인공지능개론

Recurrent Neural Networks

Neural Networks

❖ 컨볼루션 신경망의 학습을 위한 목적함수
- 분류류 문제
  - 교차 엔트로피(cross entropy)
    - 학습 데이터 출력 : t_{ik}
    - 컨볼루션 신경망 출력 : $y_k(\boldsymbol{x}_i, \boldsymbol{w})$

$$E(\boldsymbol{w}) = -\log \sum_{i=1}^{N} \sum_{k=1}^{K} t_{ik} \log y_k(\boldsymbol{x}_i, \boldsymbol{w})$$

- 회귀 문제

$$E(\boldsymbol{w}) = \frac{1}{2} \sum_{i=1}^{N} \sum_{k=1}^{K} (t_{ik} - y_k(\boldsymbol{x}_i, \boldsymbol{w}))^2$$

❖ 적용 가능 학습 알고리즘
- 경사 하강법
- 경사 하강법의 변형

Neural Networks

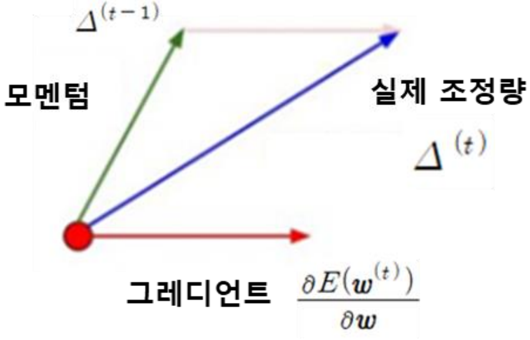
❖ 경사 하강법(Gradient descent method)

$$\boldsymbol{w}^{(t+1)} = \boldsymbol{w}^{(t)} - \eta \frac{\partial E(\boldsymbol{w}^{(t)})}{\partial \boldsymbol{w}}$$

❖ 모멘텀을 고려한 경사 하강법

$$\Delta^{(t)} = \alpha \Delta^{(t-1)} + \eta \frac{\partial E(\boldsymbol{w}^{(t)})}{\partial \boldsymbol{w}}$$

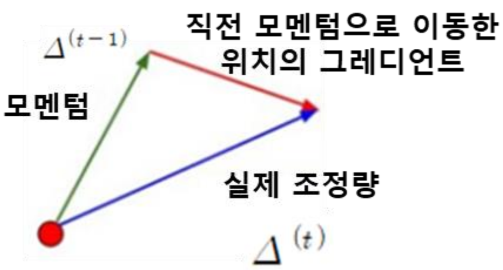
$$\boldsymbol{w}^{(t+1)} = \boldsymbol{w}^{(t)} - \Delta^{(t)}$$



❖ NAG(Nesterov accelerated gradient) 방법

$$\Delta^{(t)} = \alpha \Delta^{(t-1)} + \eta \frac{\partial E(\boldsymbol{w}^{(t)} - \alpha \Delta^{(t-1)})}{\partial \boldsymbol{w}}$$

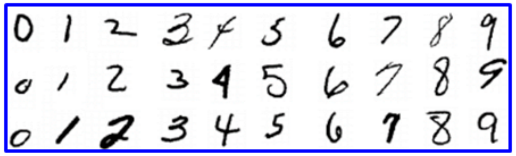
$$\boldsymbol{w}^{(t+1)} = \boldsymbol{w}^{(t)} - \Delta^{(t)}$$



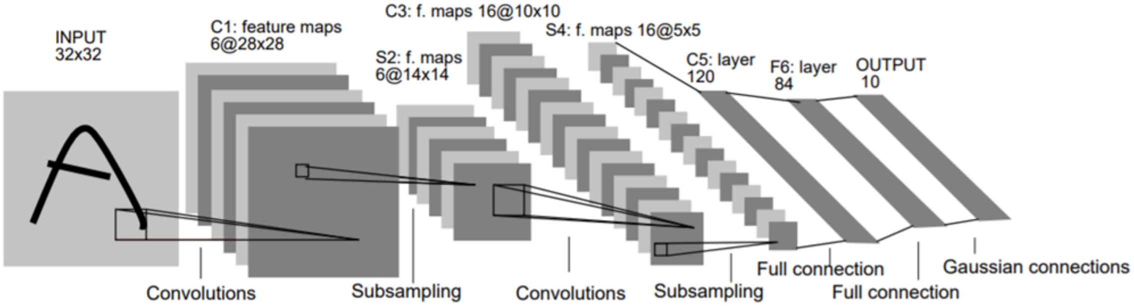
Neural Networks

❖ LeNet 모델

- Yann LeCun 등의 제안(1998)
- LeNet5 모델
  - 5 계층 구조: Conv-Pool-Conv- Pool-Conv-FC-FC(SM)
- 입력 : 32x32 필기체 숫자 영상 (**MNIST** 데이터)



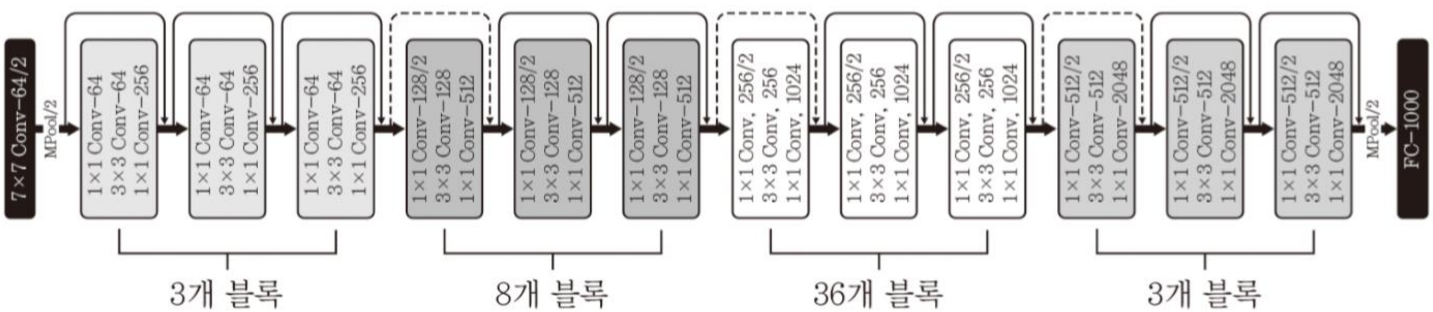
- 풀링 : 가중치x(2x2블록의 합) + 편차항
- 시그모이드 활성화 함수 사용
- 성능: 오차율 0.95%(정확도: 99.05%)



Neural Networks

❖ ResNet (Residual Net)

- 카이밍 허 등이 개발
- 2015년 ILSVRC에서 우승(상위-5 오류율: 3.75%)
- 152개 층의 모델
  - Conv-Mpool
    -[Conv-ReLU-Conv-ReLU-Conv-ReLU]x3
    -[Conv-ReLU-Conv-ReLU-Conv-ReLU]x8
    -[Conv-ReLU-Conv-ReLU-Conv-ReLU]x36
    -[Conv-ReLU-Conv-ReLU-Conv-ReLU]x3
    -APool-FC-SM

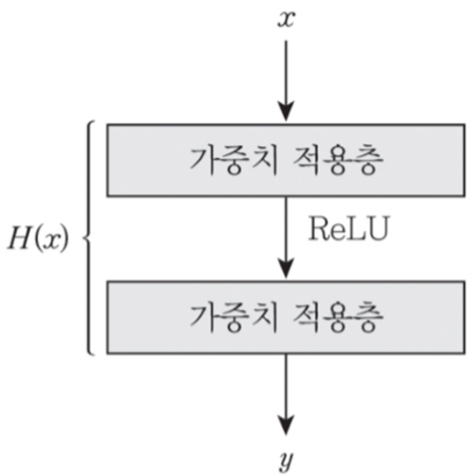


Neural Networks

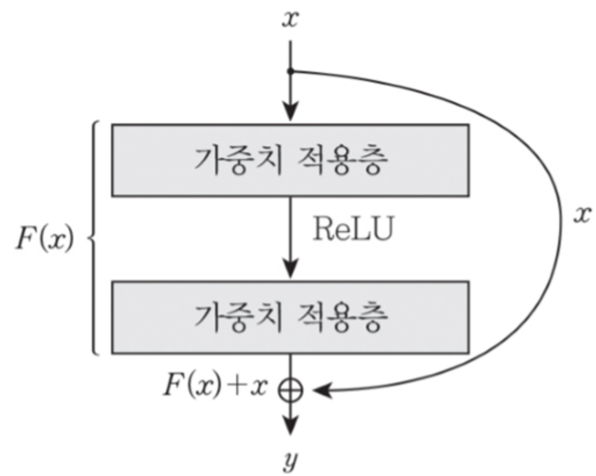
❖ **ResNet – cont.**

- **다수의 층 사용**
  - 상위 계층에서 의미있는 특징 추출 가능
  - 다수 계층 사용시 기울기 소멸 문제 발생

- **잔차 모듈(residual module)**



기존 신경망



ResNet의 잔차 모듈

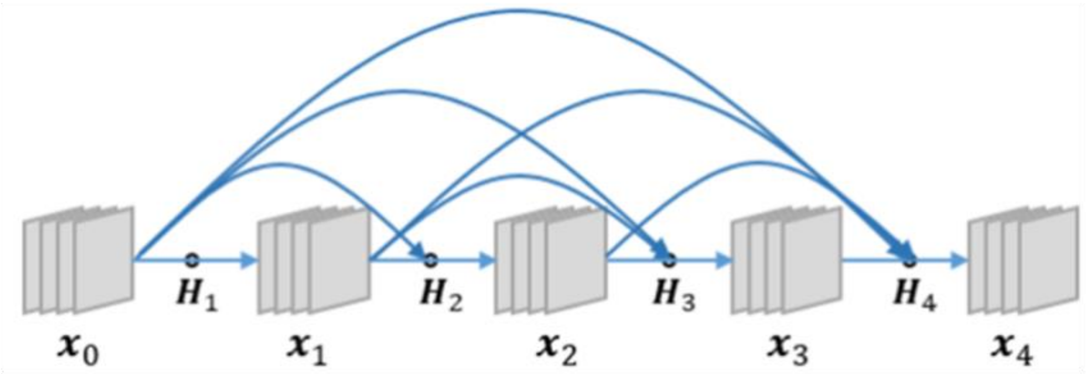
$$y = H(x)$$

$$F(x) = y - x; \quad y = F(x) + x.$$

Neural Networks

❖ **DenseNet**

- 가오 후앙(Gao Huang) 등이 개발 (2016)
- 각 층은 모든 앞 단계에서 올 수 있는 지름길 연결 구성



$$x_i = H_i([x_0, x_1, \cdots, x_{i-1}])$$

Neural Networks

❖ 재귀 신경망(Recurrent Neural Networks, RNN, 순환 신경망)

- **서열 데이터**(Sequence data)
  - 음성, 자연어 문장, 동영상, 주가 변동 등의 데이터
  - 구성요소가 순차적으로 발생하거나 구성요소 간에 순서 존재
  - 이전 값들이 현재 값에 영향을 주는 경우

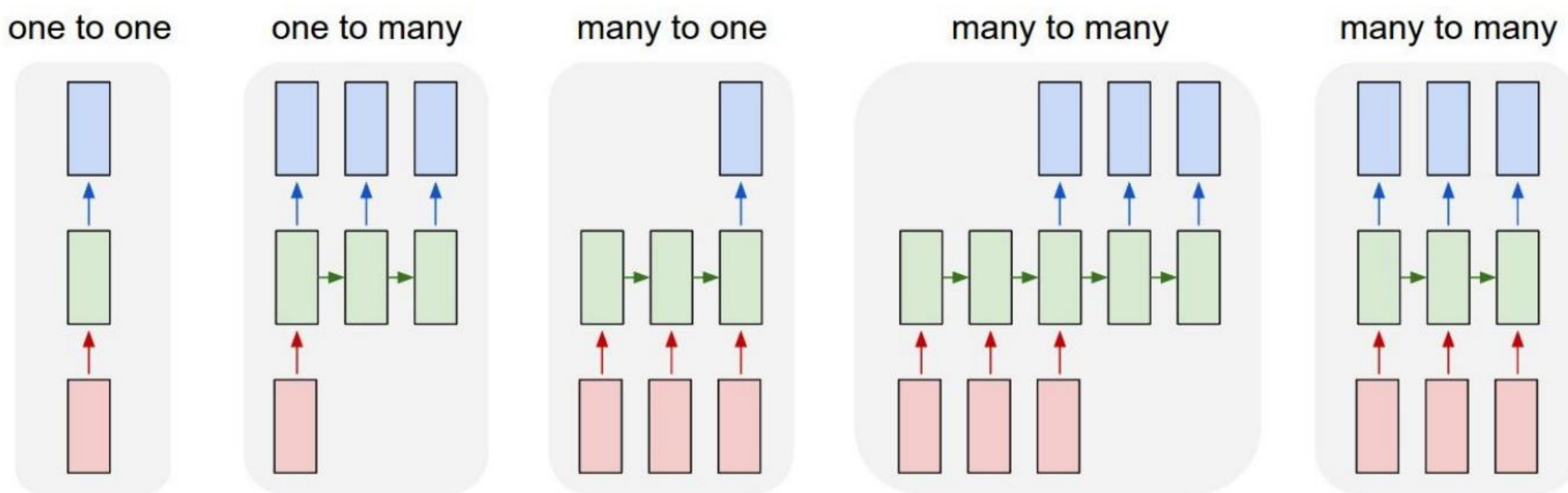
- 서열 데이터의 **분류, 예측**에서 **현재 시점**의 값과 **이전 시점의** 값들을 고려 필요

- 재귀 신경망은 서열 데이터의 학습 및 추론에 적합한 모델

- **기계 번역, 음성 인식, 필기체 인식, 영상 주석달기, 동영상에서 행동 인식, 작곡 및 작사** 등 다양한 응용 분야에서 활용

Neural Networks

Recurrent Neural Networks: Process Sequences



Vanilla Neural Networks

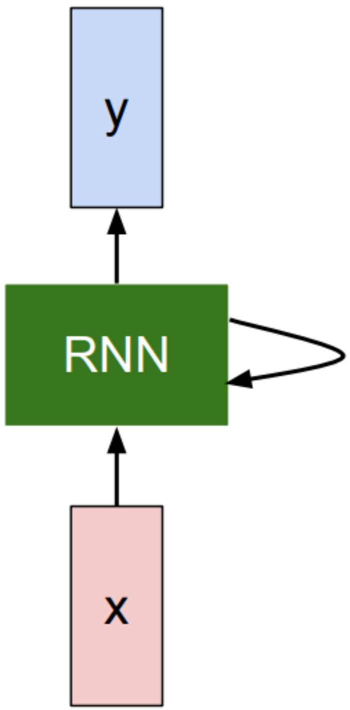
e.g. **Sentiment Classification**
sequence of words -> sentiment

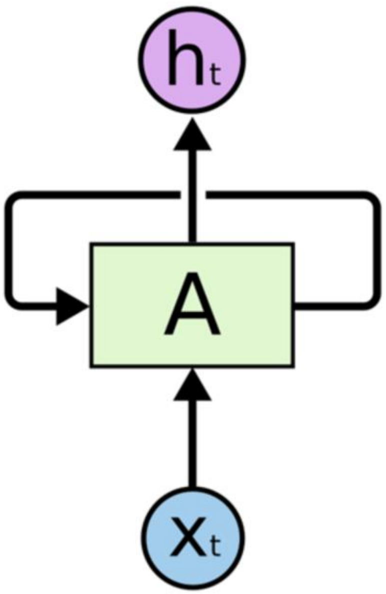
e.g. **Video classification on frame level**

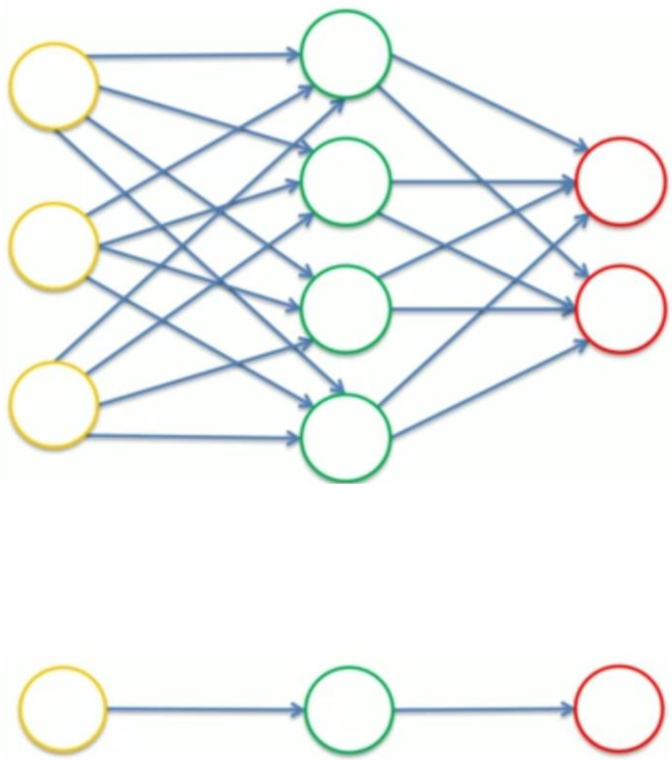
e.g. **Image Captioning**
image -> sequence of words

e.g. **Machine Translation**
seq of words -> seq of words

Neural Networks







Neural Networks

We can process a sequence of vectors **x** by applying a **recurrence formula** at every time step:

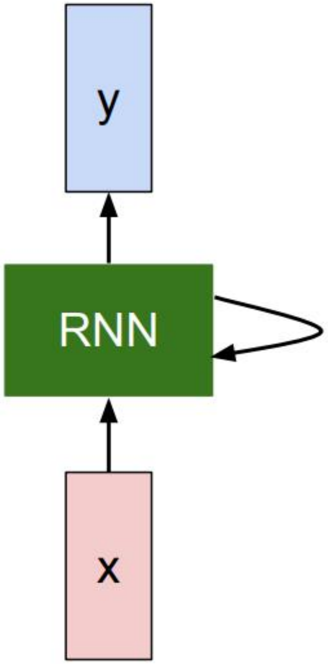
$$\boxed{h_t} = \boxed{f_W}(\boxed{h_{t-1}}, \boxed{x_t})$$

new state

old state

input vector at some time step

some function with parameters W

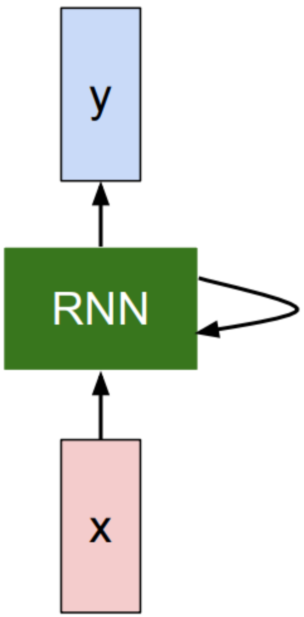


Notice: the same function and the same set of parameters are used at every time step.

Neural Networks

(Vanilla) Recurrent Neural Network

The state consists of a single *"hidden"* vector **h**:

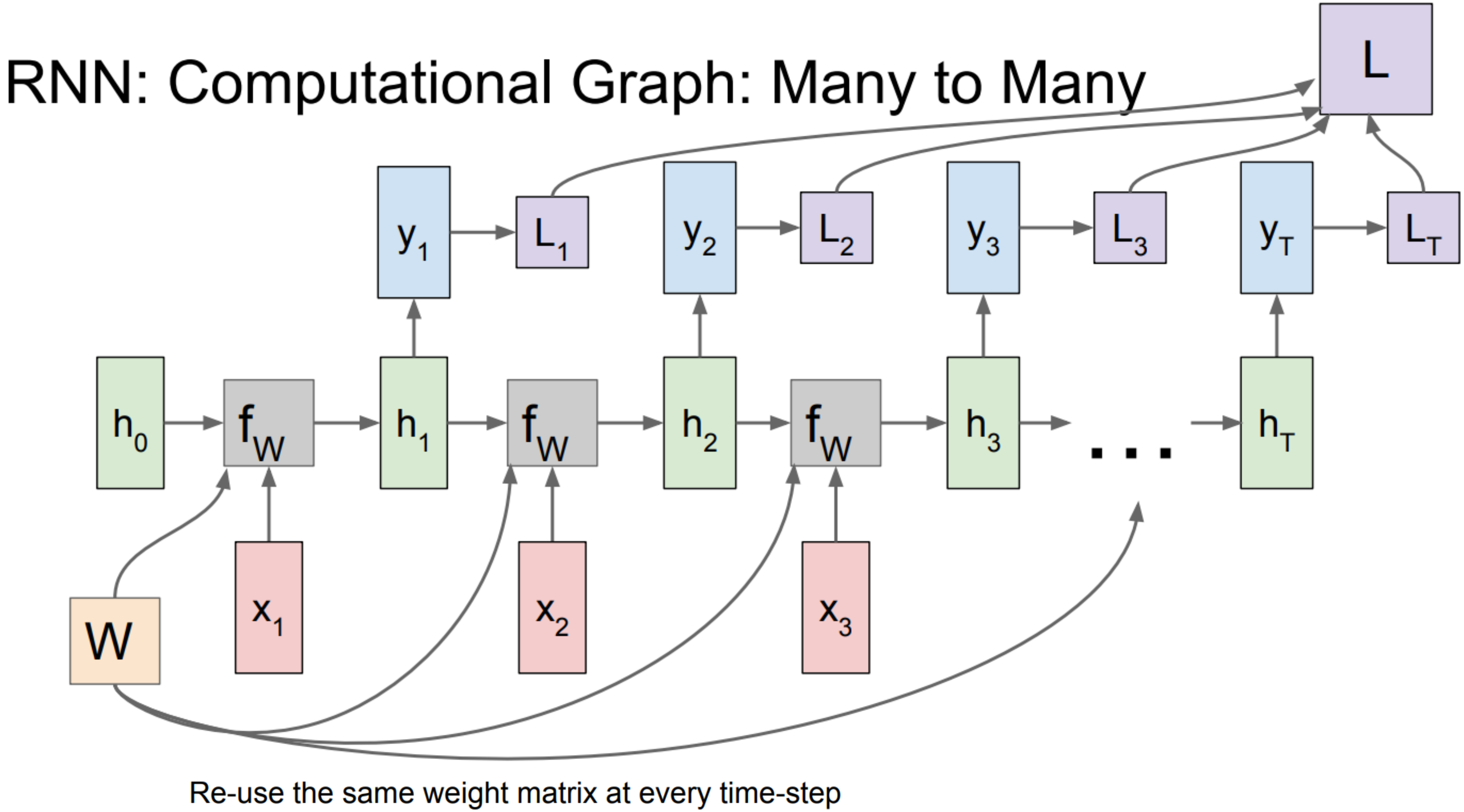


$$h_t = f_W(h_{t-1}, x_t)$$

$$h_t = \tanh(W_{hh} h_{t-1} + W_{xh} x_t)$$

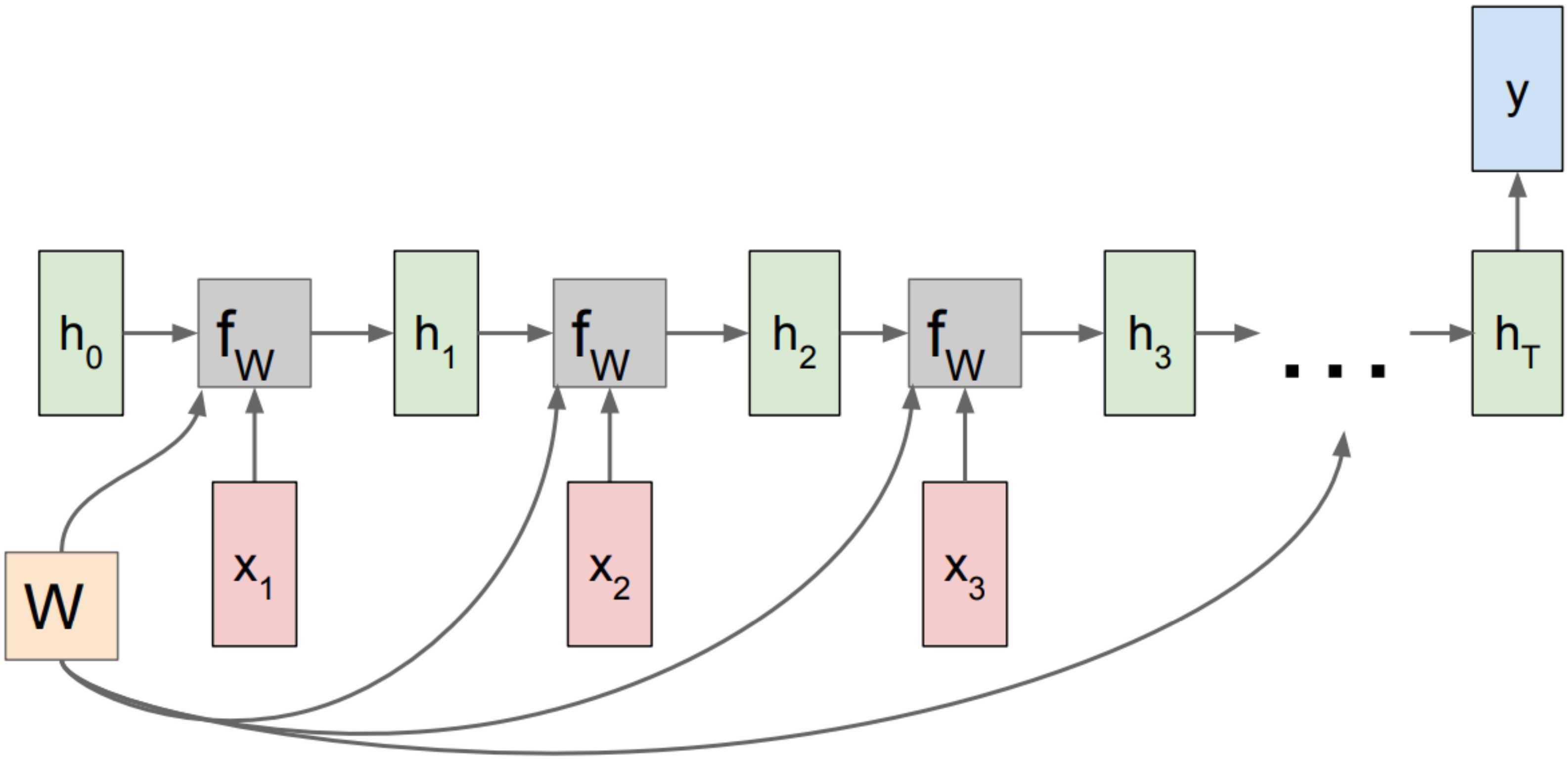
$$y_t = W_{hy} h_t$$

Neural Networks



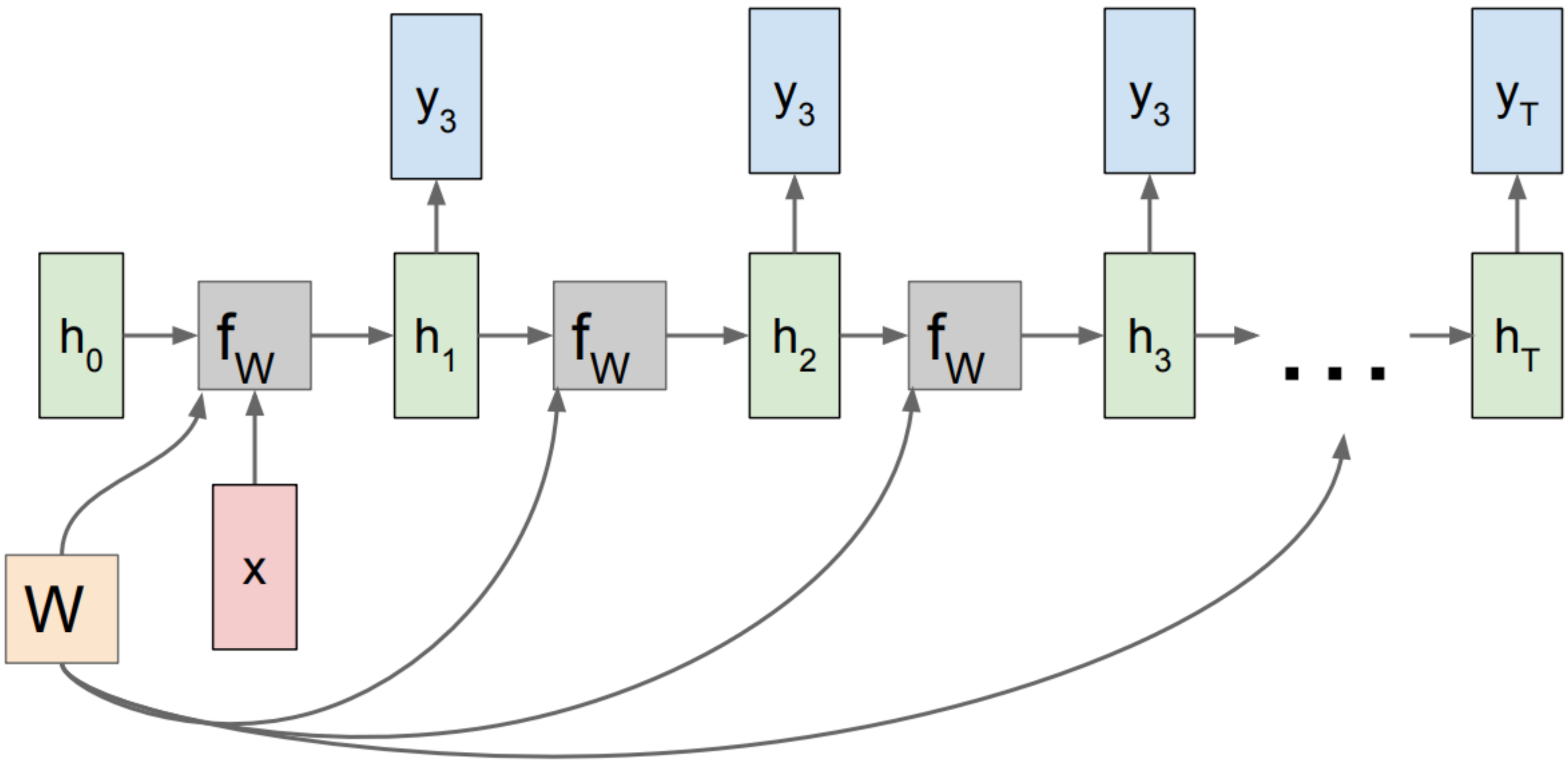
Neural Networks

RNN: Computational Graph: Many to One



Neural Networks

RNN: Computational Graph: One to Many



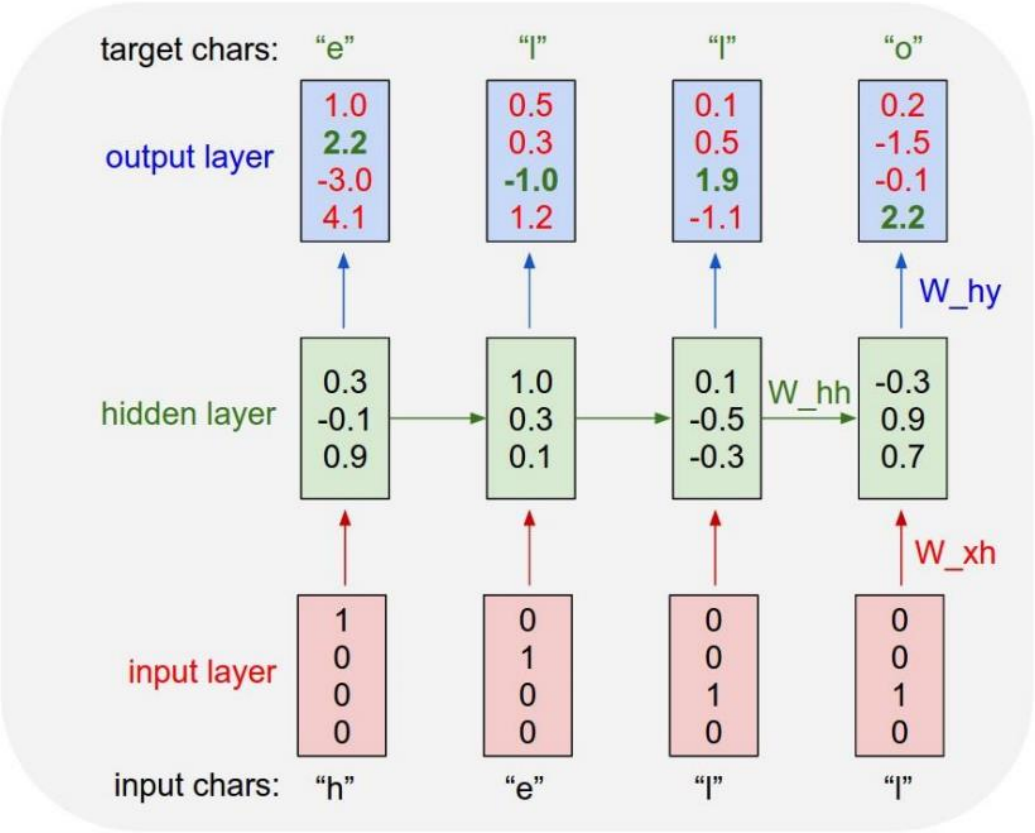
Neural Networks

$$h_t = \tanh(W_{hh}h_{t-1} + W_{xh}x_t)$$

Example:
Character-level
Language Model

Vocabulary:
[h,e,l,o]

Example training sequence:
"hello"

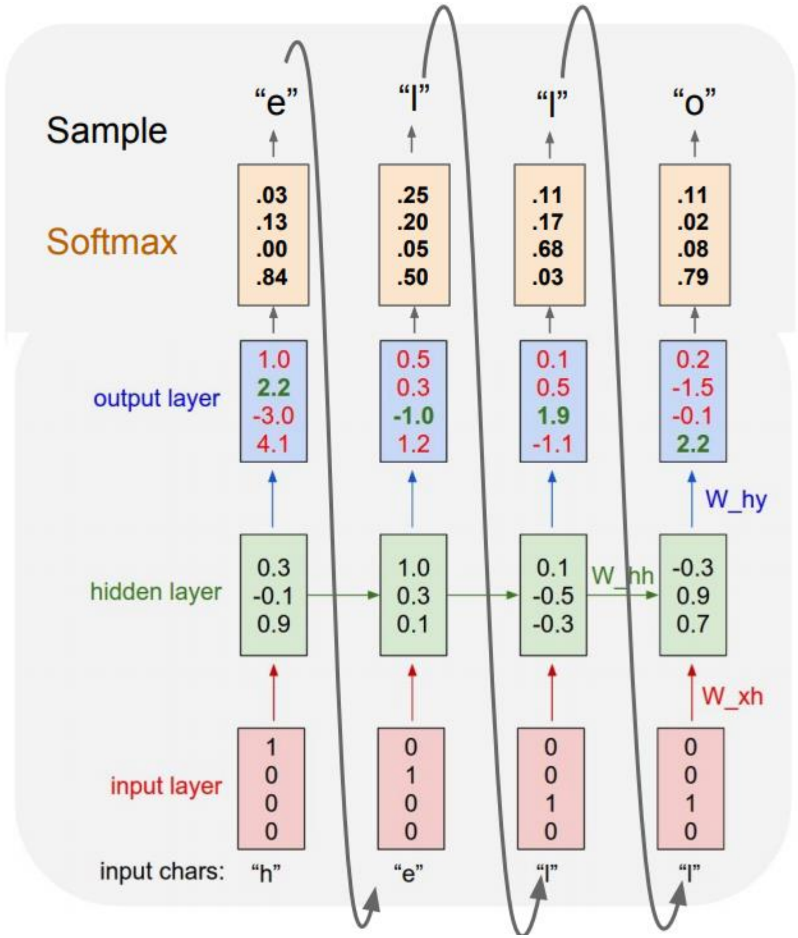


Neural Networks

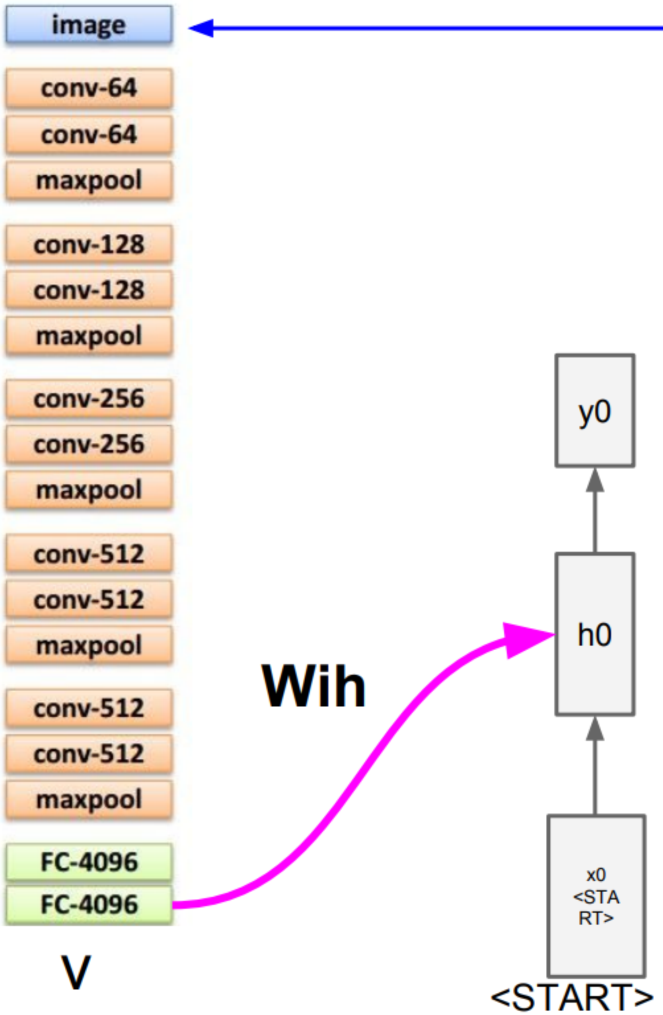
Example:
Character-level
Language Model
Sampling

Vocabulary:
[h,e,l,o]

At test-time sample characters one at a time, feed back to model



Neural Networks



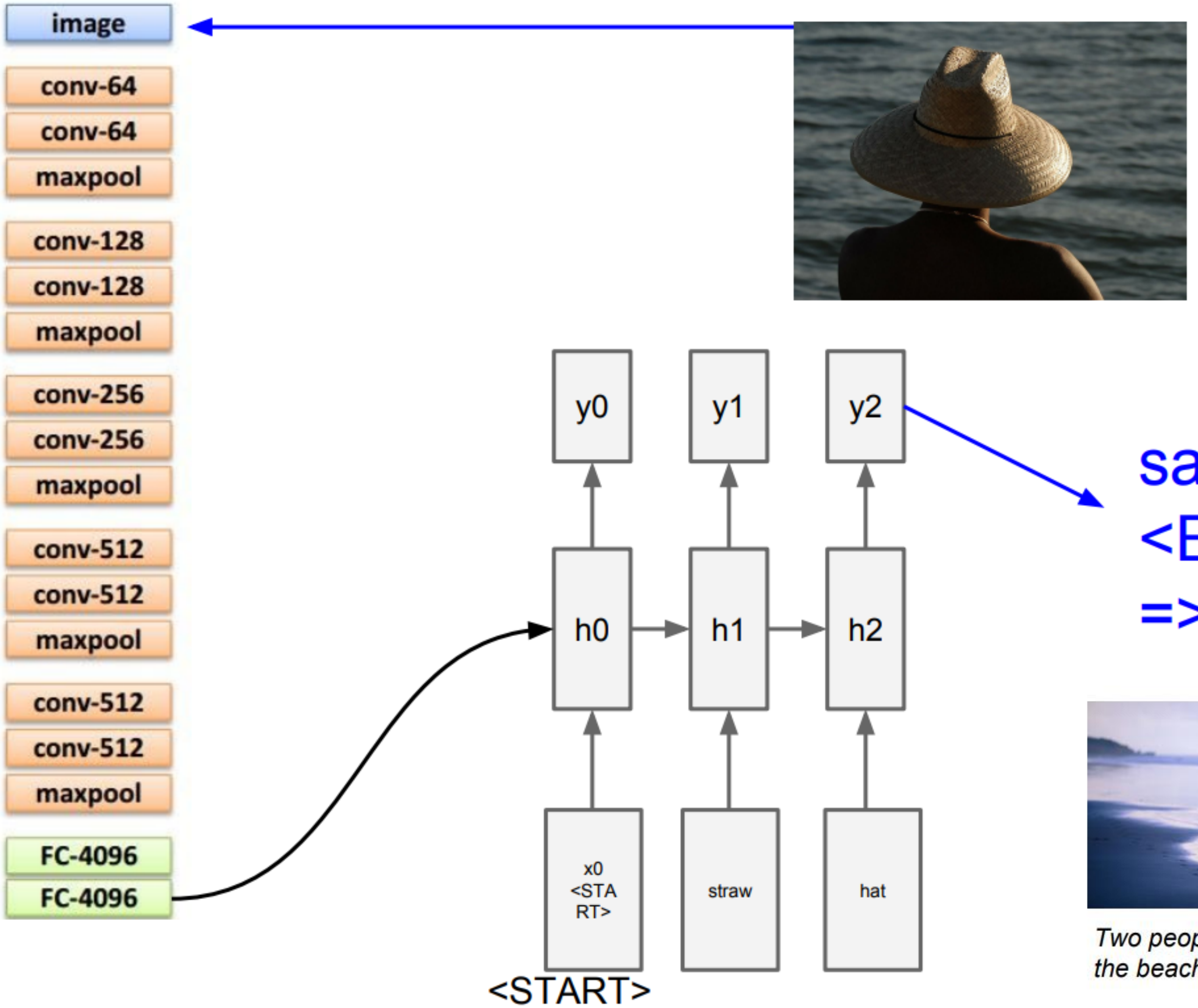
before:

h = tanh(Wxh * x + Whh * h)

now:

h = tanh(Wxh * x + Whh * h + **Wih * v**)

Neural Networks



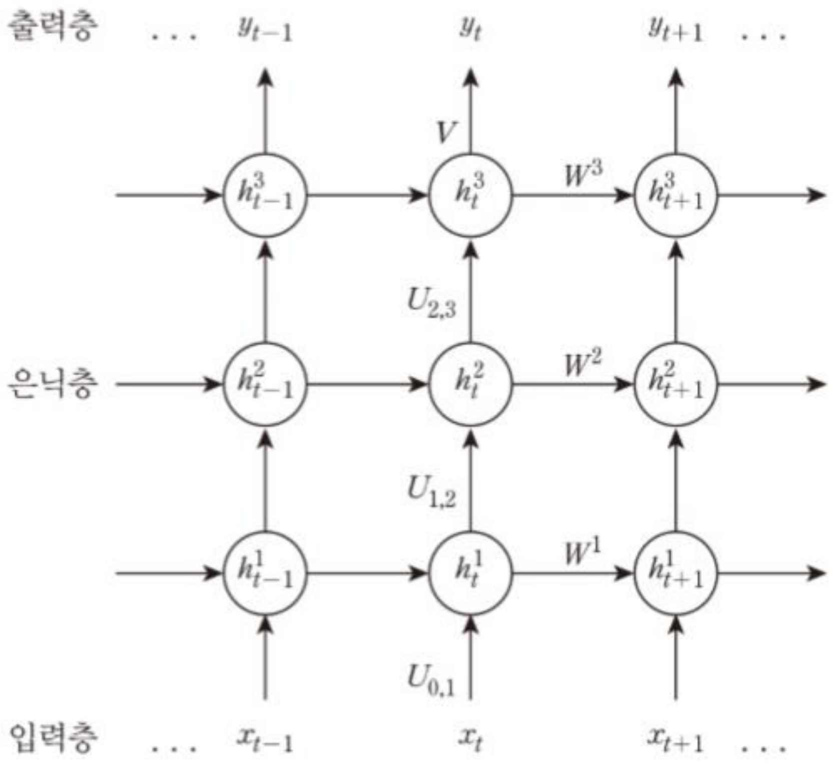
Two people walking on the beach with surfboards

A tennis player in action on the court

Neural Networks

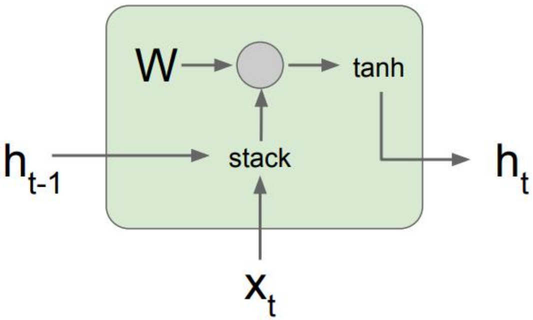
❖ 딥러닝 재귀 신경망(Deep RNN)

- 여러 개의 재귀 신경망을 쌓아서 아래층의 출력을 바로 위층의 입력으로 받아들이도록 만든 모델



Neural Networks

Vanilla RNN Gradient Flow



$$
\begin{aligned}
h_t &= \tanh(W_{hh}h_{t-1} + W_{xh}x_t) \\
&= \tanh\left(\begin{pmatrix} W_{hh} & W_{hx} \end{pmatrix} \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix} \right) \\
&= \tanh\left(W \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix} \right)
\end{aligned}
$$

Long Short Term Memory (LSTM)

Vanilla RNN

$$
\boxed{h_t = \tanh\left(W \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix} \right)}
$$

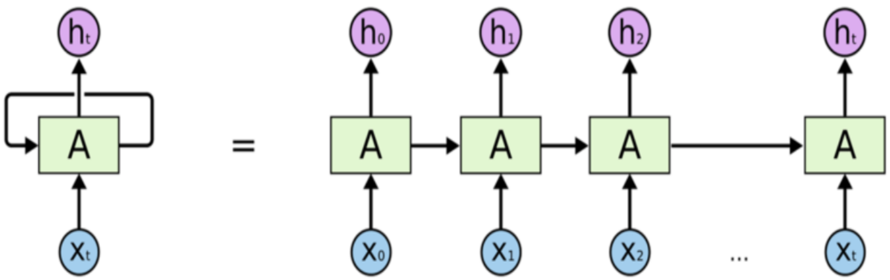
LSTM

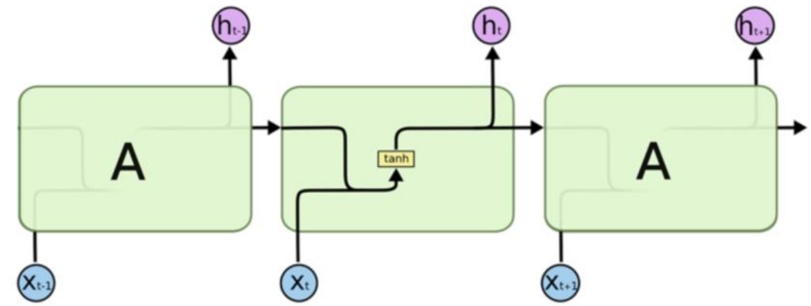
$$
\begin{pmatrix} i \\ f \\ o \\ g \end{pmatrix} = \begin{pmatrix} \sigma \\ \sigma \\ \sigma \\ \tanh \end{pmatrix} W \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix}
$$

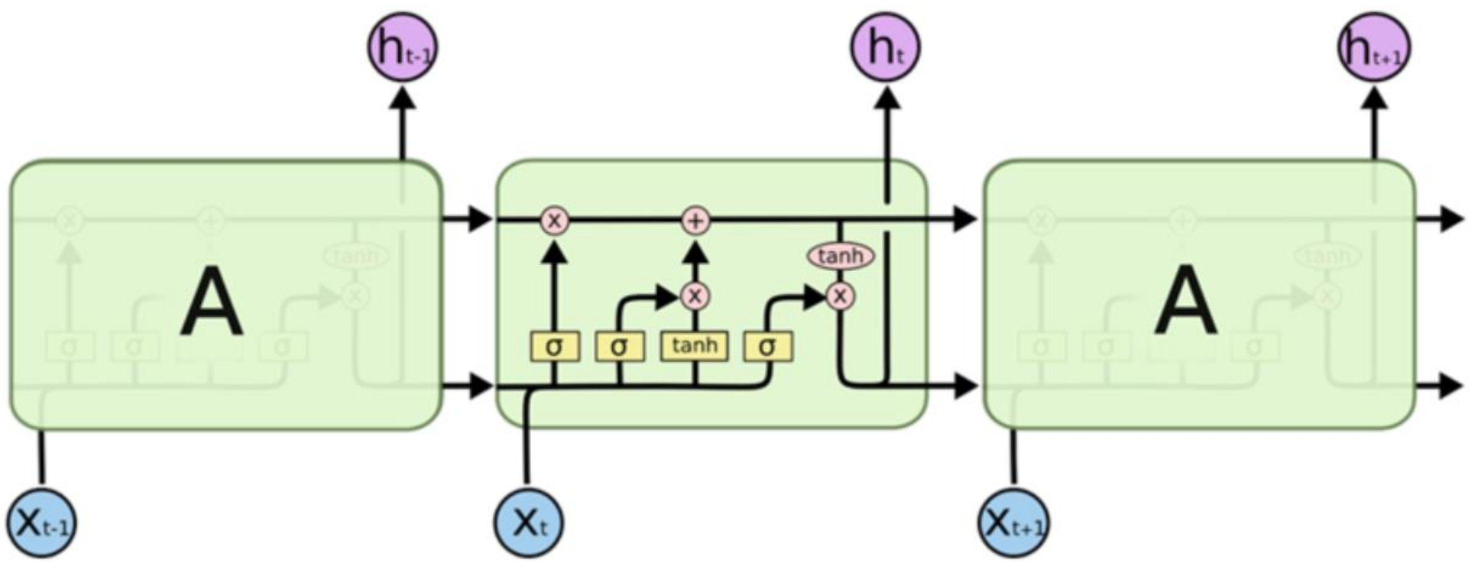
$$
c_t = f \odot c_{t-1} + i \odot g
$$

$$
h_t = o \odot \tanh(c_t)
$$

Neural Networks

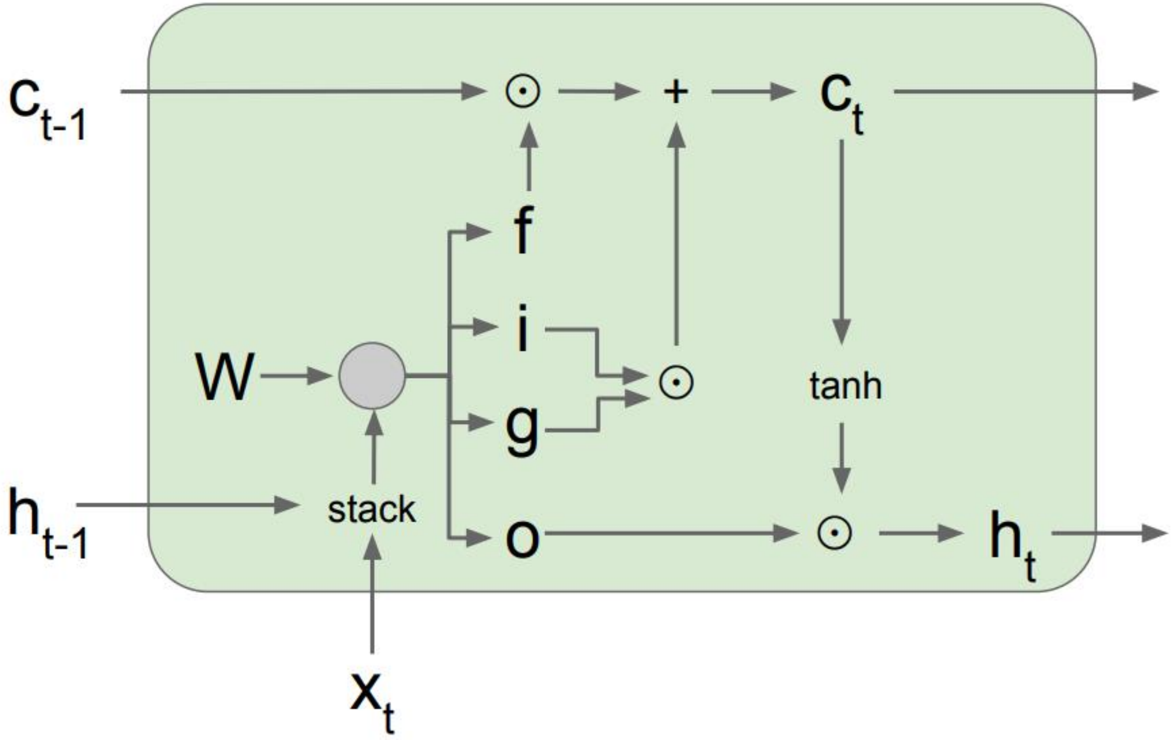






Neural Networks

Long Short Term Memory (LSTM)
[Hochreiter et al., 1997]



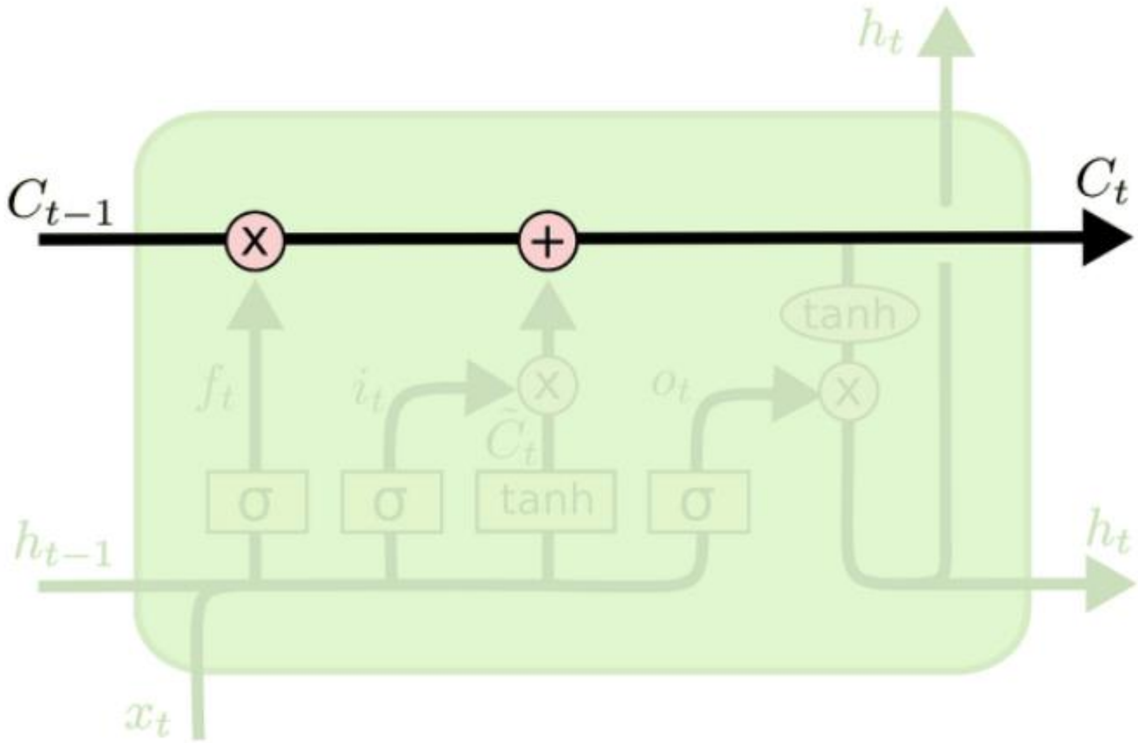
$$\begin{pmatrix} i \\ f \\ o \\ g \end{pmatrix} = \begin{pmatrix} \sigma \\ \sigma \\ \sigma \\ \tanh \end{pmatrix} W \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix}$$

$$c_t = f \odot c_{t-1} + i \odot g$$

$$h_t = o \odot \tanh(c_t)$$

Neural Networks

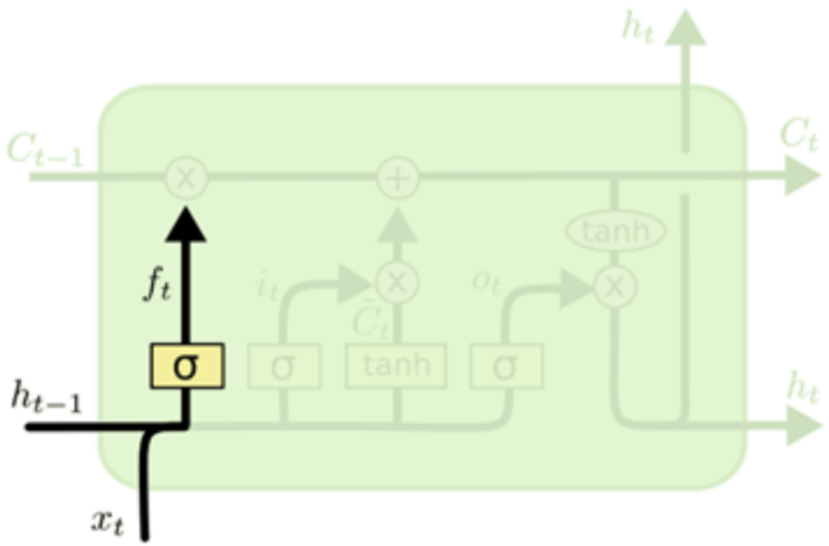
Cell state



forget gate layer / input gate layer / tanh layer

Neural Networks

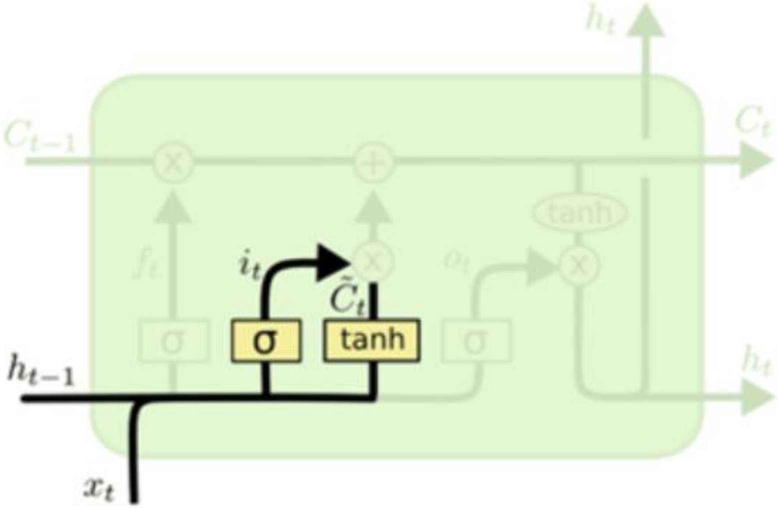
Forget gate layer



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

Neural Networks

Input gate layer / tanh layer

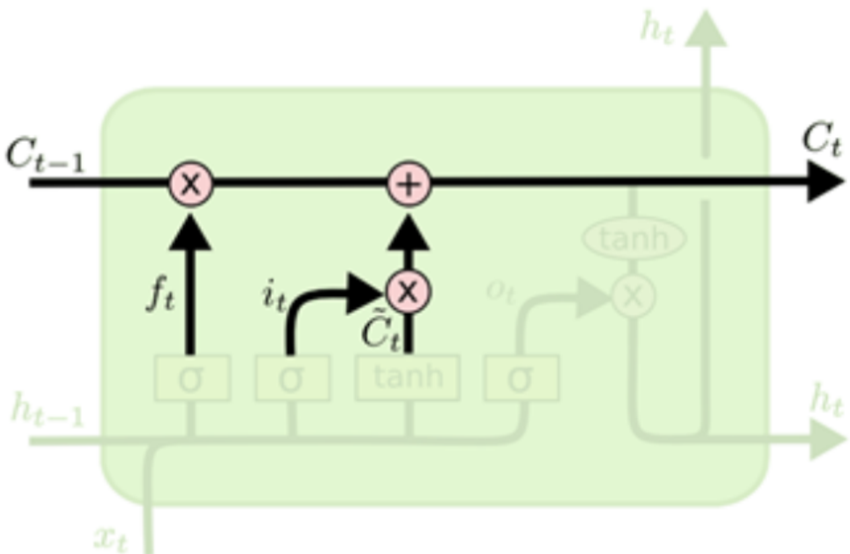


$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \ + \ b_i\right)$$

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

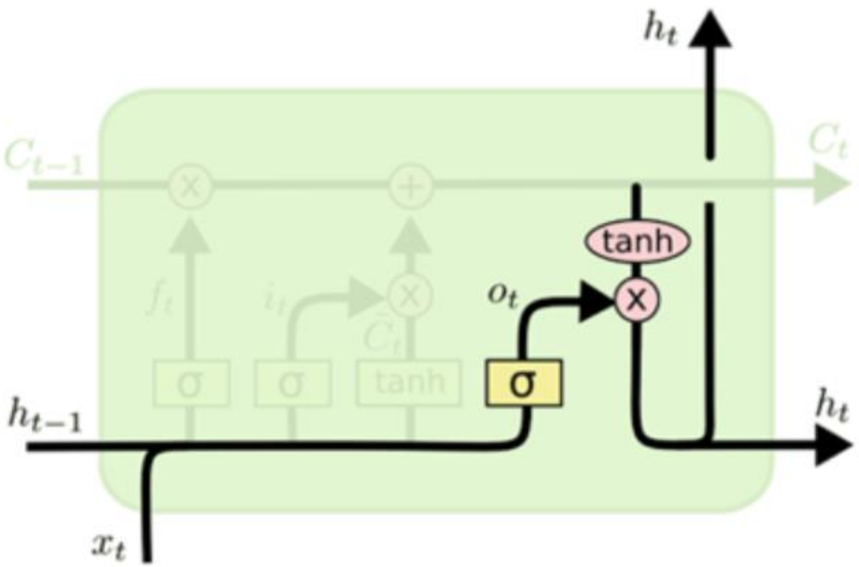
Neural Networks

Update cell state



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

Neural Networks



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

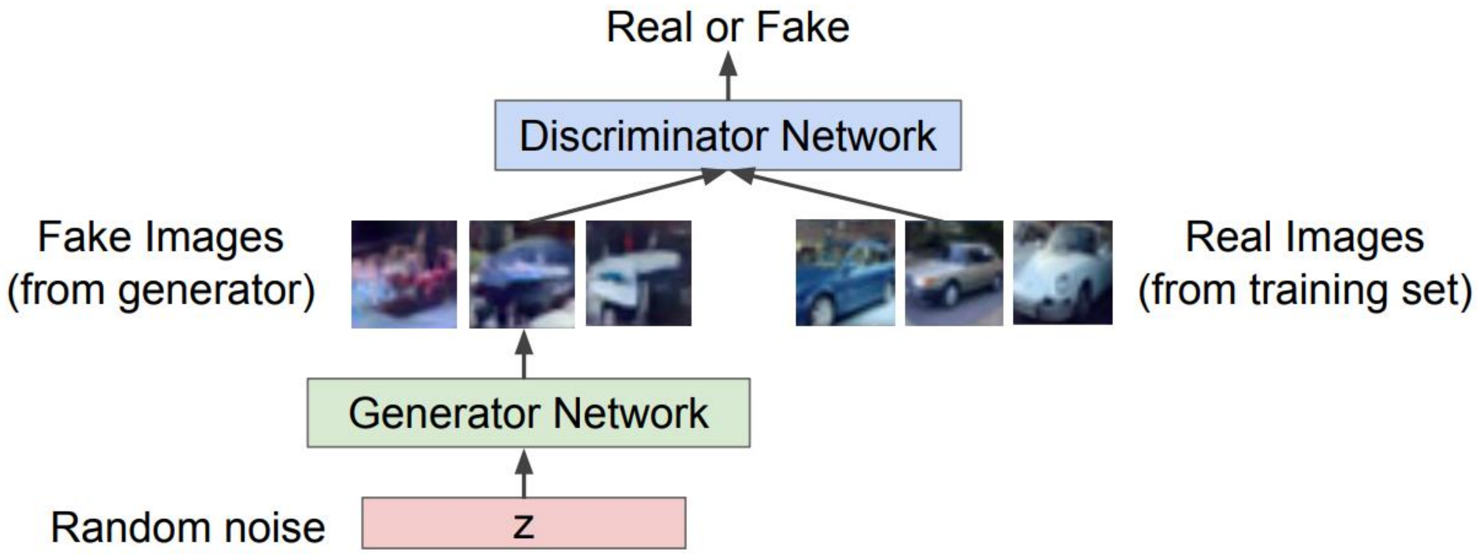
$$h_t = o_t * \tanh\left(C_t\right)$$

Neural Networks

Training GANs: Two-player game

Ian Goodfellow et al., "Generative Adversarial Nets", NIPS 2014

Generator network: try to fool the discriminator by generating real-looking images
Discriminator network: try to distinguish between real and fake images



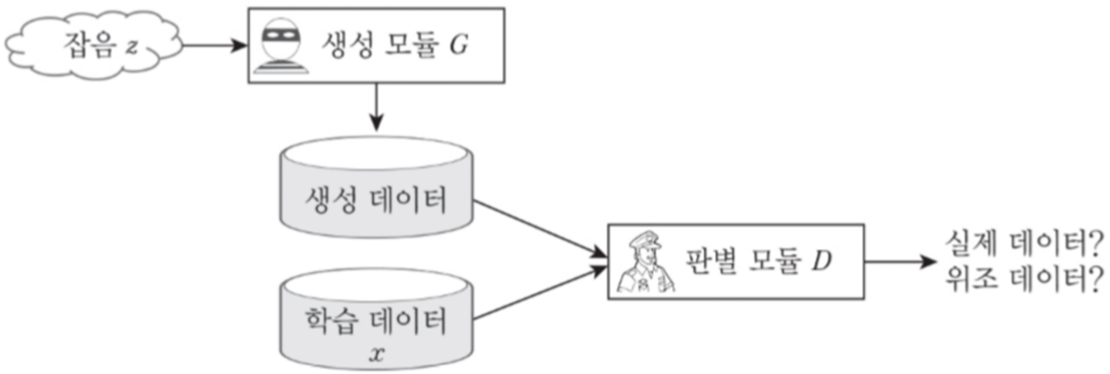
Neural Networks

❖ **생성 모듈** G

- 데이터를 생성하는 모듈
- 위조범 역할 : 잡음 z를 사용하여 데이터 $G(z)$ 생성

❖ **판별 모듈** D

- 생성된 데이터가 학습데이터의 특성을 갖는 평가하여 판단하는 모듈
- 경찰관 역할 : $D(x)$ 계산
  - 학습 데이터 x에 대한 큰 값: $D(x)$가 큰 값
  - 생성된 데이터 $G(z)$에 대해서 작은 값
    - » $D(G(z))$의 값은 작음



Neural Networks

- ❖ **생성 모듈과 판별 모듈의 동시 학습**
  - **생성 모듈 G**
    - 판별 모듈을 속이도록 학습
    - 생성 모듈이 생성한 데이터 $G(z)$에 대해 판별 모듈이 큰 값을 주도록 학습
      - 잡음 $z \sim P_z(z)$으로 부터 생성한 데이터 $G(z)$에 대해 $D(G(z))$이 커지도록 학습
  - **판별 모듈 D**
    - 생성된 데이터(가짜 데이터)를 잘 식별하도록 학습
    - 학습 데이터 $x \sim P_{data}$에 판별 모듈의 출력값 $D(x)$는 크고,
      생성된 데이터 $G(z)$에 대한 $1 - D(G(z))$의 값은 커지도록 학습

  - **두 명이 하는 게임(two-player game)과 유사**

  - **최대화시킬 목적 함수**

$$L(D,G) = E_{\boldsymbol{x} \sim p_{data}(\boldsymbol{x})}[\log D(x)] + E_{\boldsymbol{z} \sim p_z(\boldsymbol{z})}[\log(1 - D(G(z))]$$

Training GANs: Two-player game

Ian Goodfellow et al., "Generative Adversarial Nets", NIPS 2014

Generator network: try to fool the discriminator by generating real-looking images
Discriminator network: try to distinguish between real and fake images

Train jointly in **minimax game**

Discriminator outputs likelihood in (0,1) of real image

Minimax objective function:

$$\min_{\theta_g} \max_{\theta_d} \left[\mathbb{E}_{x \sim p_{data}} \log \underbrace{D_{\theta_d}(x)}_{\text{Discriminator output for real data x}} + \mathbb{E}_{z \sim p(z)} \log(1 - \underbrace{D_{\theta_d}(G_{\theta_g}(z))}_{\text{Discriminator output for generated fake data G(z)}}) \right]$$

- Discriminator (θ_d) wants to **maximize objective** such that D(x) is close to 1 (real) and D(G(z)) is close to 0 (fake)
- Generator (θ_g) wants to **minimize objective** such that D(G(z)) is close to 1 (discriminator is fooled into thinking generated G(z) is real)

Neural Networks

Generative Adversarial Nets: Interpretable Vector Math

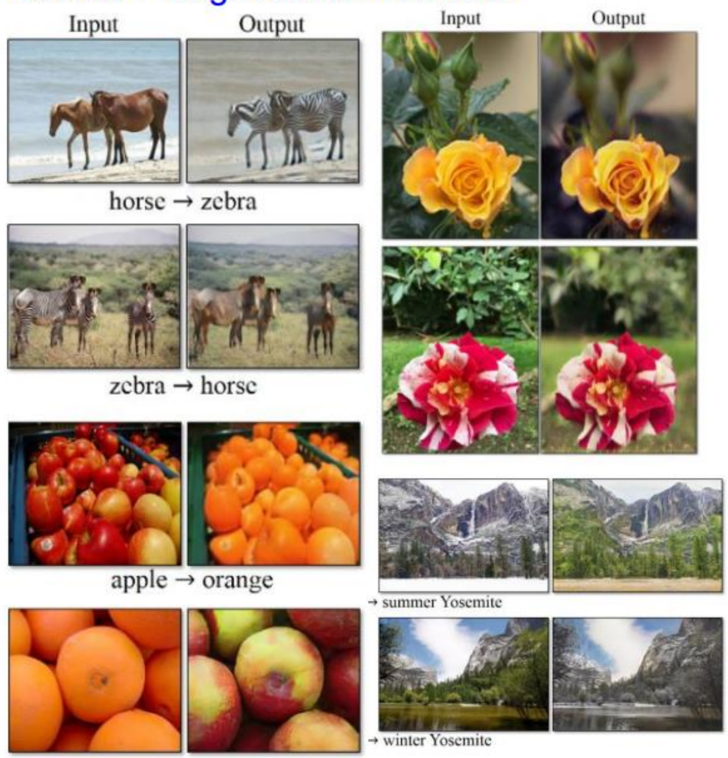
Neural Networks

Better training and generation

LSGAN. Mao et al. 2017.

BEGAN. Bertholet et al. 2017.

Source->Target domain transfer



CycleGAN. Zhu et al. 2017.

Text -> Image Synthesis

Reed et al. 2017.

Many GAN applications

Pix2pix. Isola 2017. Many examples at https://phillipi.github.io/pix2pix/

Neural Networks