 영역에 대해 서로 동일한 연관성(중요도)로 처리하는 대신에 특정 범위에 한정에 처리한다면 훨씬 효과적일 것이 아이디어가 제시됨
- Convolution은 비전에서 주로 filter연산을 뜻하며 이미지의 특징(feature)를 찾기 위해 filtering을 수행



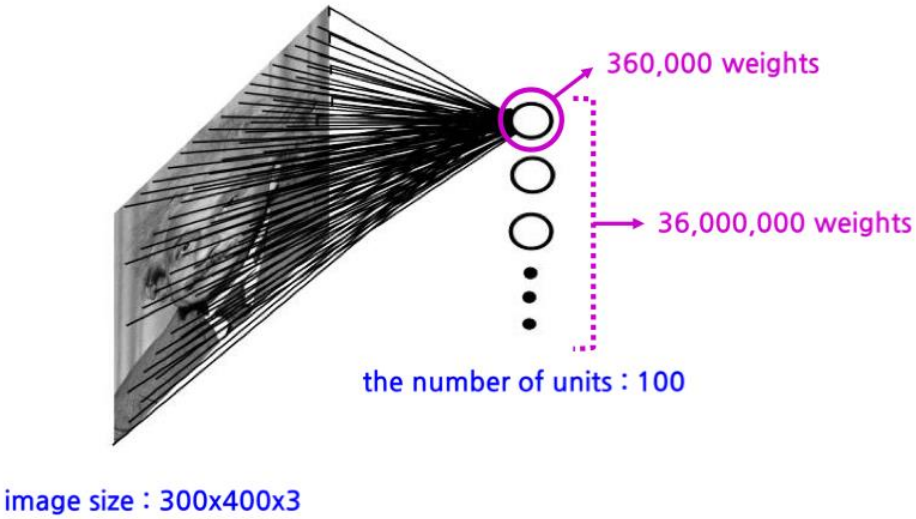
- 원영상에 대해, 다양한 3*3 filter 연산을 한 경우이며, 필터의 종류에 따라 각기 다른 특징을 끄집어 낼 수 있음



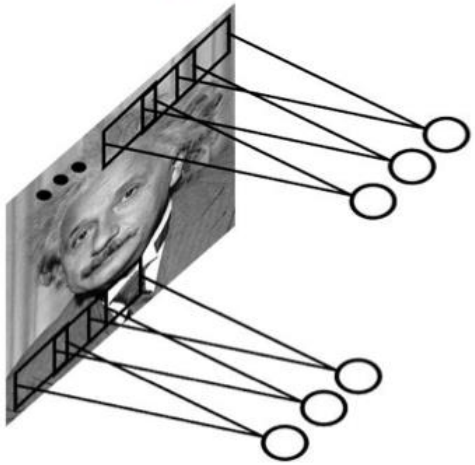
- CNN의 특징
  - Locality(Local Connectivity)
    - CNN은 Local 정보를 활용하여 인접한 신호들에 대한 correlation 관계를 비선형 필터를 적용하여 추출함.
    - 이런 필터를 여러 개 적용하면 다양한 local 특징을 추출할 수 있으며
    - 이러한 과정을 거치면서 영상의 크기는 줄어들며 global feature를 얻을 수 있게 됨
  - Shared weights
    - Filter를 반복적으로 적용함으로써 변수를 획기적으로 줄일 수 있음
    - Topology 변화에 무관한 항상성(invariance)를 얻을 수 있음

Motivation of CNN

- 전통적인 Neural Networks에서는, 모든 output unit이 모든 input unit과 interact
- 그러나, CNN에서는 특정 영역의 feature와 output이 interact하며 이것을 sparse interaction이라고 함
- Sparse interaction은 원본 이미지보다 크기가 작은 filter를 사용함으로써 실현됨



This full connectivity is wasteful and the huge number of parameters would quickly lead to over-fitting!

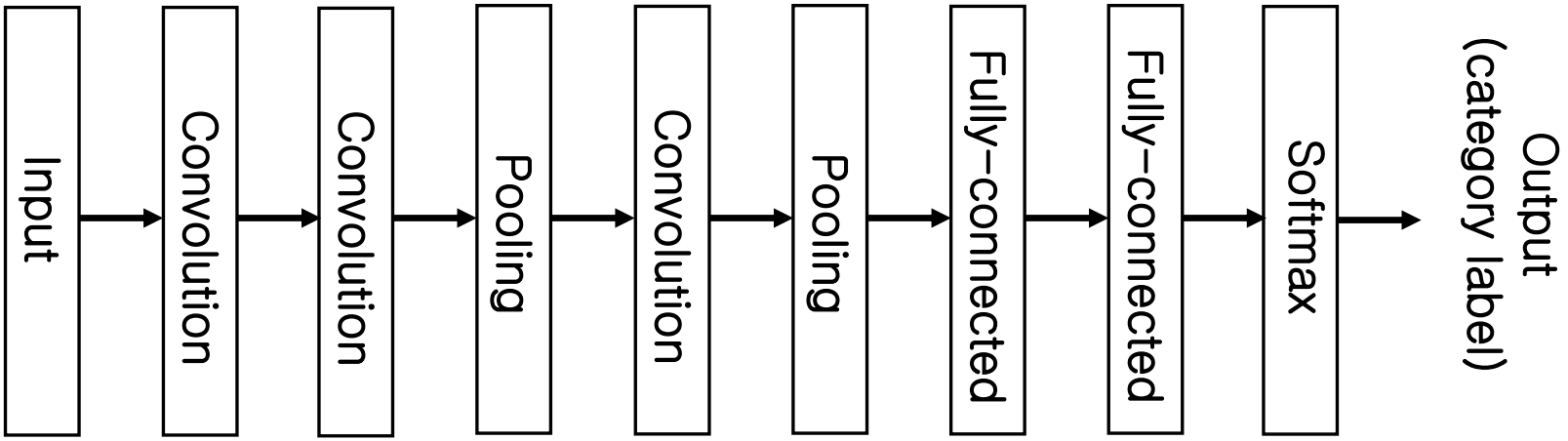


<Traditional Neural Networks>

<Convolutional Neural Networks>

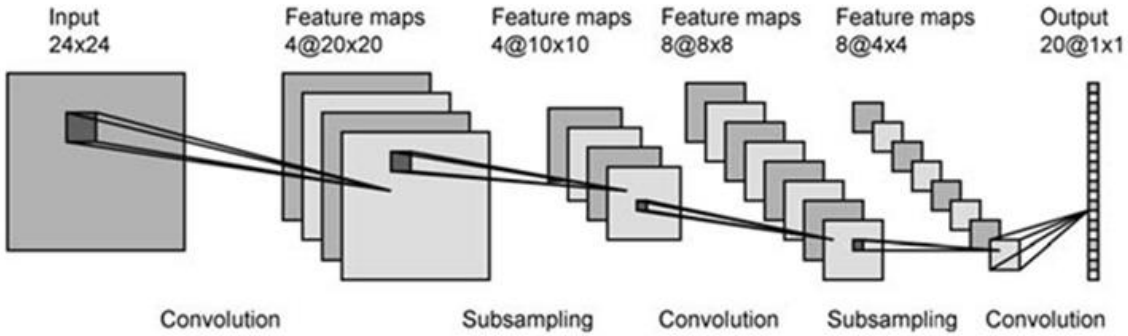
CNN Architecture

- CNN은 Convolution layer, Pooling layer 그리고 fully-connected layer로 구성됨
- Input layer 이후에 convolution layer와 pooling layer가 반복되고 마지막 1 또는 2 layer 에서 fully-connected layer로 구성

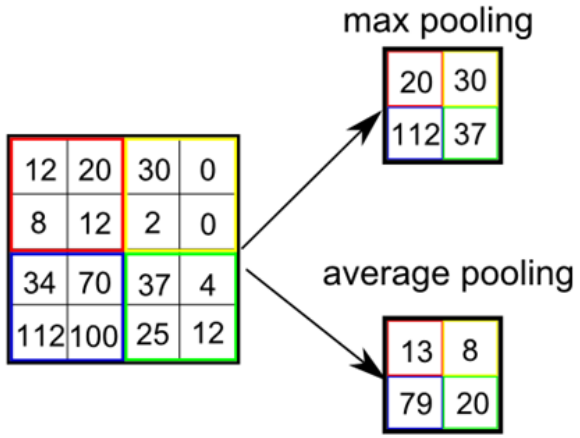


<CNN Architecture>

CNN Architecture

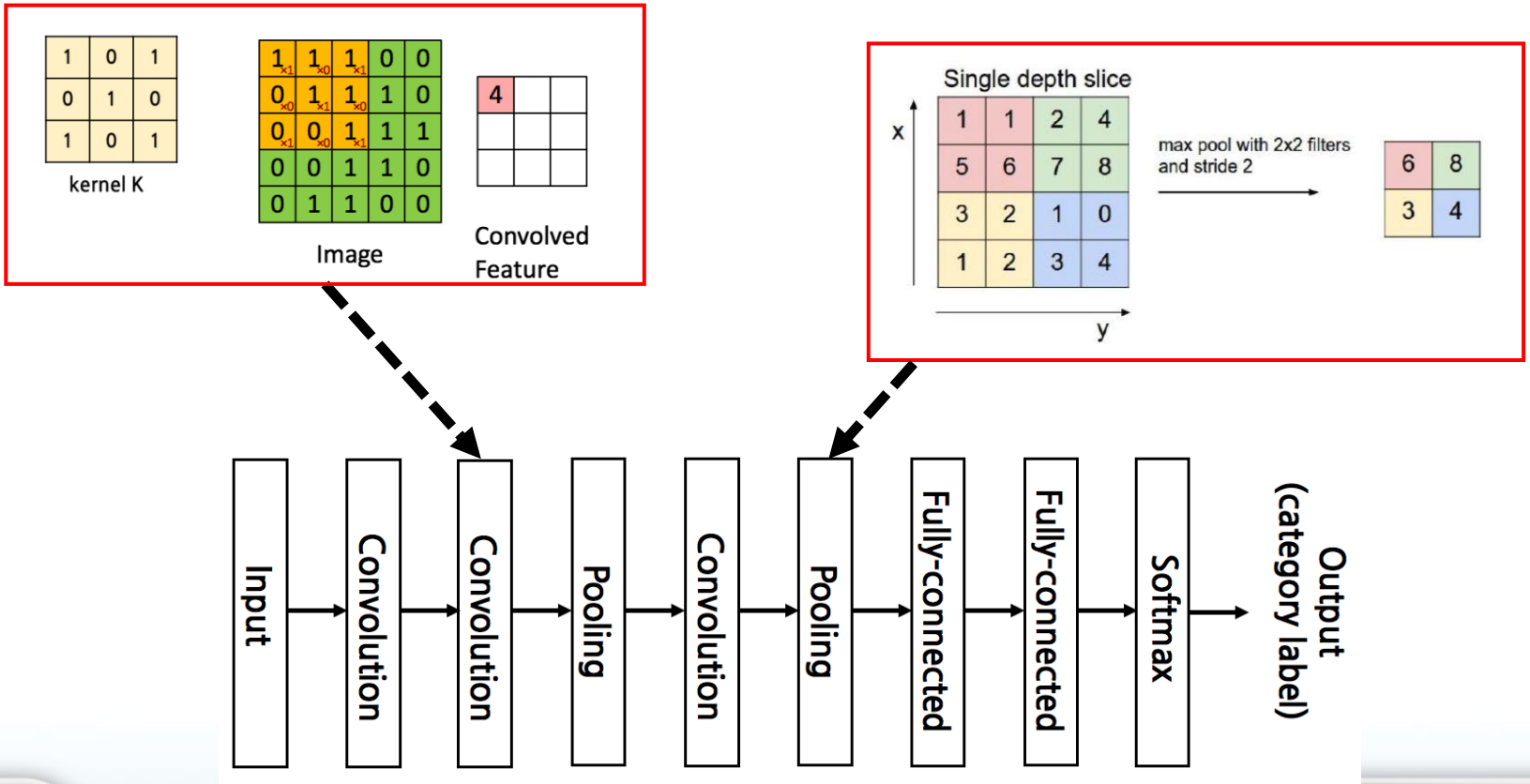


- **Convolution**
  - 입력 영상으로부터 Convolution(filter)를 통해, feature map을 생성
  - 여러 개의 다른 특징을 추출하고 싶다면 여러 개의 convolution kernel를 사용
- **Sub-sampling(Pooling)**
  - 가장 강한 신호만 전달하는 방식을 채택하여 가장 큰 값을 선택하는 방법인 max-pooling 을 주로 사용
  - 이동이나 변형 등에 무관한 학습 결과를 보이기 위해서 Convolution+Sub-sampling 과정을 여러 번 거쳐 대표할 수 있는 특징을 얻는 것이 중요

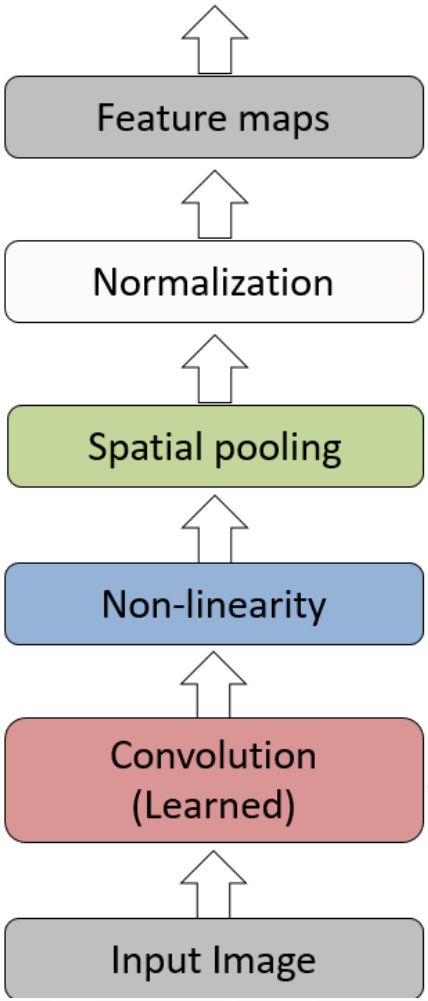


Convolution and Pooling Layers

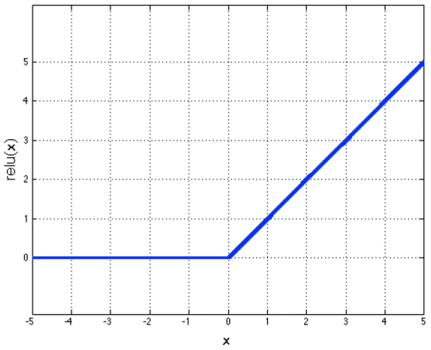
- Convolution layer는 filter의 명암 패턴과 유사한 패턴이 입력된 이미지가 어디에 있는지 검출
- Pooling layer는 overfitting을 막기위해 convolution layer의 출력 이미지의 특징은 살리고, 크기는 줄임

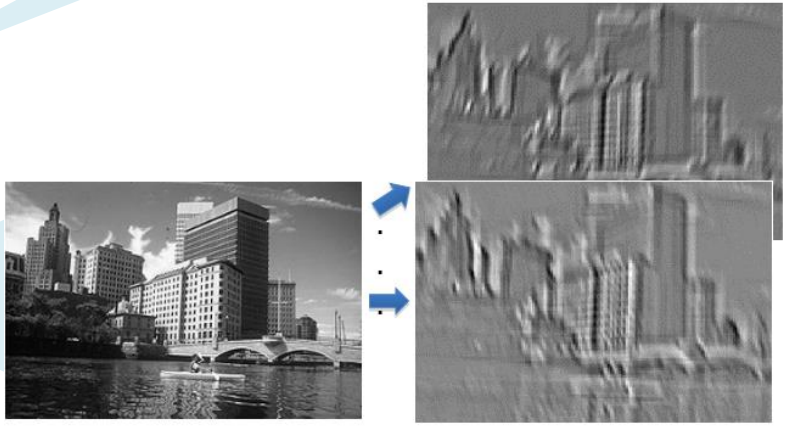


CNN



Rectified Linear Unit (ReLU)



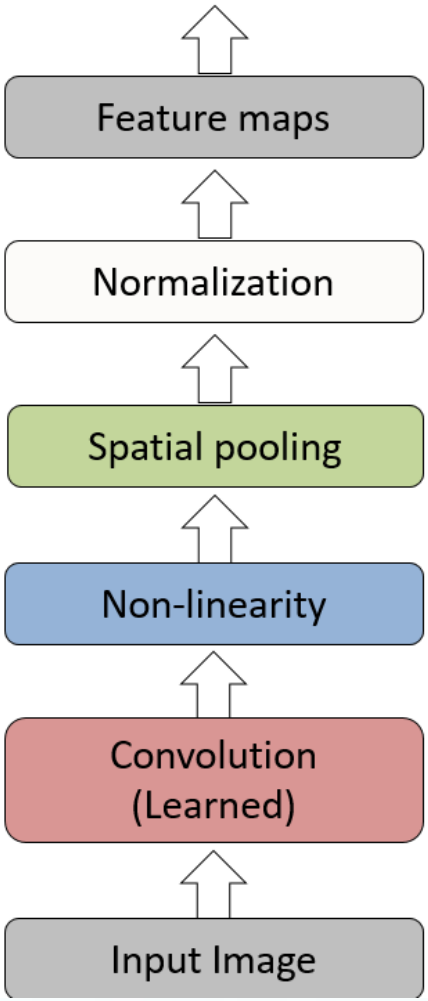


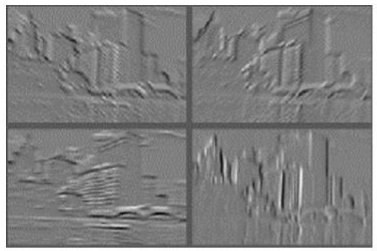
Input Feature Map

Convolution (filtering) 통해 대표적인 feature 를 뽑아냄

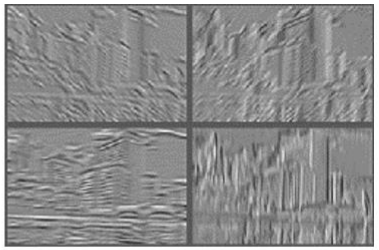
출처 : Convolutional Neural Networks by Jia-Bin Huang, Univ. of Illinois

CNN

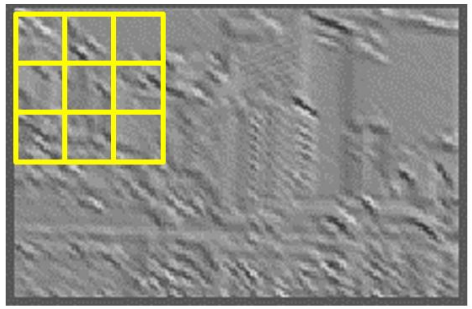




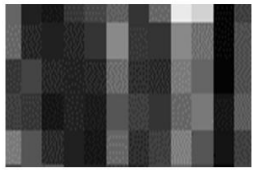
Feature Maps



Feature Maps
After Contrast
Normalization



Max pooling

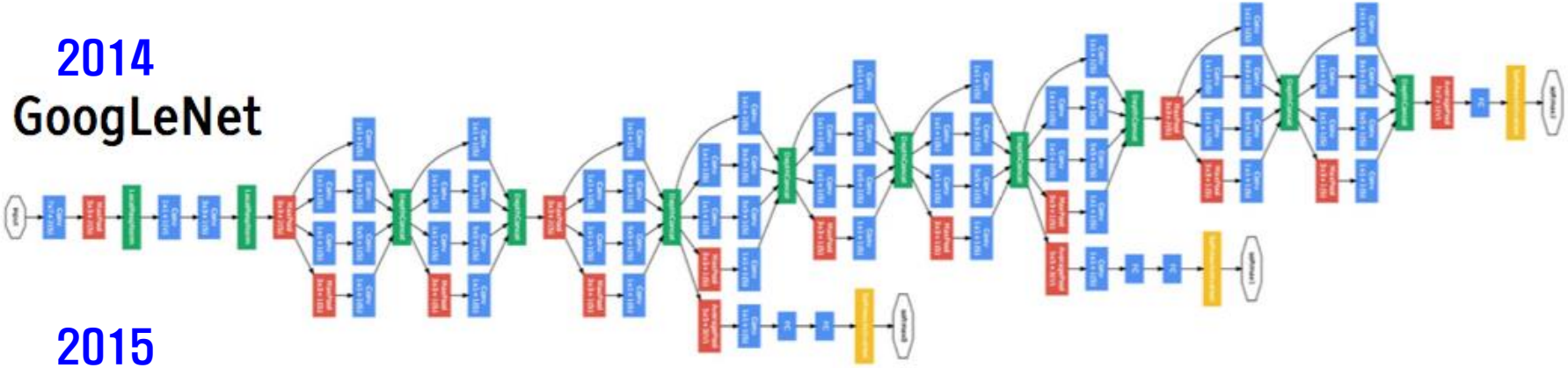


Pooling에선 특징을 (어느정도) 유지하면서 이미지 down sizing을 통해 복잡도를 줄임. Overfitting도 개선됨

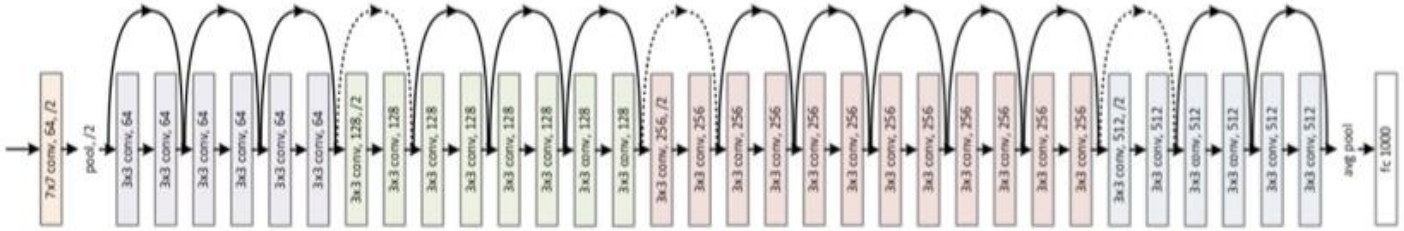
Popular CNN Architectures

2014
VGG

2014
GoogLeNet



2015
ResNet

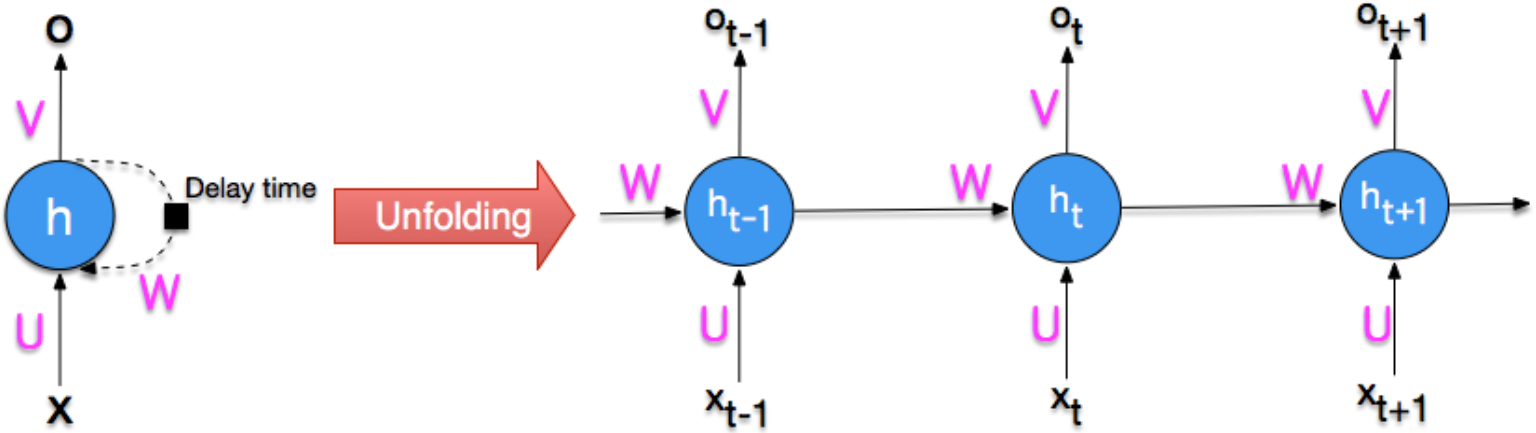


2. Recurrent Neural Network

2-1 RNN 소개
2-2 RNN 한계
2-3 LSTM(Long Short-Term Memory)

RNN(Recurrent Neural Networks)

- RNN은 Deep Learning 알고리즘 중 순차적인 데이터를 학습하여 classification 또는 prediction을 수행
- 기존의 DNN(Deep Neural Networks)의 경우 각 layer마다 parameter들이 독립적이었으나, RNN은 이를 공유하고 있음
- 따라서 현재의 출력 결과는 이전 time step의 결과에 영향을 받으며, hidden layer는 일종의 메모리 역할을 하게됨



<RNN Architecture>

Applications of RNN

텍스트 생성

이전 단어들을 보고 다음 단어가 나올 확률을 예측하는 언어 모델을 기반으로 텍스트를 생성하는 generative model을 만들 수 있음

언어 번역

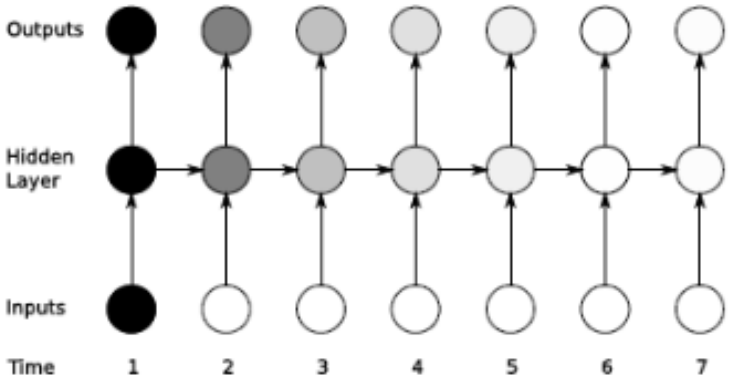
Word sequence를 입력으로 사용하여 번역할 언어의 word sequence로 출력

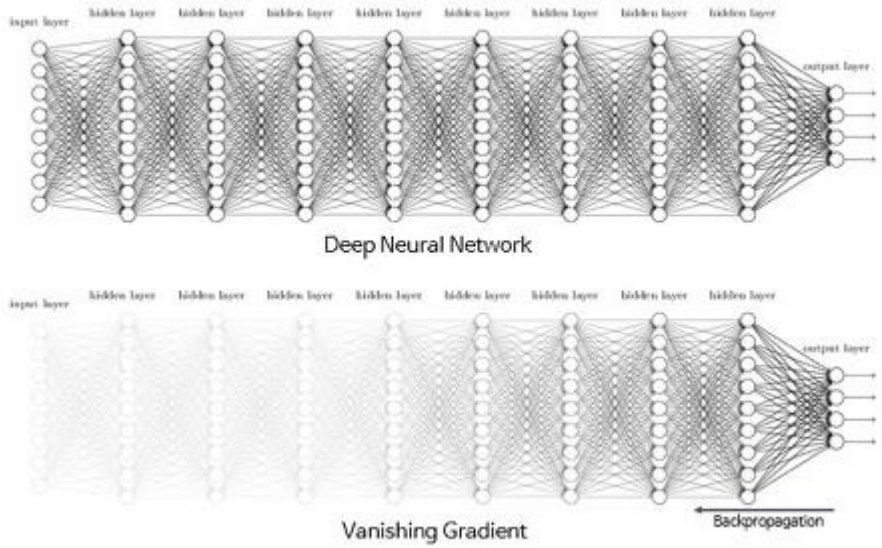
음성 인식

Acoustic signal을 입력으로 받아 phonetic segment의 sequence 또는 probabillity distribution을 추측

RNN의 문제점

- Vanishing gradient problem은 networks의 weight가 업데이트 되는 과정에서 gradient(weight에 대한 일종의 업데이트 비율)가 1보다 작은 값이 계속 곱해지면서 gradient가 사라지는 현상
- 따라서, 먼 과거의 상태는 현재의 학습에 아무런 영향을 미치지 못하게 됨

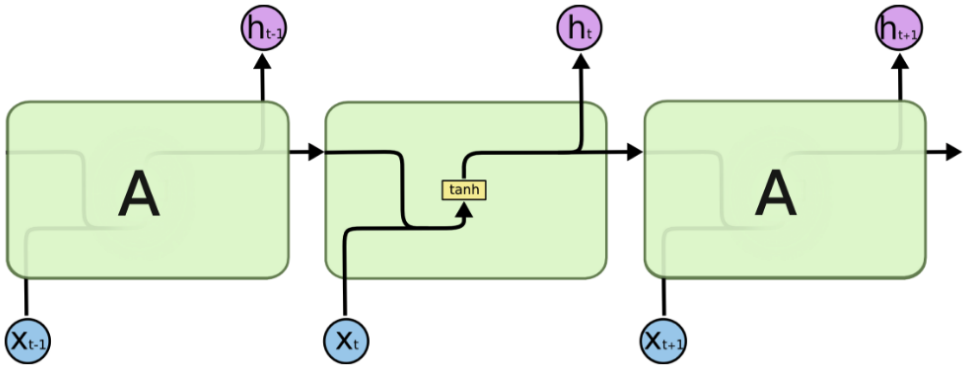




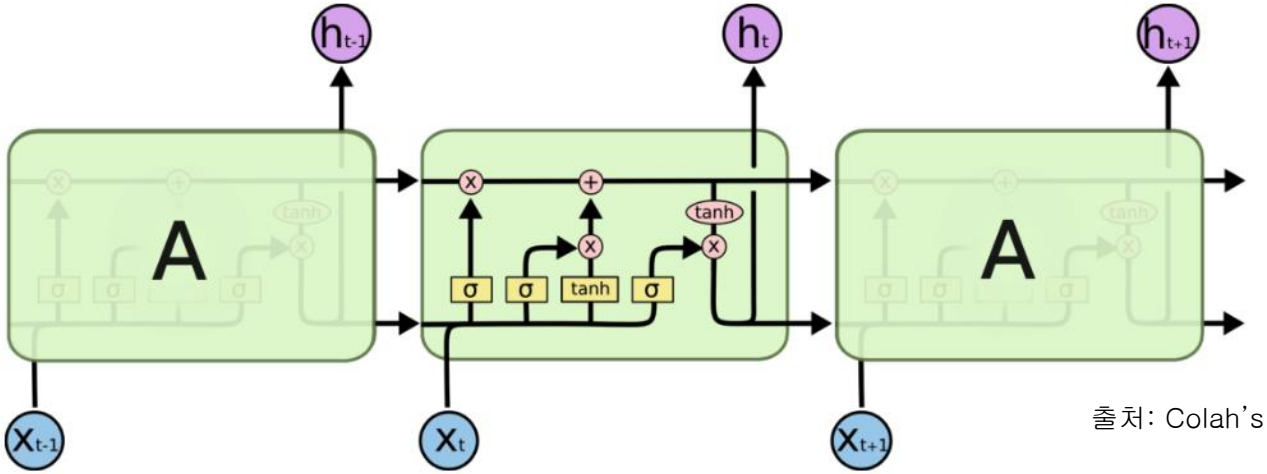
<Vanishing Gradient Problem>

RNN 문제점 해결책

- LSTM(Long Short Term Memory)은 Vanishing gradient problem의 장기 의존성 문제를 해결하기 위하여 hidden layer에 적용되는 architecture
- LSTM은 3개의 gates(input, forget, output)로 현재 노드의 상태 정보를 제어
- Forget gate는 이전 상태 정보를 저장할지를 결정하고, input gate는 입력되는 새로운 정보를 저장할지 결정함, output gate는 갱신된 cell의 출력값을 제어함



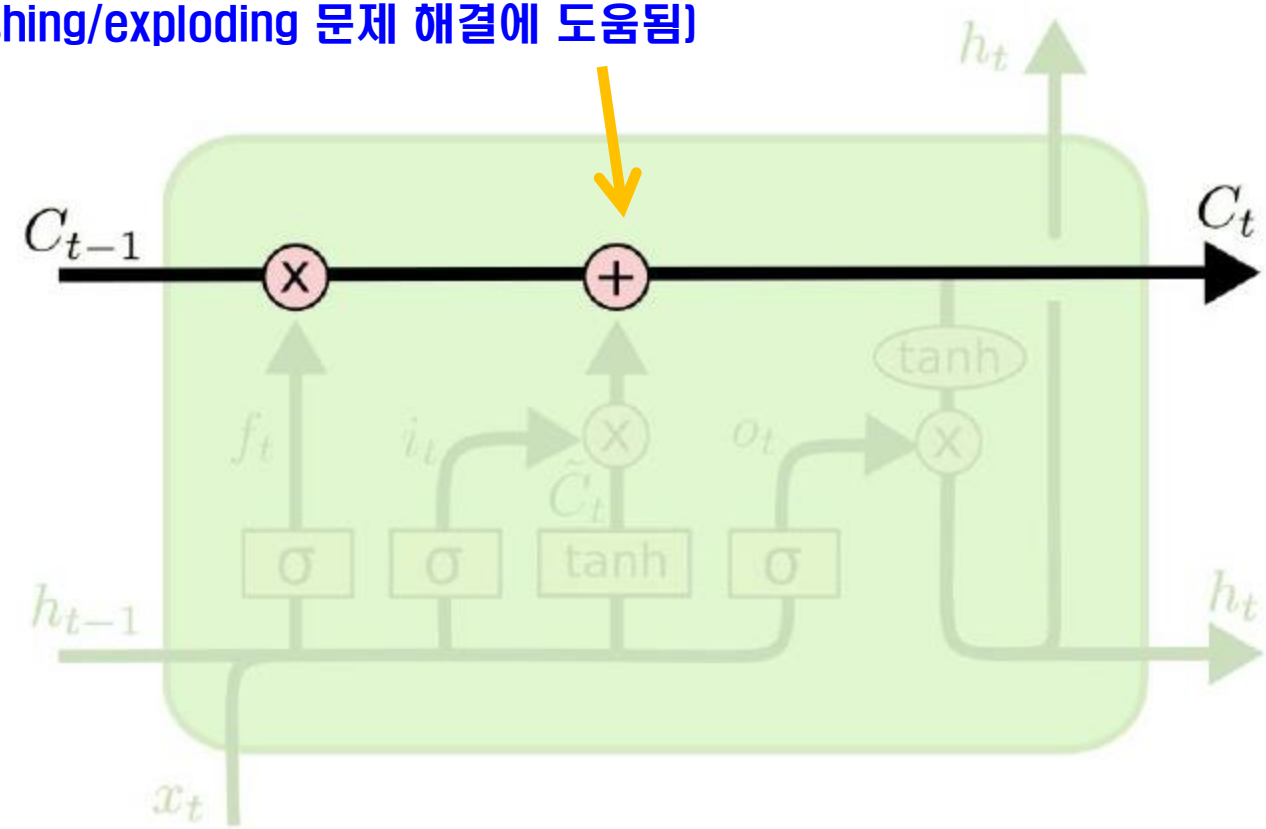
<RNN에 LSTM 적용>



출처: Colah's blog

LSTM

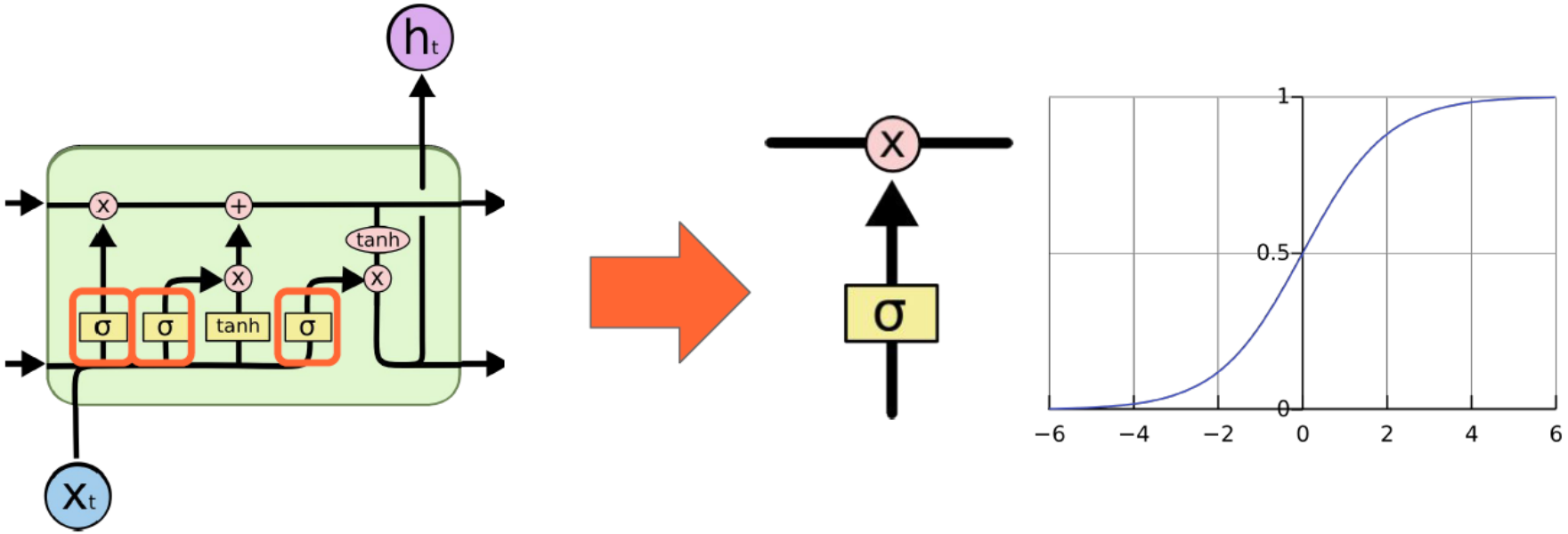
- Ct 는 기본 셀의 상태를 의미함
- 또한, 값을 반영하기 위해 "product" 대신 "sum" 을 사용함 (vanishing/exploding 문제 해결에 도움됨)



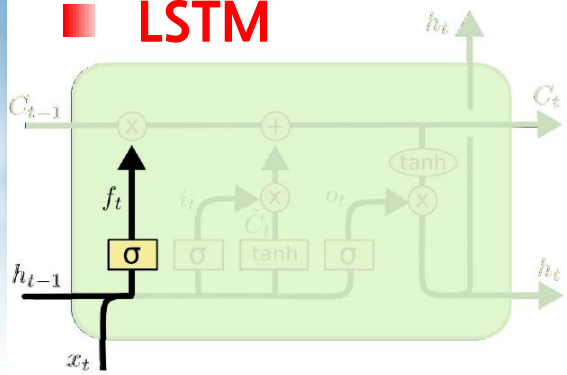
출처: Colah's blog

LSTM

– 세가지 gate는 모두 “sigmoid” 유닛에 의해 제어됨



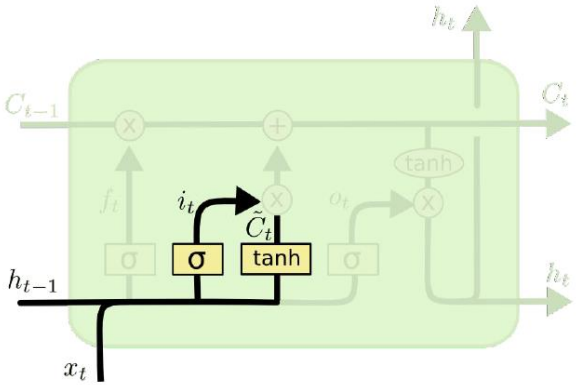
LSTM



<Forget gate>

$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \; + \; b_f\right)$$

Concatenate



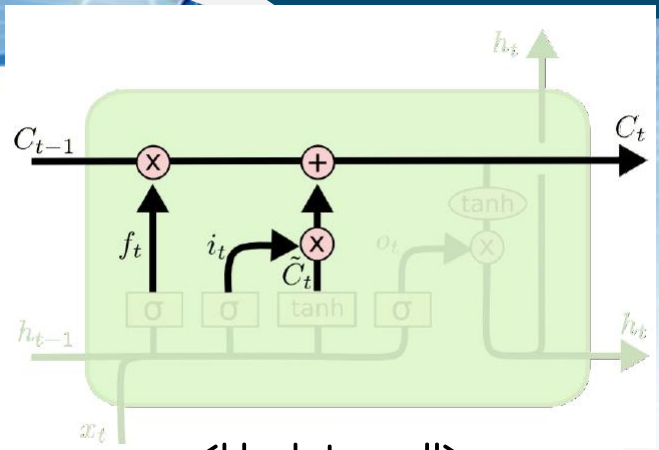
<Input gate>

Input Gate Layer

$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \; + \; b_i\right)$$

New contribution to cell state

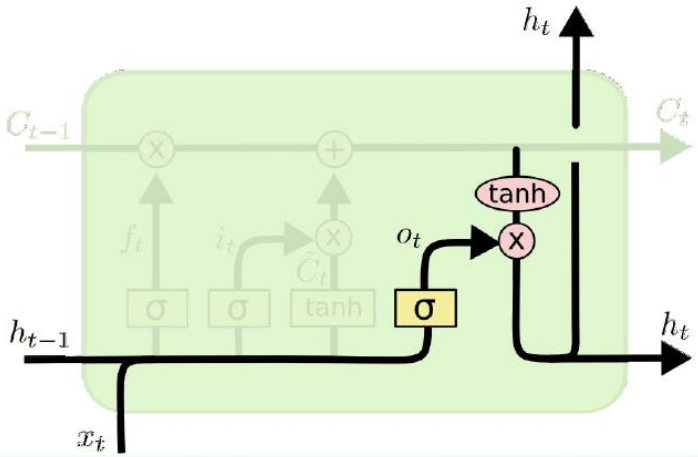
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \; + \; b_C)$$ 기존 neuron과 동일함



<Update cell>

Update Cell State (memory):

$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$



Output Gate Layer

$$o_t = \sigma\left(W_o \; [h_{t-1}, x_t] \; + \; b_o\right)$$

Output to next layer

$$h_t = o_t * \tanh\left(C_t\right)$$

처: Colah's blog

Thank you!