아티스트를 위한 머신러닝 & 딥러닝

텐서플로를 활용한 딥러닝 #6

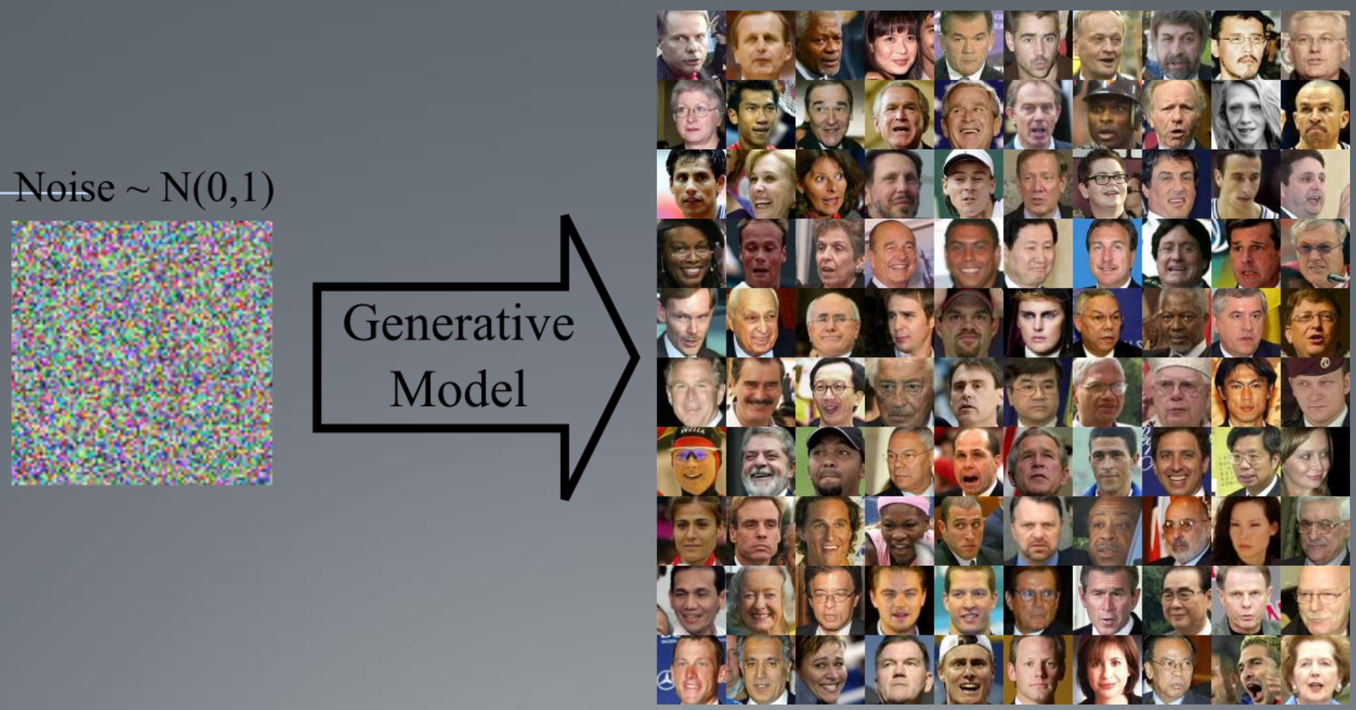
서울대학교 & V.DO / 김대식

art center nabi v.do

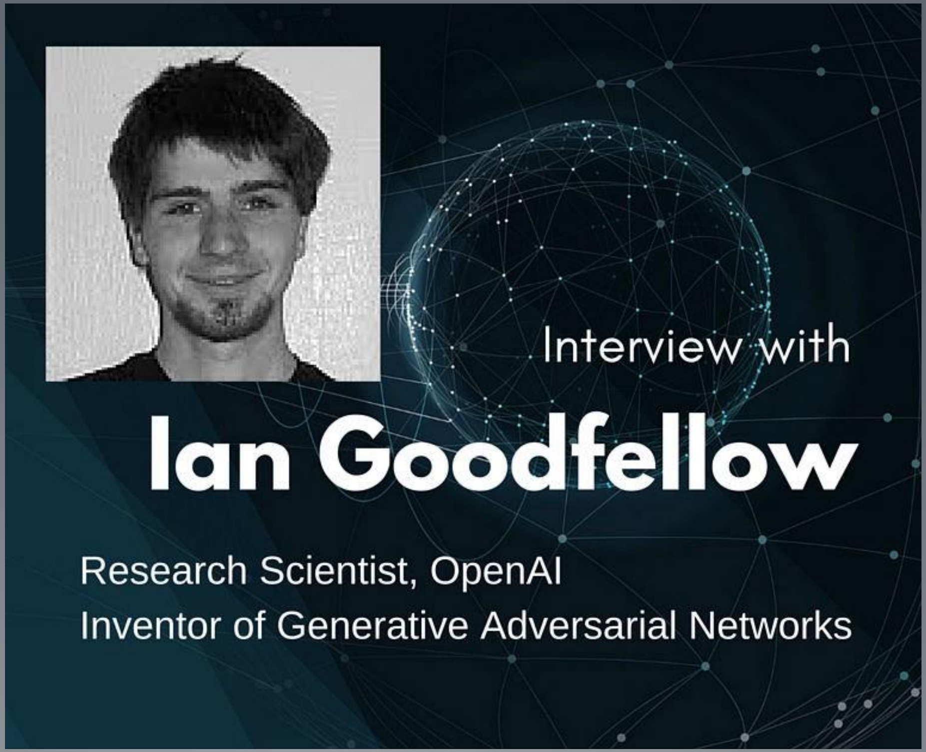
Recap

art center nabi v.do

Generative Adversarial network(GAN)

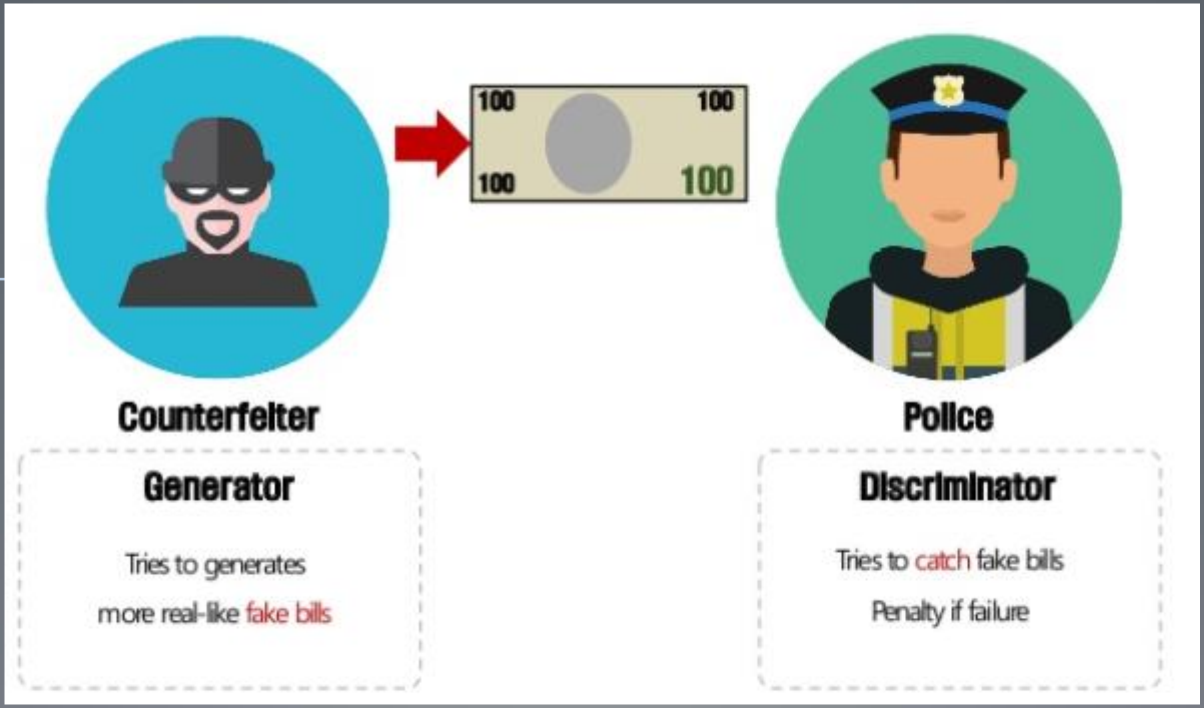


Generative Adversarial network(GAN)



art center nabi v.do

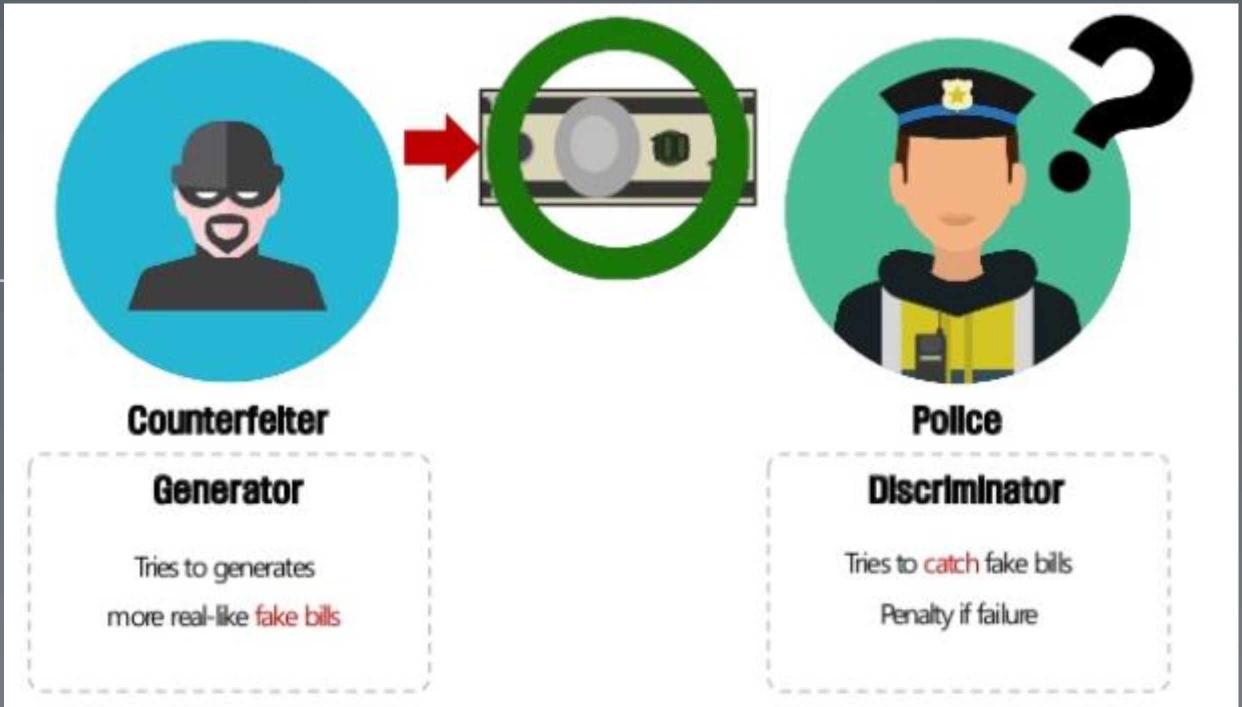
위조지폐범 vs 경찰


100
100
100
100
Counterfelter
Generator
Tries to generates
more real-like fake bills
Police
Discriminator
Tries to catch fake bills
Penalty if failure

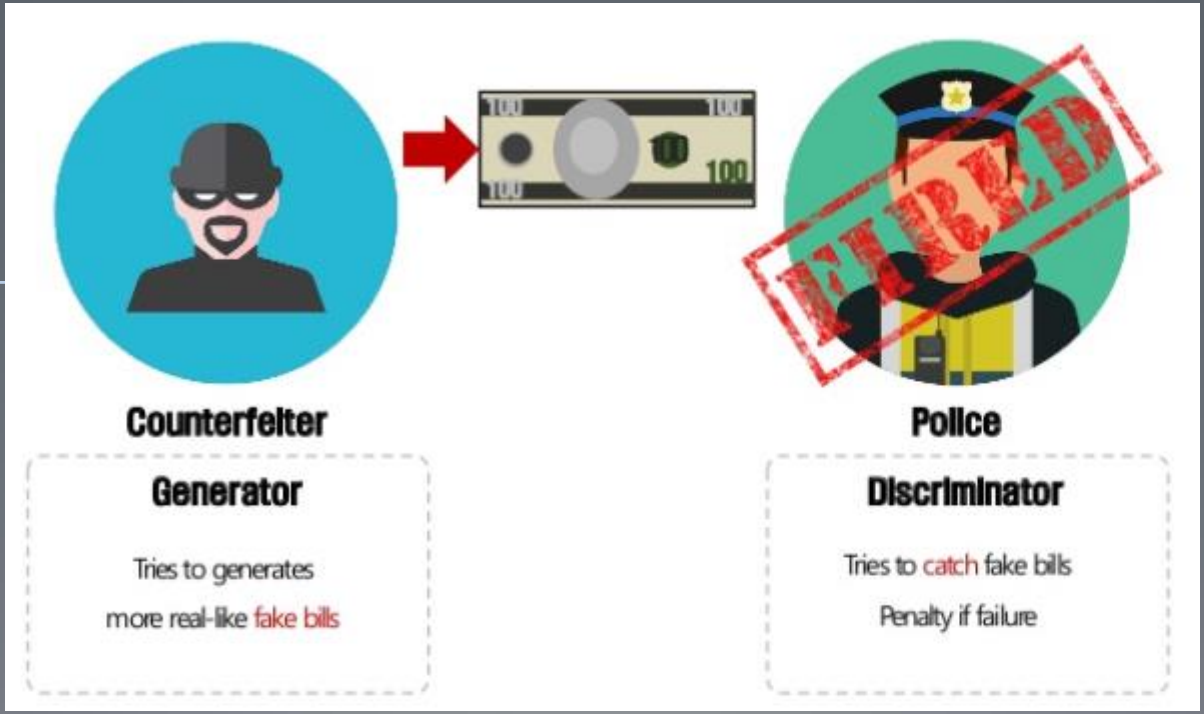
위조지폐범 vs 경찰


100
100
BUSTED!
100
100
Counterfelter
Generator
Tries to generates
more real-like fake bills
Police
Discriminator
Tries to catch fake bills
Penalty if failure

위조지폐범 vs 경찰

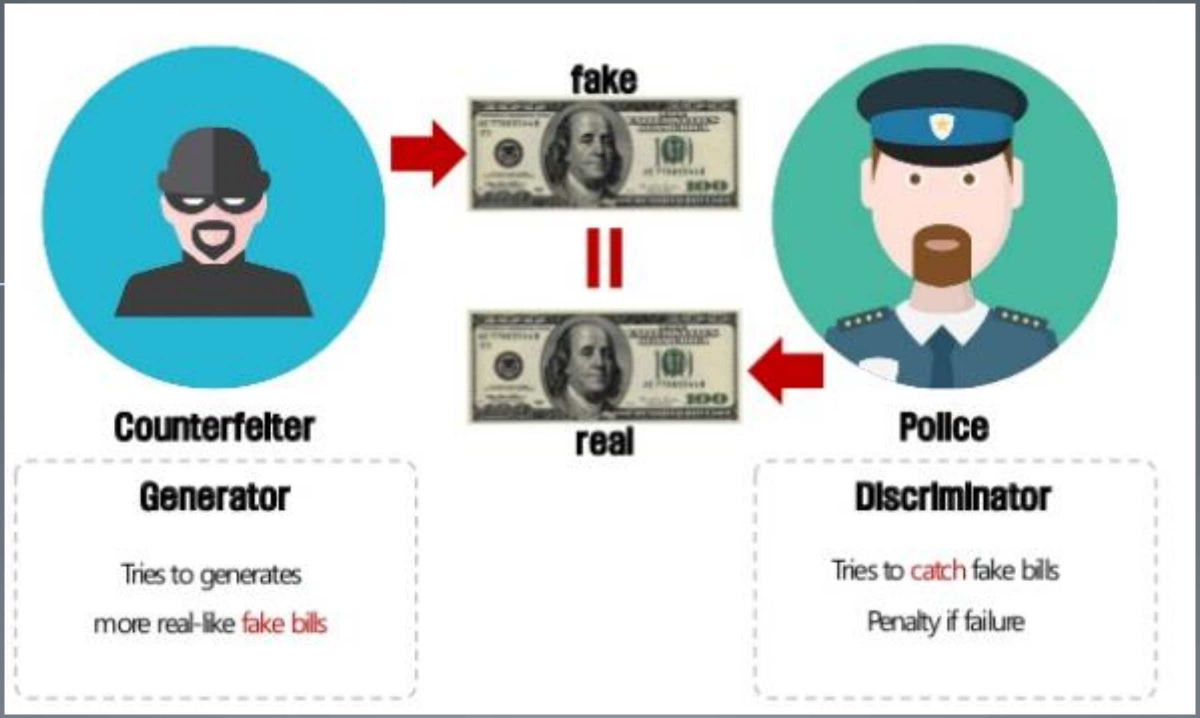

Counterfelter
Generator
Tries to generates
more real-like fake bills
Police
Discriminator
Tries to catch fake bills
Penalty if failure

위조지폐범 vs 경찰

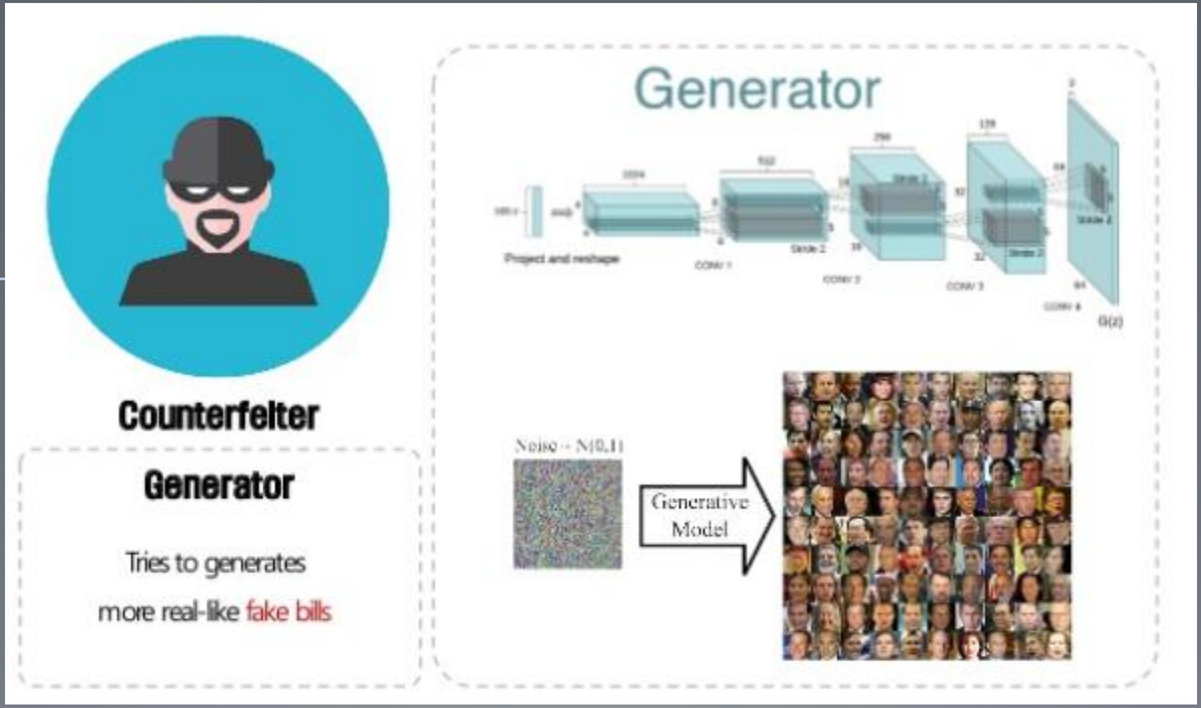


100
Counterfelter
Generator
Tries to generates
more real-like fake bills
FIRED
Police
Discriminator
Tries to catch fake bills
Penalty if failure

위조지폐범 vs 경찰

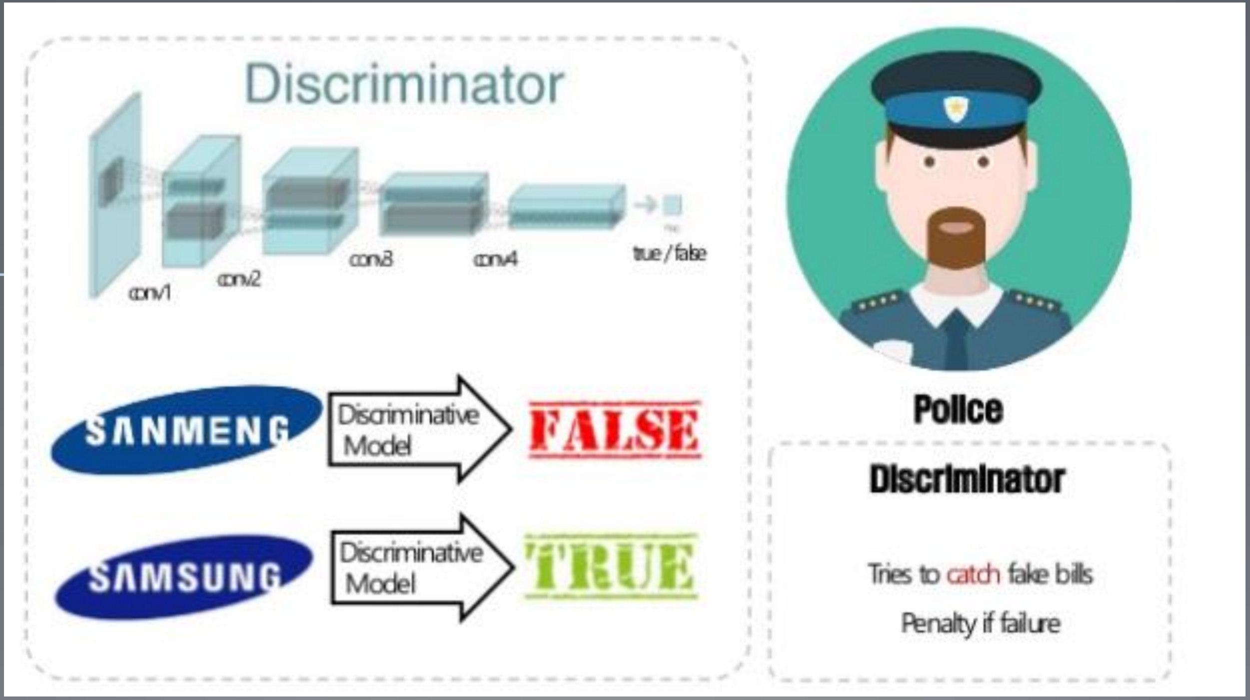

fake
real
Counterfeiter
Generator
Tries to generates
more real-like fake bills
Police
Discriminator
Tries to catch fake bills
Penalty if failure

위조지폐범 vs 경찰



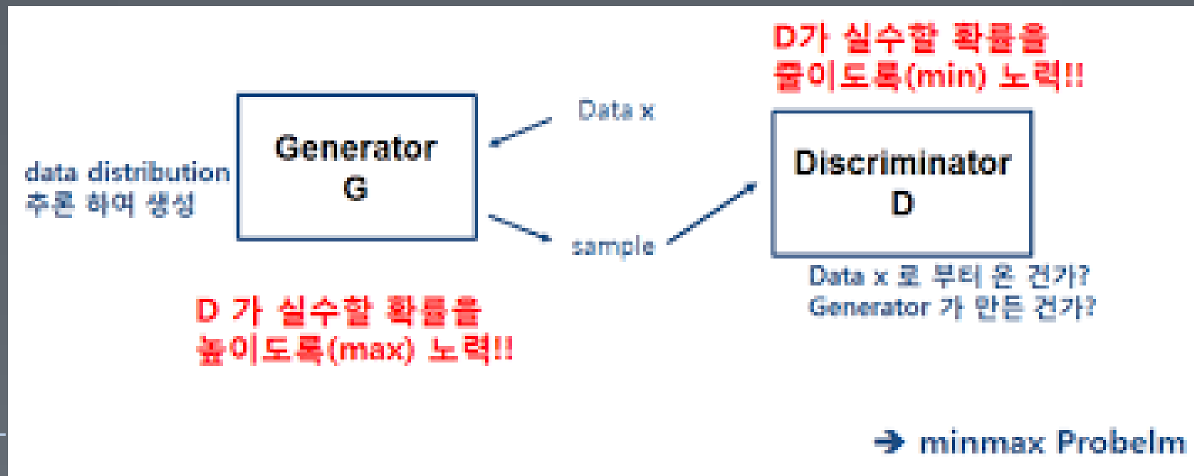
art center nabi v.do

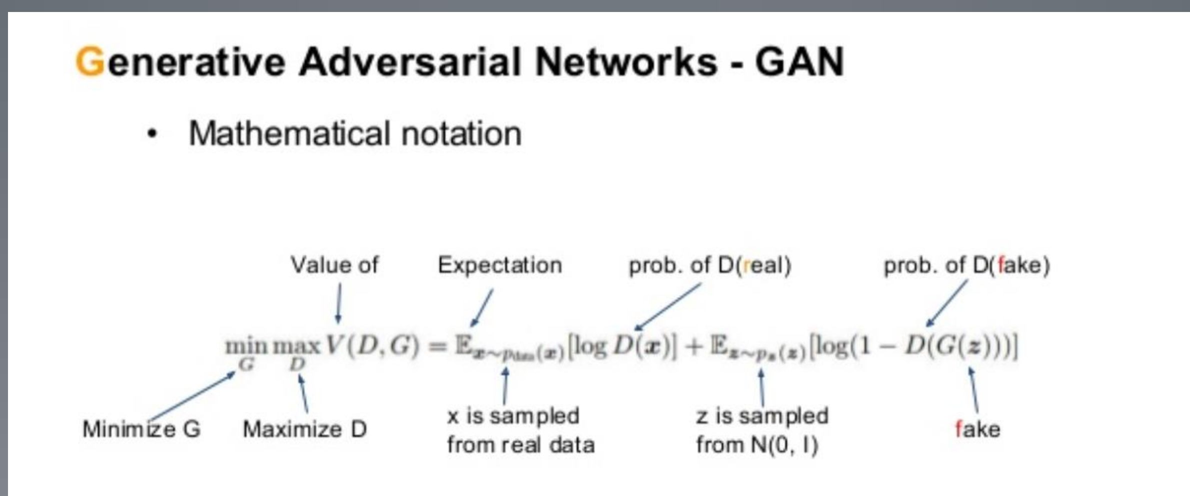
위조지폐범 vs 경찰



Discriminator
conv1
conv2
conv3
conv4
true / false
SANMENG
Discriminative Model
FALSE
SAMSUNG
Discriminative Model
TRUE
Police
Discriminator
Tries to catch fake bills
Penalty if failure

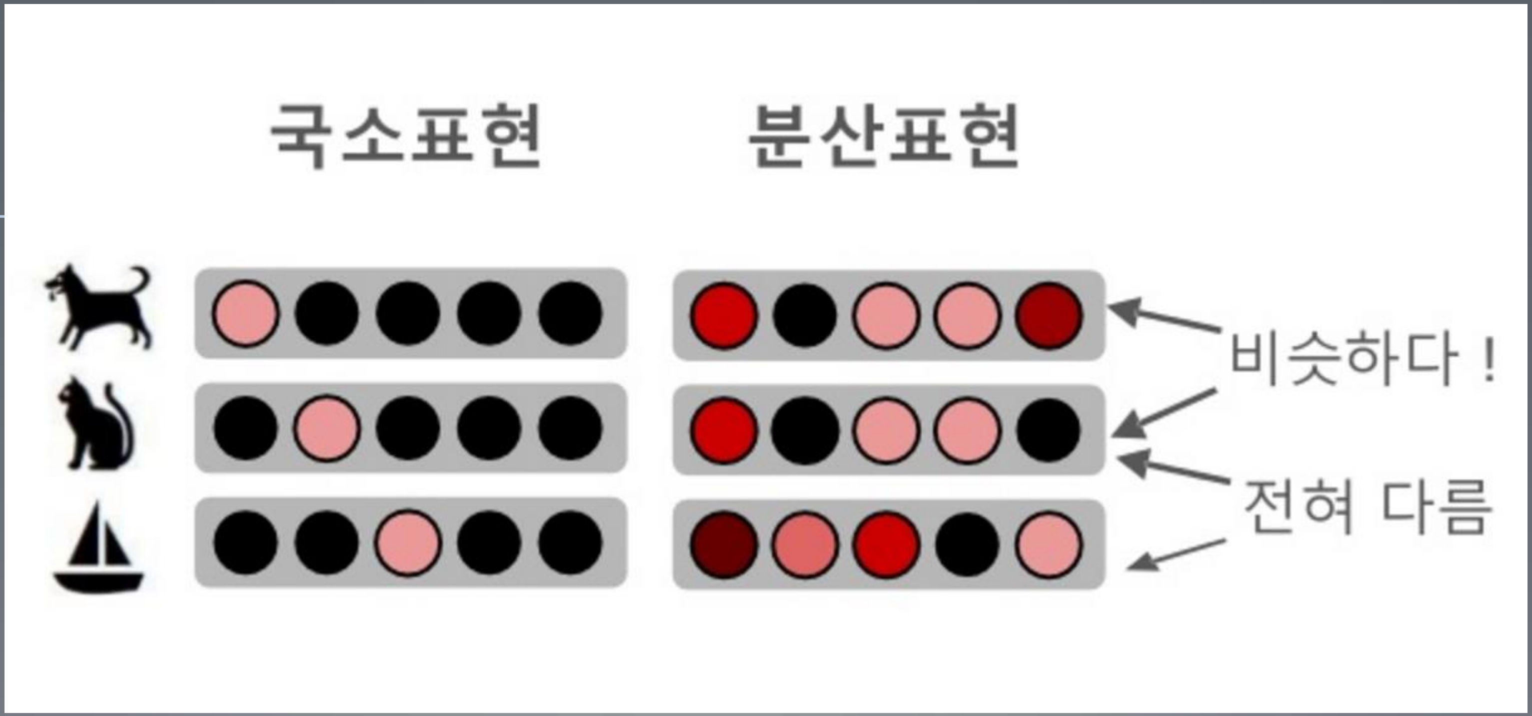
GAN 학습법


D가 실수할 확률을
줄이도록(min) 노력!!
Generator
G
Data x
data distribution
추론 하여 생성
sample
Discriminator
D
Data x 로 부터 온 건가?
Generator 가 만든 건가?
D 가 실수할 확률을
높이도록(max) 노력!!
➔ minmax Probelm


Generative Adversarial Networks - GAN
• Mathematical notation
Value of
Expectation
prob. of D(real)
prob. of D(fake)
$\min_G \max_D V(D,G) = \mathbb{E}_{x \sim p_{data}(x)}[\log D(x)] + \mathbb{E}_{z \sim p_z(z)}[\log(1 - D(G(z)))]$
Minimize G
Maximize D
x is sampled
from real data
z is sampled
from N(0, I)
fake

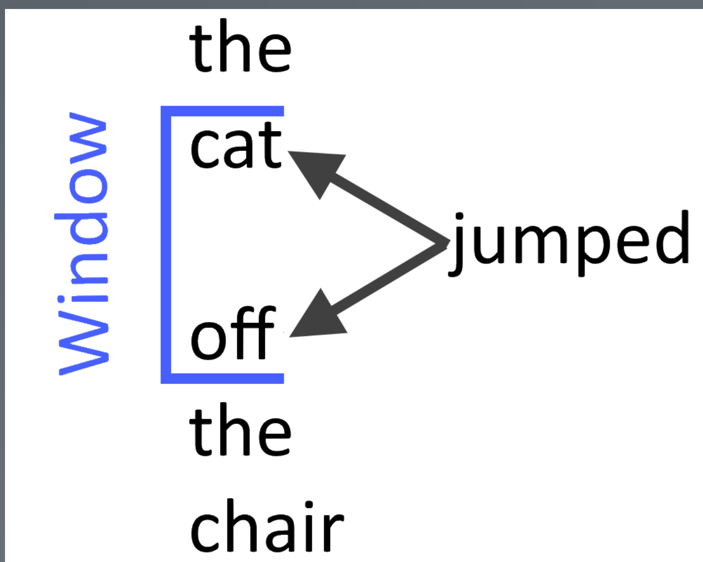
Word Vector

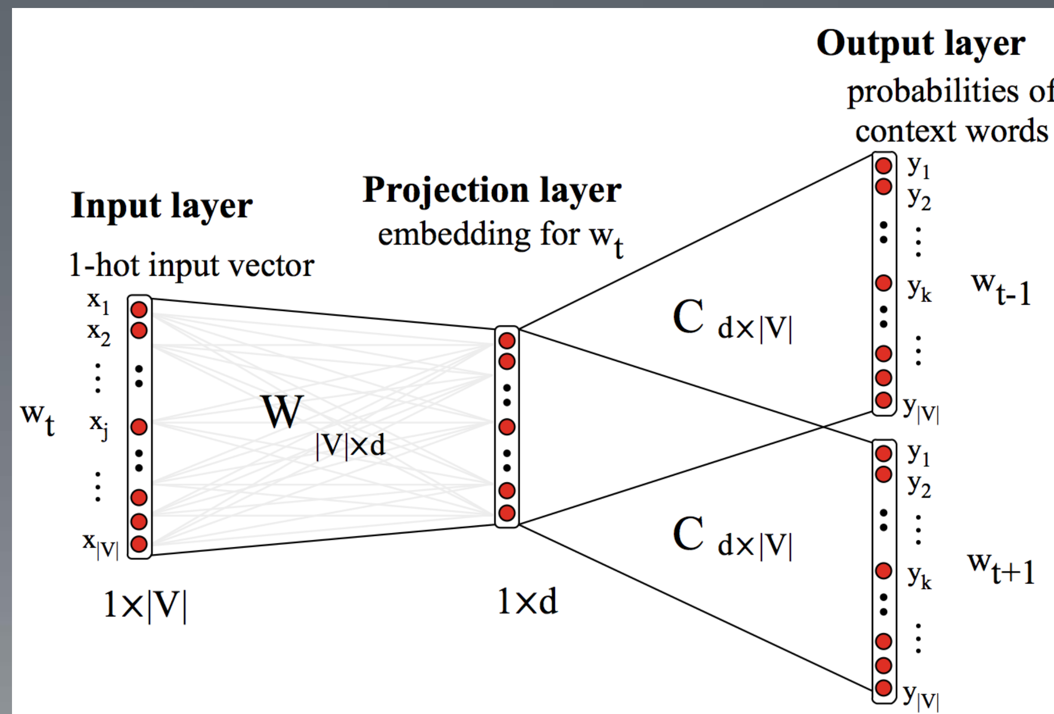
분산 표현 (Distributed Representation)



- CBOW
  - 문맥으로 부터 단어 예측
  - 소규모 데이터 셋에 성능 유리
- Skip-gram
  - 단어로부터 문맥 예측
  - 대규모 데이터셋에 유리

Skip-gram 구조


the
Window
cat
jumped
off
the
chair


Output layer
probabilities of context words
Input layer
1-hot input vector
Projection layer
embedding for w_t
w_t
x_1
x_2
x_j
$x_{|V|}$
W $|V| \times d$
C $d \times |V|$
C $d \times |V|$
y_1
y_2
y_k
$y_{|V|}$
w_{t-1}
w_{t+1}
$1 \times |V|$
$1 \times d$

Word Embedding

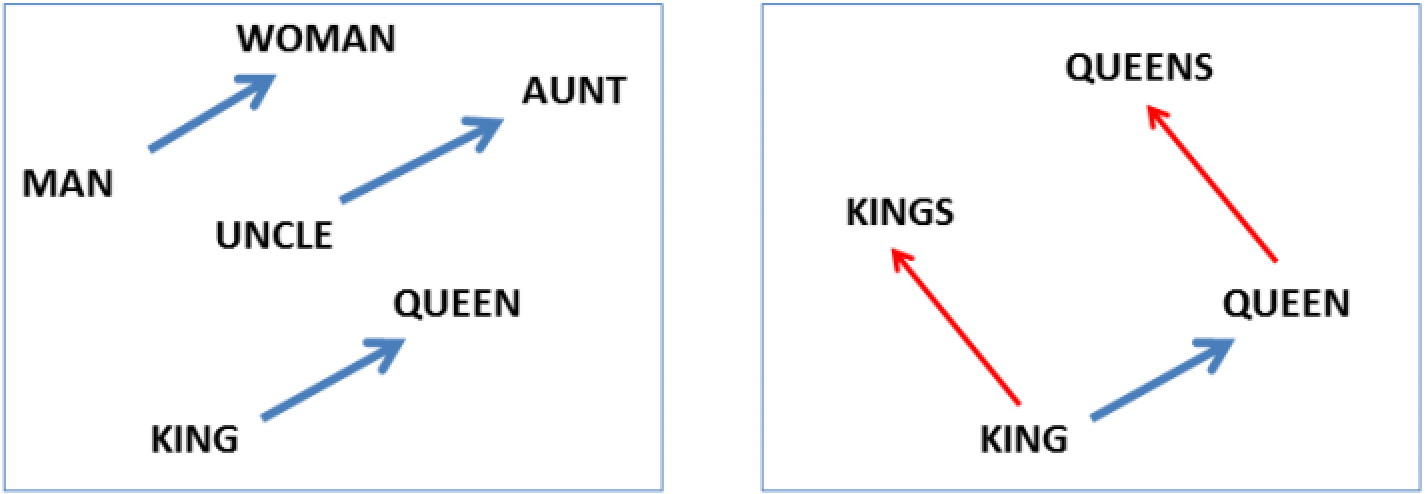
1-hot input

W

Distributed representation

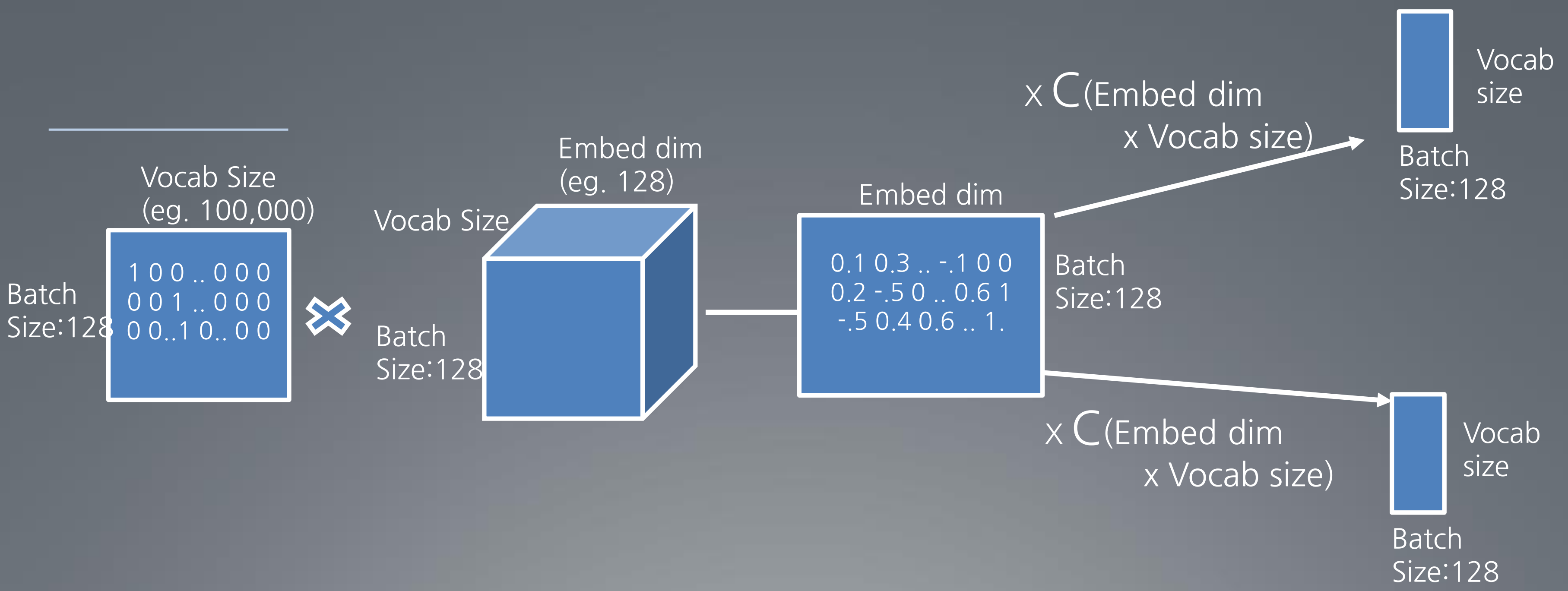
$$[0 \quad 0 \quad 0 \quad \boxed{1} \quad 0] \times \begin{bmatrix} 17 & 24 & 1 \\ 23 & 5 & 7 \\ 4 & 6 & 13 \\ 10 & 12 & 19 \\ 11 & 18 & 25 \end{bmatrix} = [10 \quad 12 \quad 19]$$

Results



(Mikolov et al., NAACL HLT, 2013)

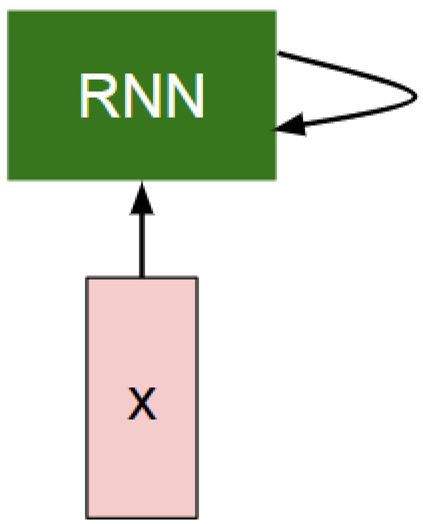
실습 : skip-gram 예제



RNN
(Recurrent Neural Network)

art center nabi v.do

Recurrent Neural Network



Recurrent Neural Network

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

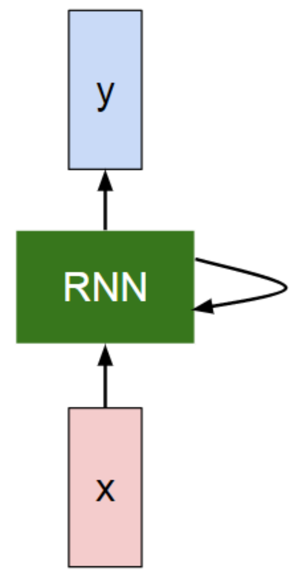
$$\boxed{h_t} = \boxed{f_W}(\boxed{h_{t-1}}, \boxed{x_t})$$

new state

old state

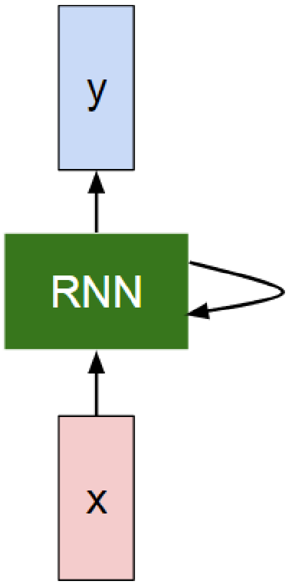
input vector at some time step

some function with parameters W



(Vanilla) Recurrent Neural Network

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

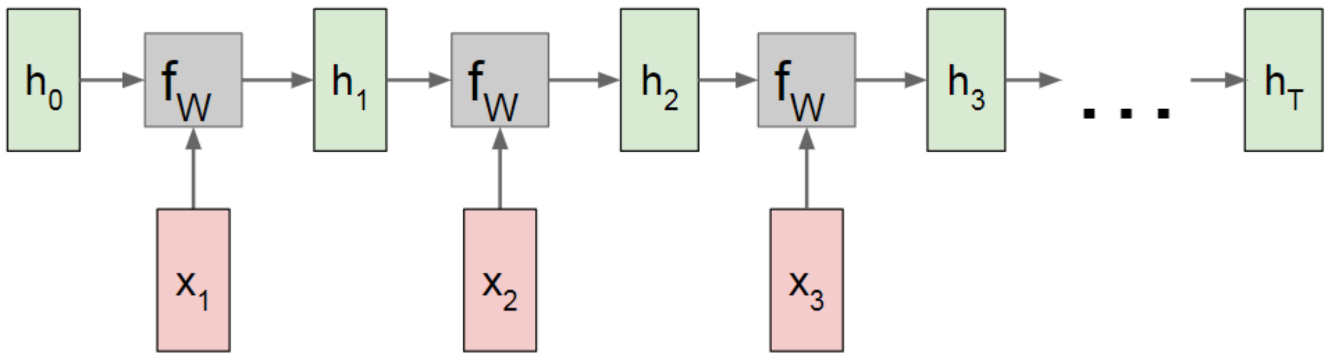


$$h_t = f_W(h_{t-1}, x_t)$$

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

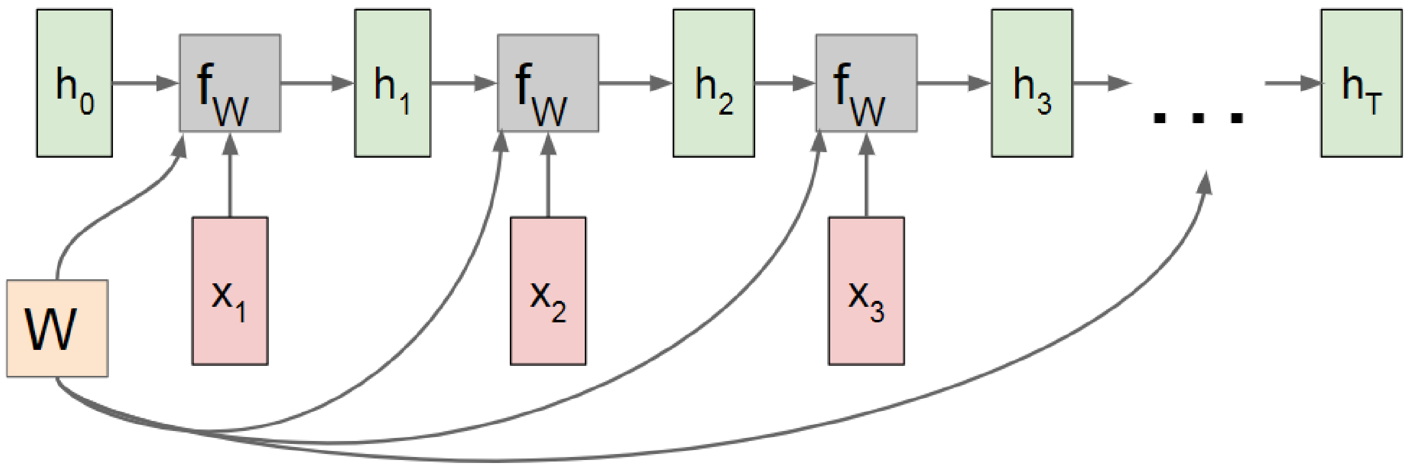
$$y_t = W_{hy}h_t$$

RNN: Computational Graph

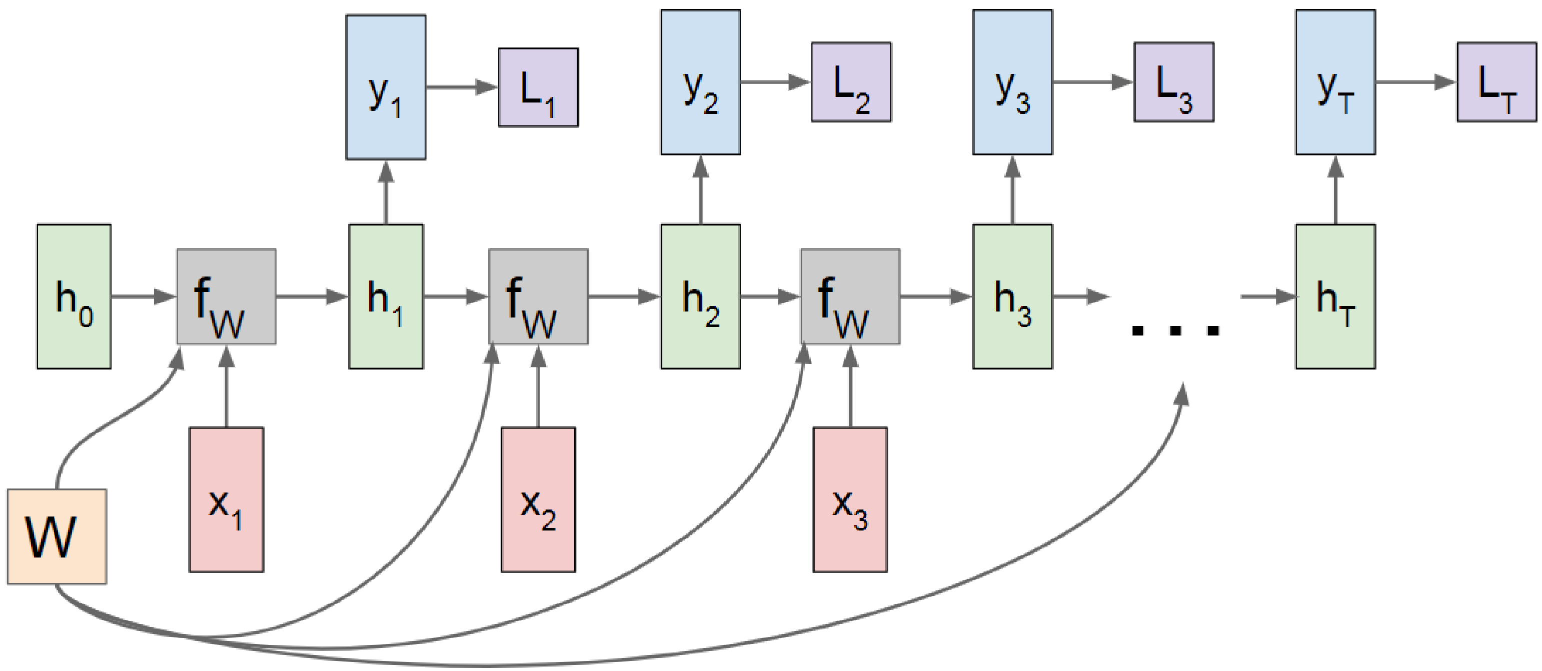


RNN: Computational Graph

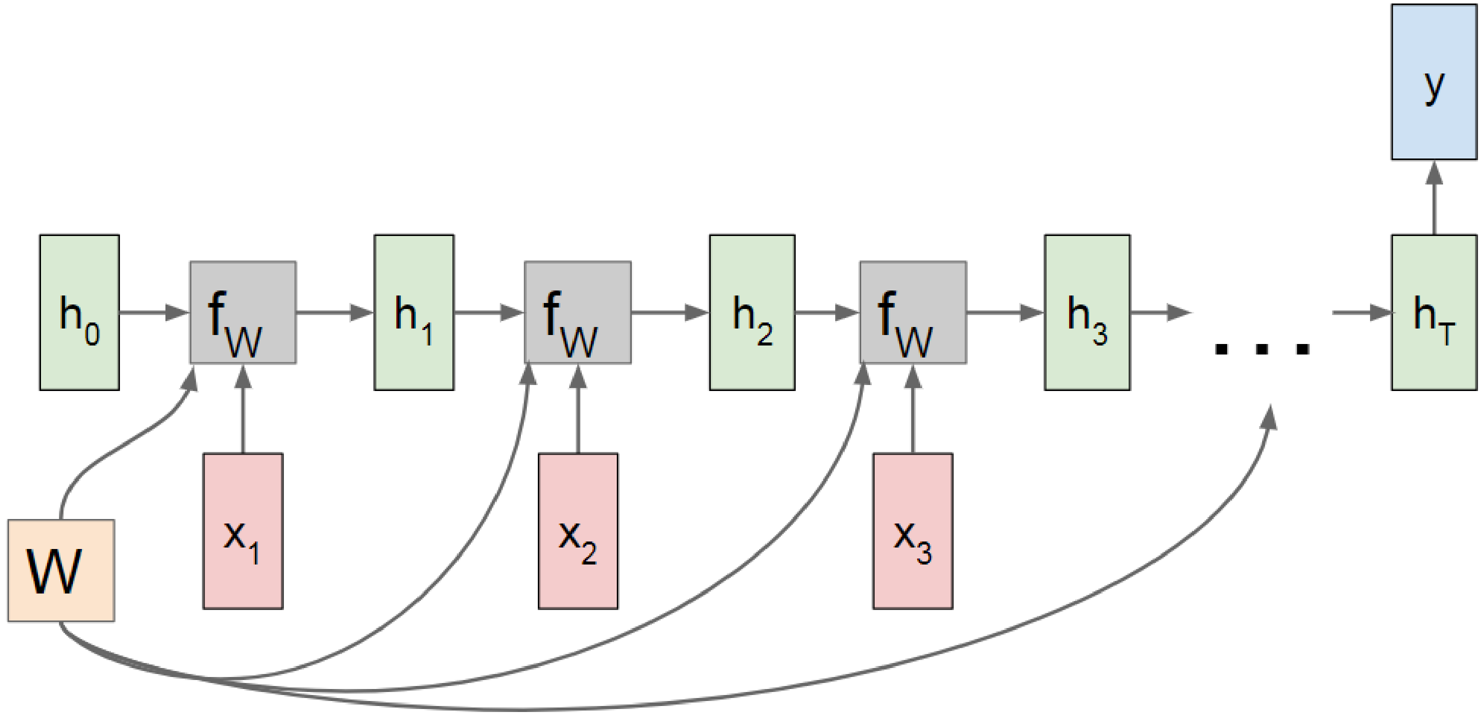
Re-use the same weight matrix at every time-step



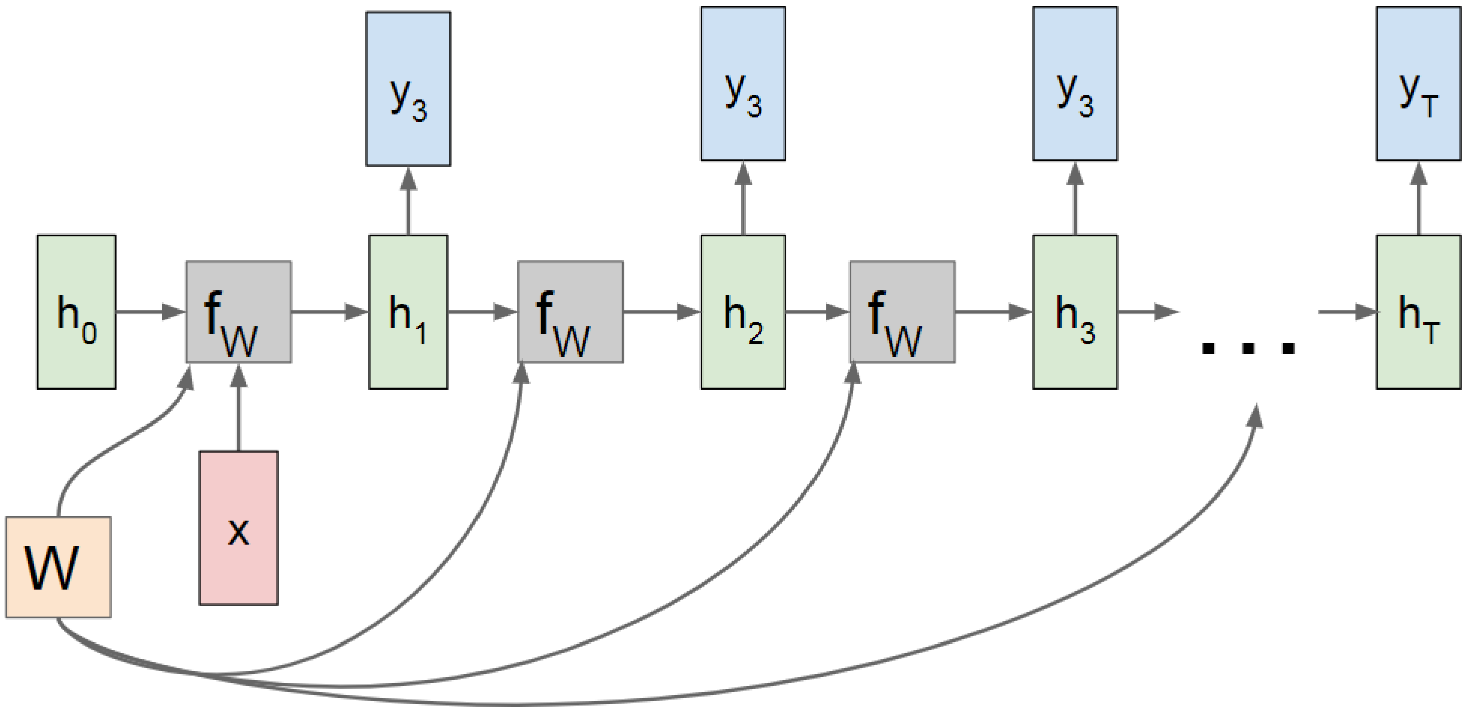
RNN: Computational Graph: Many to Many



RNN: Computational Graph: Many to One



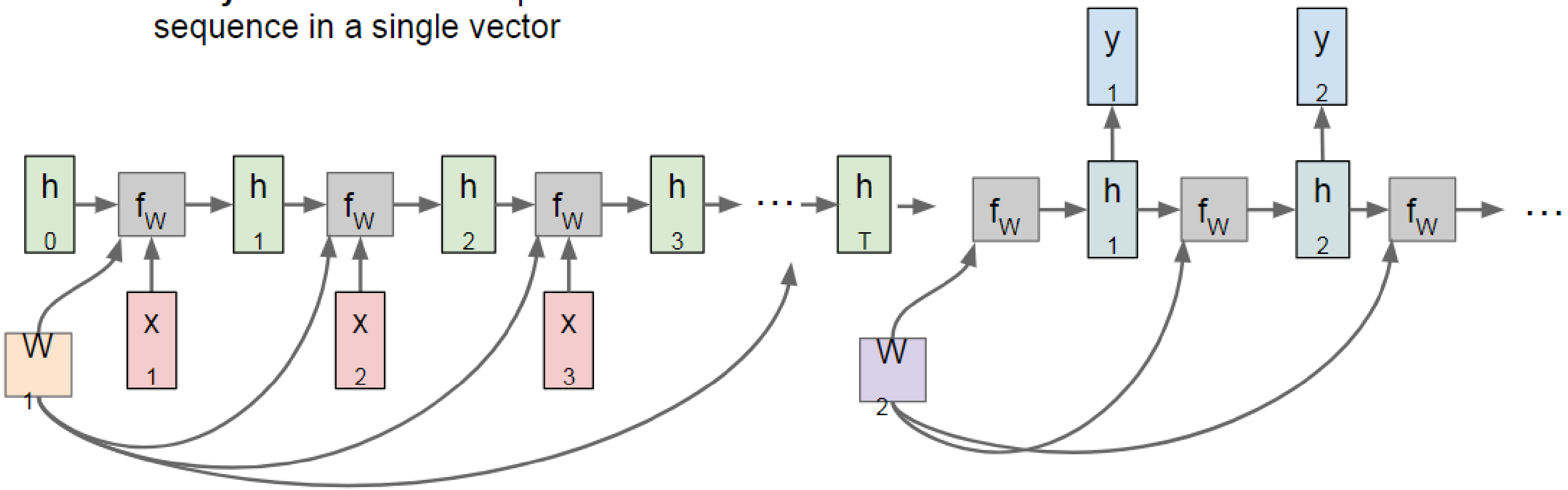
RNN: Computational Graph: One to Many



Sequence to Sequence: Many-to-one + one-to-many

One to many: Produce output sequence from single input vector

Many to one: Encode input sequence in a single vector



Long Short Term Memory (LSTM)

Vanilla RNN

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

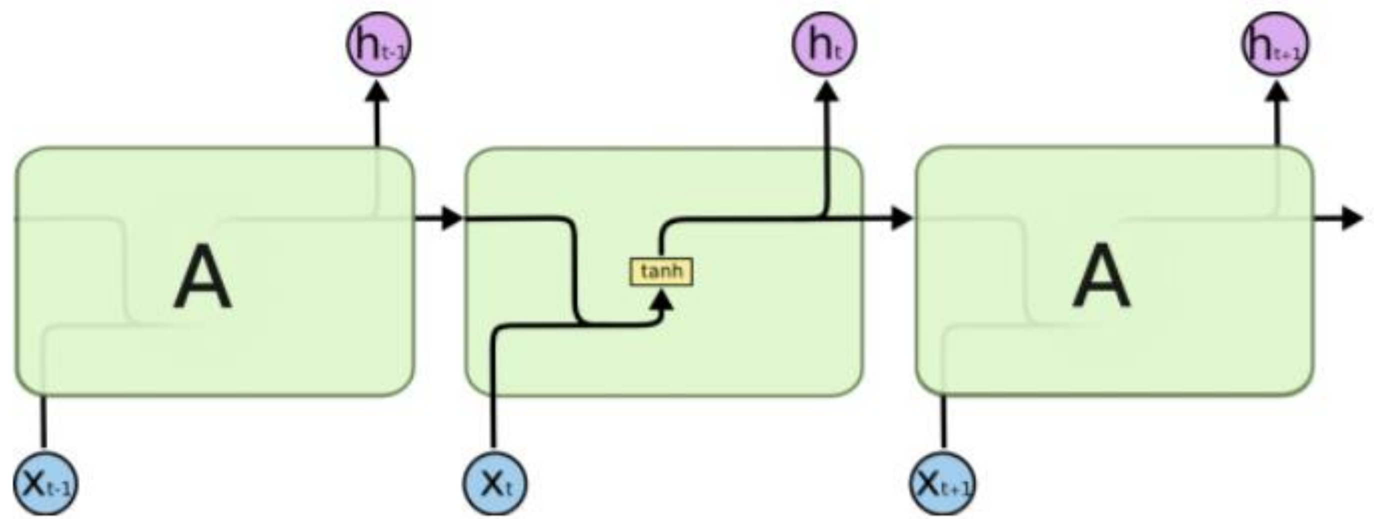
LSTM

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

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

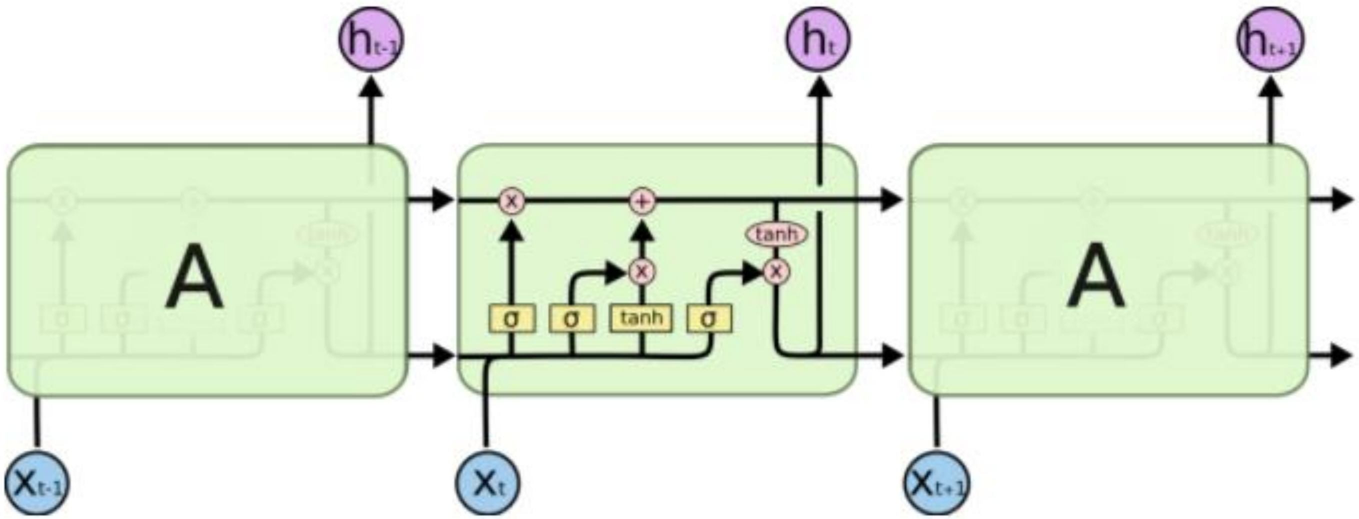
$$h_t = o \odot \tanh(c_t)$$

RNN Cell



The repeating module in a standard RNN contains a single layer.

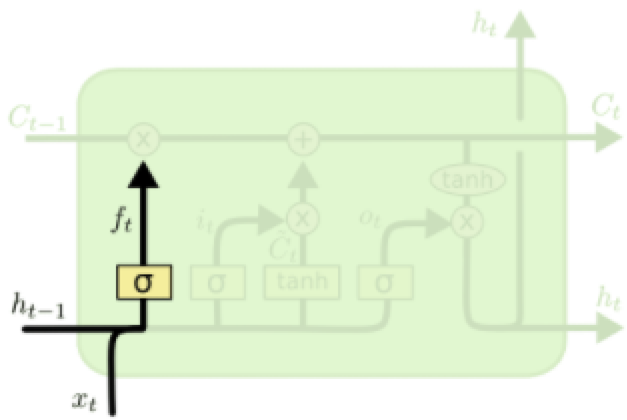
LSTM Cell



The repeating module in an LSTM contains four interacting layers.

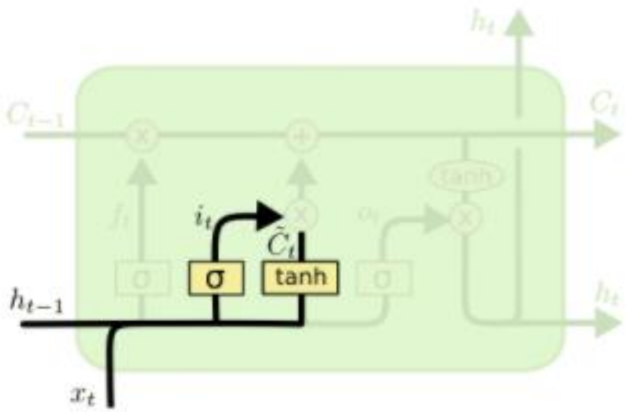
art center nabi v.do

LSTM Cell



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

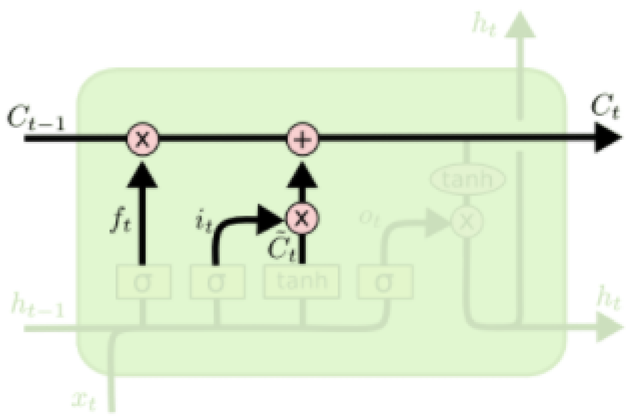
LSTM Cell



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

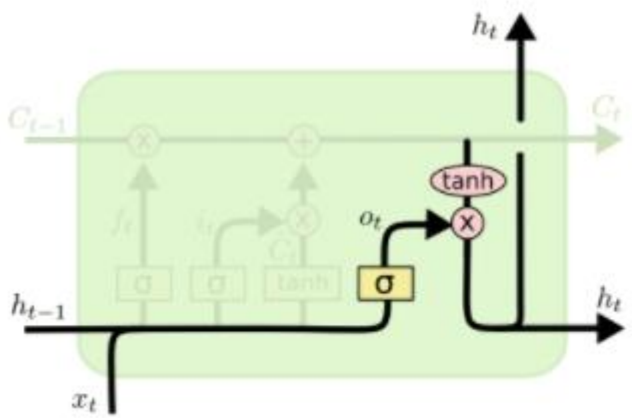
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \ + \ b_C)$$

LSTM Cell



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

LSTM Cell



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

Default RNN Cells in Tensorflow

- RNNCELL, GRUCELL, LSTMCELL

Base interface for all RNN Cells

- `tf.contrib.rnn.RNNCell`

Core RNN Cells for use with TensorFlow's core RNN methods

- `tf.contrib.rnn.BasicRNNCell`
- `tf.contrib.rnn.BasicLSTMCell`
- `tf.contrib.rnn.GRUCell`
- `tf.contrib.rnn.LSTMCell`
- `tf.contrib.rnn.LayerNormBasicLSTMCell`

- https://www.tensorflow.org/api_guides/python/contrib.rnn

RNN Cell

- Tf.contrib.rnn.RNNCell
  - Rnn cell, Lstm cell, GRU cell들의 부모 클래스

Properties

`output_size`

Integer or TensorShape: size of outputs produced by this cell.

RNN cell의 hidden node의 수

`state_size`

size(s) of state(s) used by this cell.

It can be represented by an Integer, a TensorShape or a tuple of Integers or TensorShapes.

Methods

`zero_state(batch_size, dtype)`

RNN cell의 hidden node의 초기값(0으로 초기화)

Return zero-filled state tensor(s).

Args:

- `batch_size` : int, float, or unit Tensor representing the batch size.
- `dtype` : the data type to use for the state.

Returns:

If `state_size` is an int or TensorShape, then the return value is a `N-D` tensor of shape `[batch_size x state_size]` filled with zeros.

BasicRNNCell

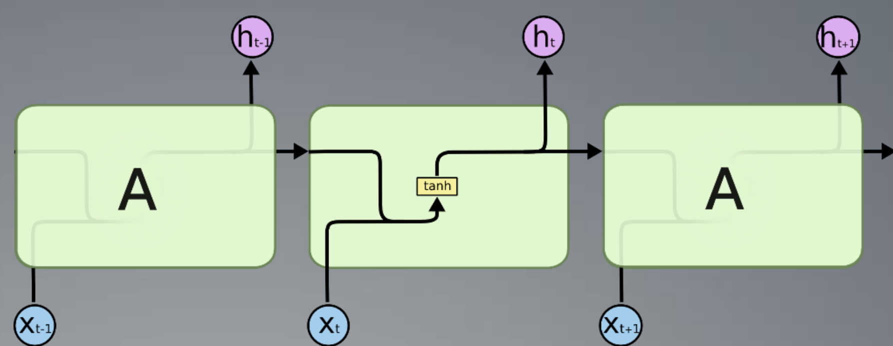
- Tensorflow.contrib.rnn.BasicRNNCell
  - 기본적인 RNN Cell : hidden node로만 이루어져 있음
  -

Deprecated and unused

```
tf.nn.rnn_cell.BasicRNNCell.__init__(num_units, input_size=None,
activation=tanh)
```

RNN cell의 hidden node의 수

Activation function 설정



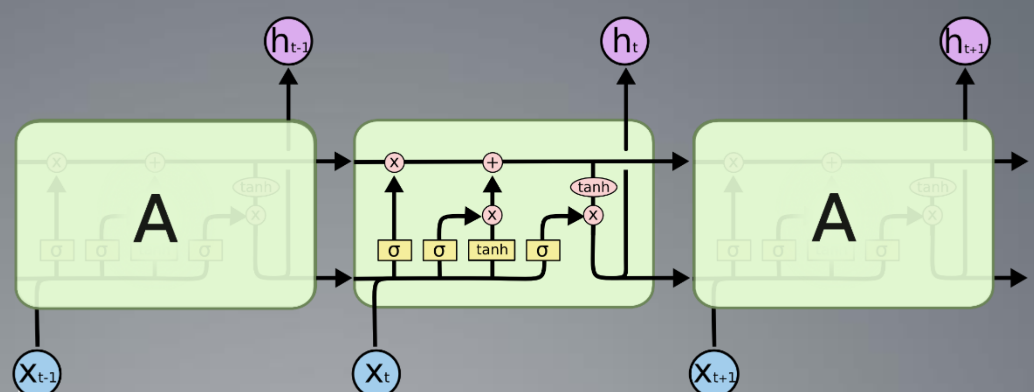
art center nabi v.do

BasicLSTMCell

- Tensorflow.contrib.rnn.BasicLSTMCell
  – LSTM 셀로 4개의 게이트로 이루어짐
  –

Forget gate의 bias 초기값

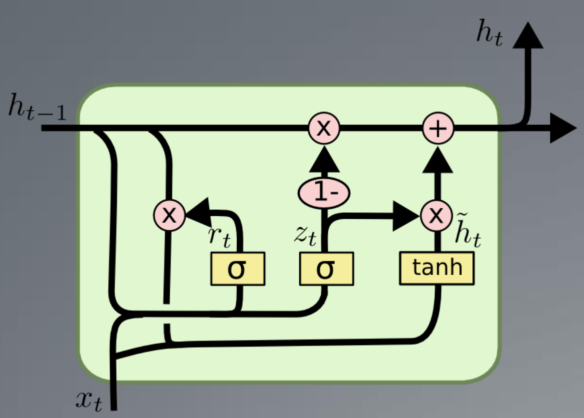
```
tf.nn.rnn_cell.BasicLSTMCell.__init__(num_units, forget_bias=1.0,
input_size=None, state_is_tuple=False, activation=tanh)
```



GRUCell

- Tensorflow.contrib.rnn.GRUCell
  - GRU 셀로 LSTM보다 간단한 구조

```
tf.nn.rnn_cell.GRUCell.__init__(num_units, input_size=None,
activation=tanh)
```



$$z_t = \sigma\left(W_z \cdot [h_{t-1}, x_t]\right)$$

$$r_t = \sigma\left(W_r \cdot [h_{t-1}, x_t]\right)$$

$$\tilde{h}_t = \tanh\left(W \cdot [r_t * h_{t-1}, x_t]\right)$$

$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t$$

RNN Cell wrappers

- Tensorflow.contrib.rnn.MultiRNNCell

```
tf.nn.rnn_cell.MultiRNNCell.__init__(cells, state_is_tuple=False)
```

- Tensorflow.contrib.rnn.DropoutWrapper

```
tf.nn.rnn_cell.DropoutWrapper.__init__(cell, input_keep_prob=1.0,
output_keep_prob=1.0, seed=None)
```

Dropout 비율

RNN Constructing Module

- Tensorflow.nn.dynamic_rnn

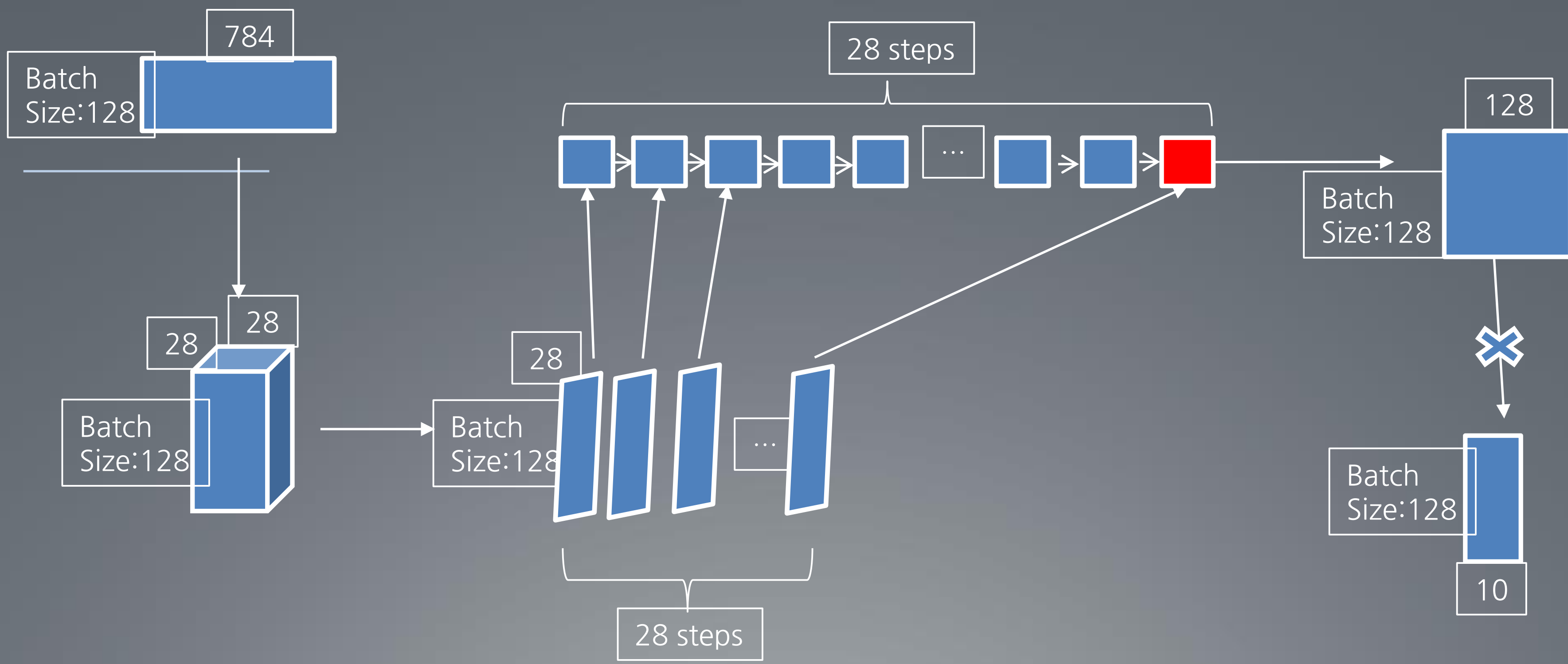
```
tf.nn.rnn(cell, inputs, initial_state=None,
dtype=None, sequence_length=None, scope=None)
```

Args:

- `cell`: An instance of RNNCell. RNN cell
- `inputs`: A length T list of inputs, each a `Tensor` of shape `[batch_size, input_size]`, or a nested tuple of such elements. Input data
- `initial_state`: (optional) An initial state for the RNN. If `cell.state_size` is an integer, this must be a `Tensor` of appropriate type and shape `[batch_size, cell.state_size]`. If `cell.state_size` is a tuple, this should be a tuple of tensors having shapes `[batch_size, s] for s in cell.state_size`. State의 초기값
- `dtype`: (optional) The data type for the initial state and expected output. Required if initial_state is not provided or RNN state has a heterogeneous dtype.
- `sequence_length`: Specifies the length of each sequence in inputs. An int32 or int64 vector (tensor) size `[batch_size]`, values in `[0, T)`.
- `scope`: VariableScope for the created subgraph; defaults to "RNN".

art cen

실습 : MNIST 예제



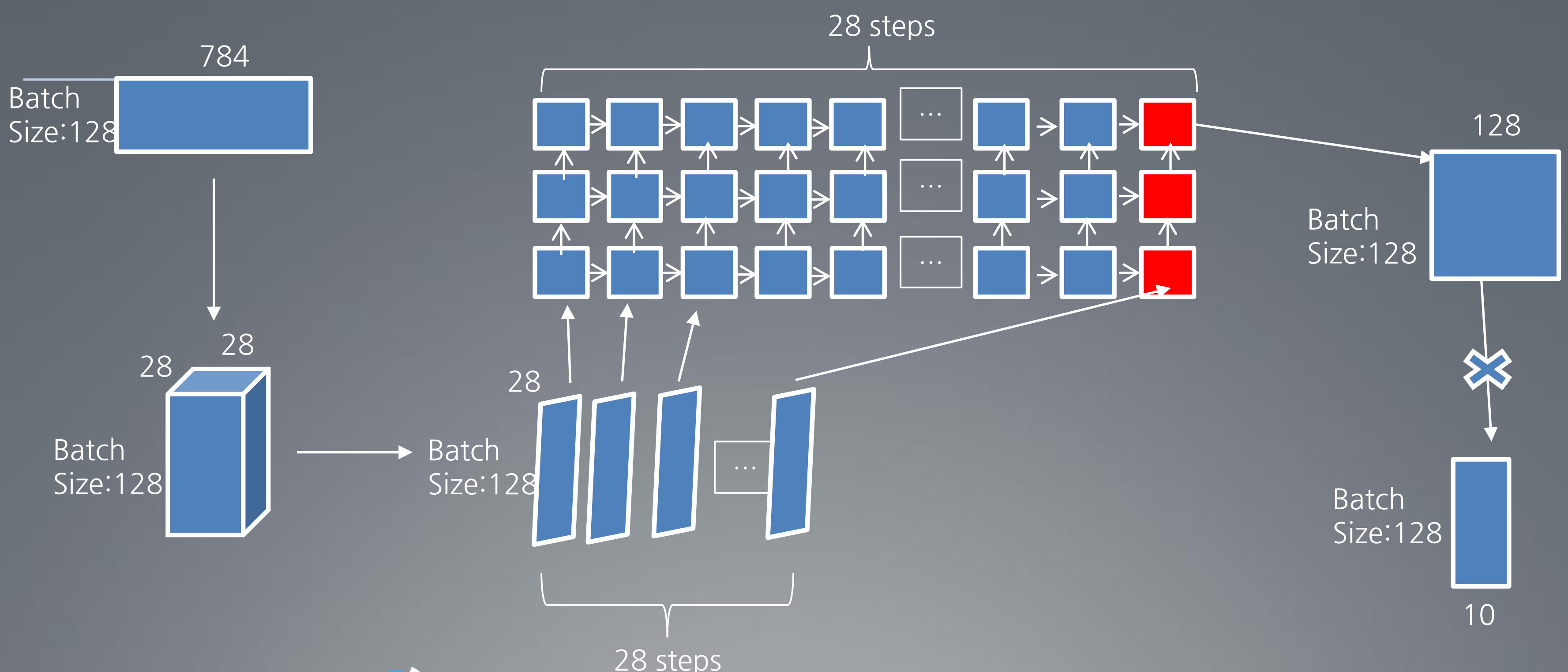
art center nabi v.do

Multi RNN

```
cell = tf.contrib.rnn.BasicRNNCell(n_hidden)
cell = tf.contrib.rnn.DropoutWrapper(cell, output_keep_prob=0.5)
cell = tf.contrib.rnn.MultiRNNCell([cell] * num_layers)
outputs, states = tf.nn.dynamic_rnn(cell, x_t, dtype=tf.float32)
```

- RNN Cell의 List를 MultiRNNCell의 initial argument로 입력
- DropoutWrapper를 이용하여 layer간에 dropout적용

실습 : MNIST 예제


784
Batch
Size:128
28
28
Batch
Size:128
Batch
Size:128
28
28 steps
…
28 steps
…
…
…
128
Batch
Size:128
Batch
Size:128
10

실습: Text Generation(1/5)

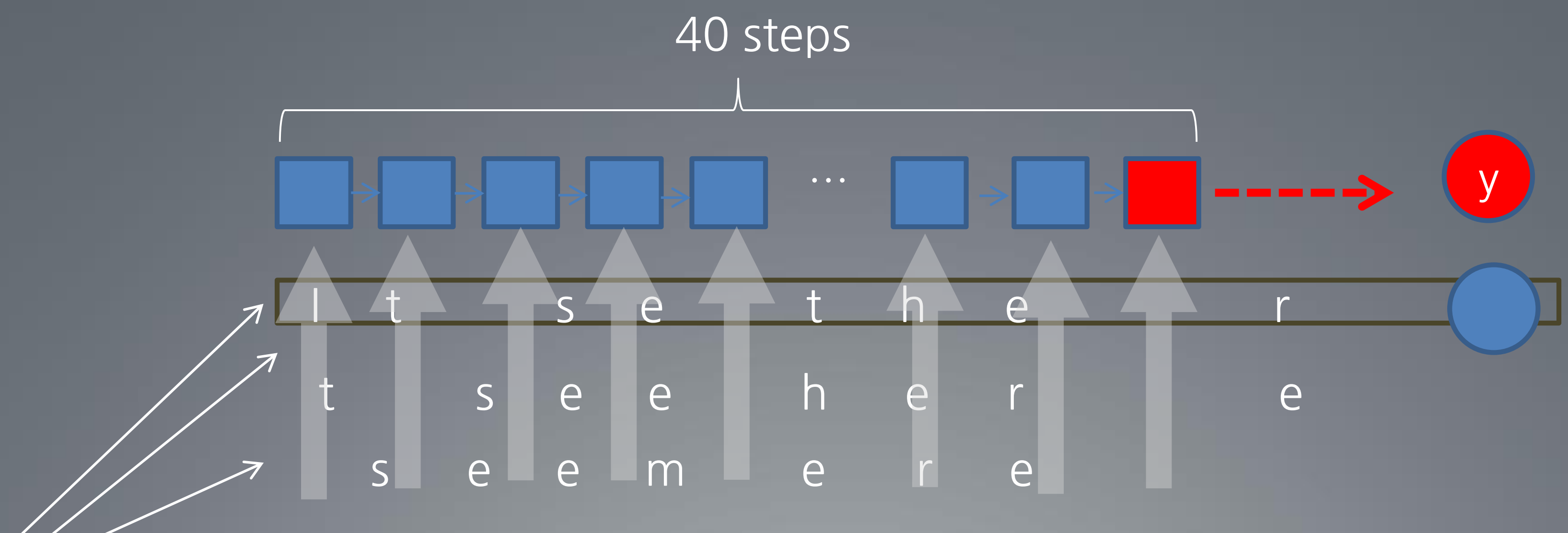
- RNN이 가장 많이 이용되는 분야인 NLP 예제
- 그 중 word단위가 아닌 character단위로 텍스트 분석 및 예측
- 알파벳 character 전후 관계와 문장 전체의 information을 이용

Text Generation (2/5)

- Dataset : nietzsche의 선악의 저편 text (OSIA/data)
- 40 steps의 LSTM 1 layer를 이용하여 다음 character 예측

Text Generation (3/5)

- 데이터 전처리
  - Character 59개를 모두 index화
  - 1 character씩 움직이면서 sentence와 예측할 다음 character를 target으



It seems to me that there is everywhere an attempt at present to divert attention

Text Generation (4/5)

- Training Detail
  - Adam optimizer 사용
  - Learning rate = 0.01
  - Batch size = 128
  - LSTM Hidden cell의 수 = 128

Text Generation (5/5)

- Character Sampling
  - 1000 iteration마다 200 characters 연속 생성
  - 생성되는 character를 다시 input으로 사용
  - 트레이닝이 진행될수록 문장을 이루는 character 생성

감사합니다