딥러닝 9

현운용

Recurrent Neural Networks = RNNs

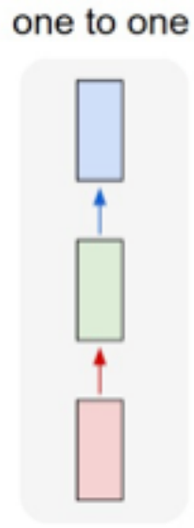
RNNs

인간이 정보를 받아들일때,
기존에 있었던 관성/고집 으로 인해 정보를 해석하여 받아들임.

RNNs

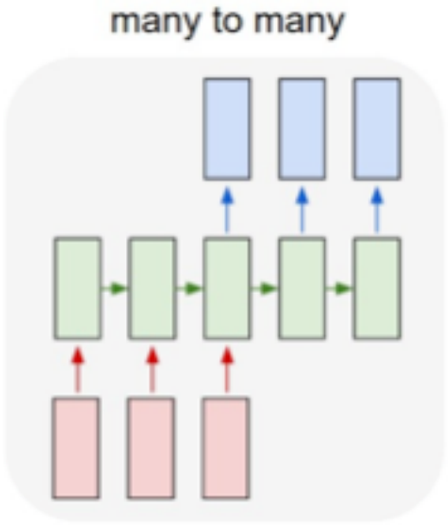
Traditional Neural Networks : DNN

낮에는 치킨장사! 밤에는 잠복근무!
지금까지 이런 수사는 없었다!

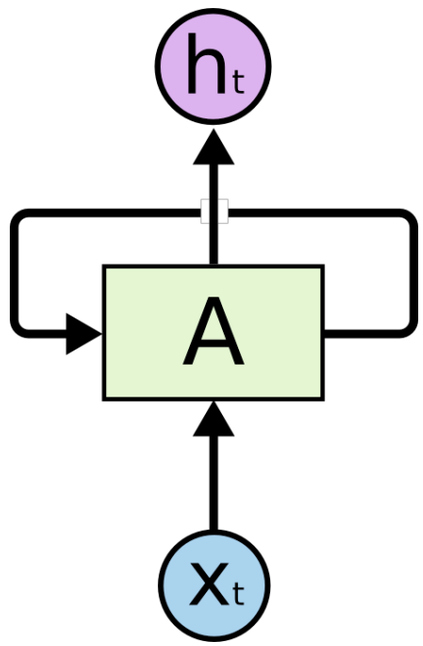


Recurrent Neural Networks : RNNs

낮에는 치킨장사! 밤에는 잠복근무!
지금까지 이런 수사는 없었다!



RNNs

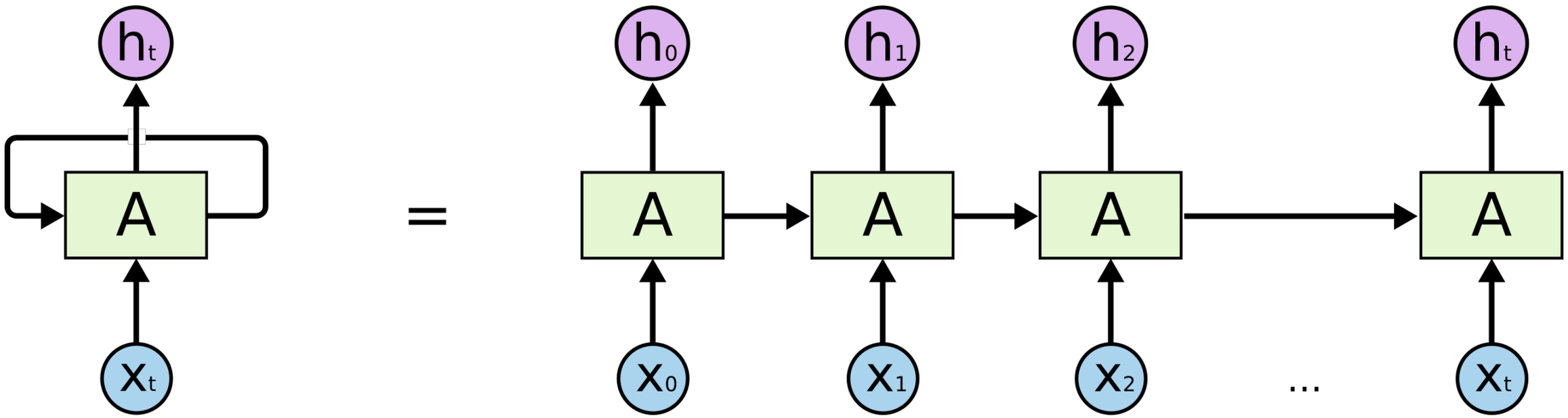


A는 하나의 Layer
Xt는 입력
Ht는 출력

Recurrent Neural Networks have loops

RNNs

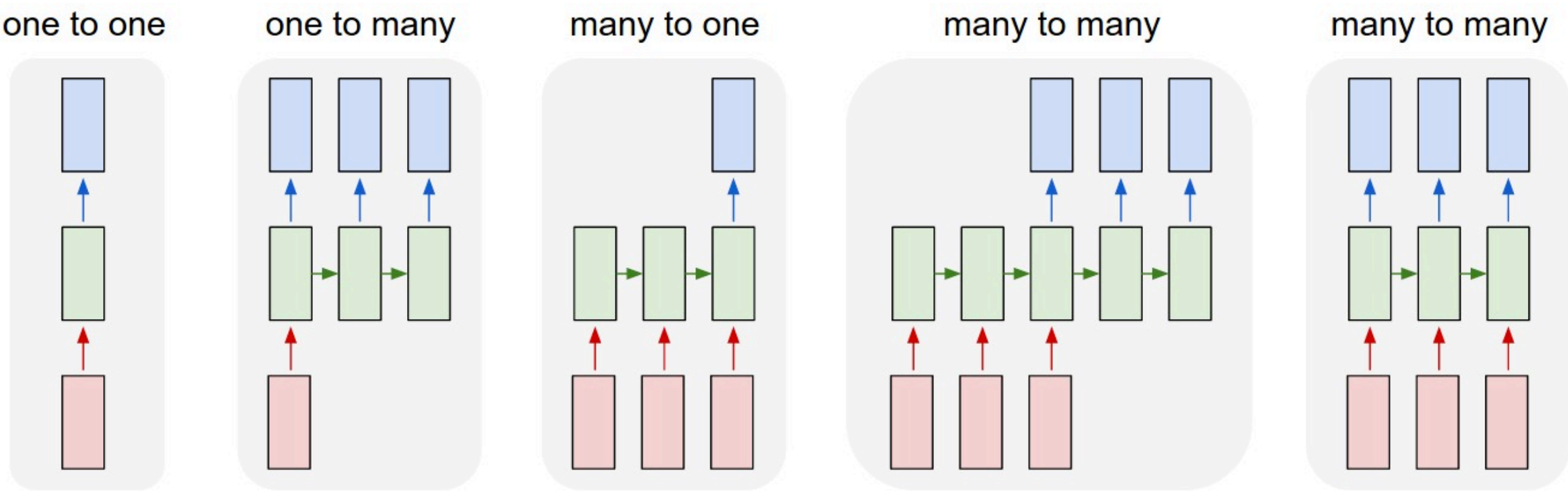
t개의 입력에 대한 뉴럴 연산(행렬 곱) 연산을 수행하면서,
이전의 연산을 기억하여 다음 뉴럴 연산에 참조함



An unrolled recurrent neural network

RNNs

Traditional Neural Networks



RNNs

Speech Recognition

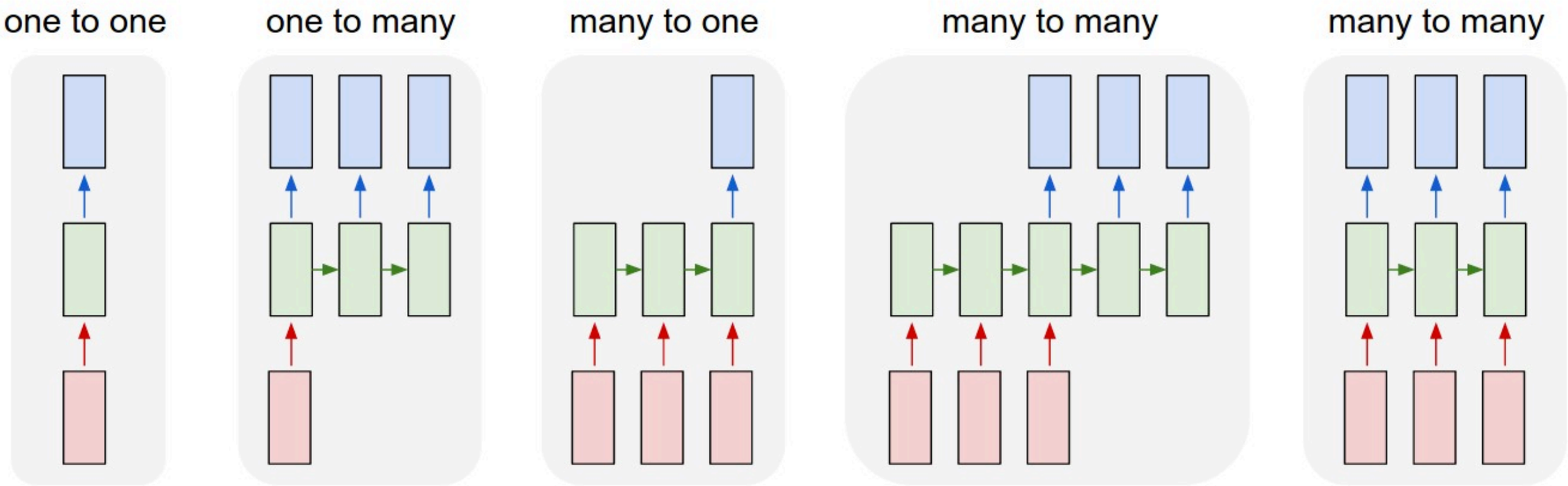
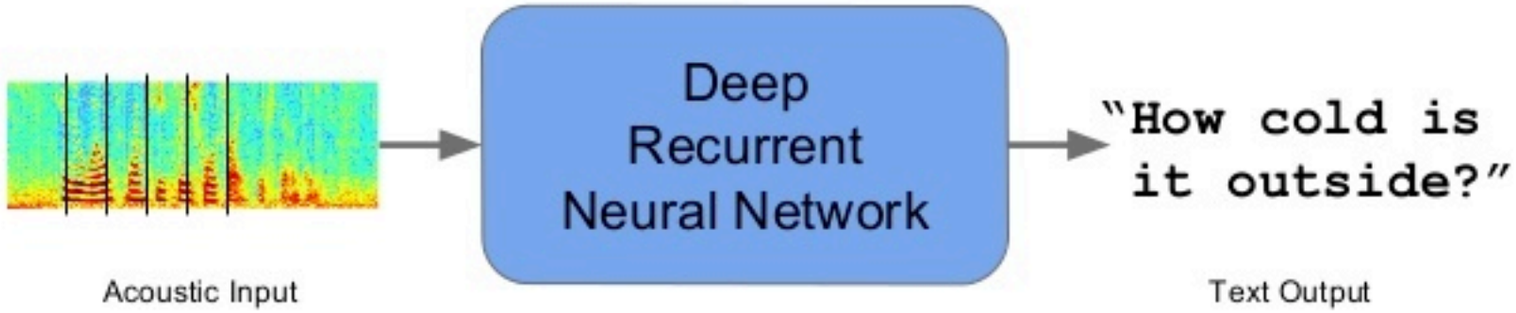


Image Captioning

Translation

RNNs

Speech Recognition

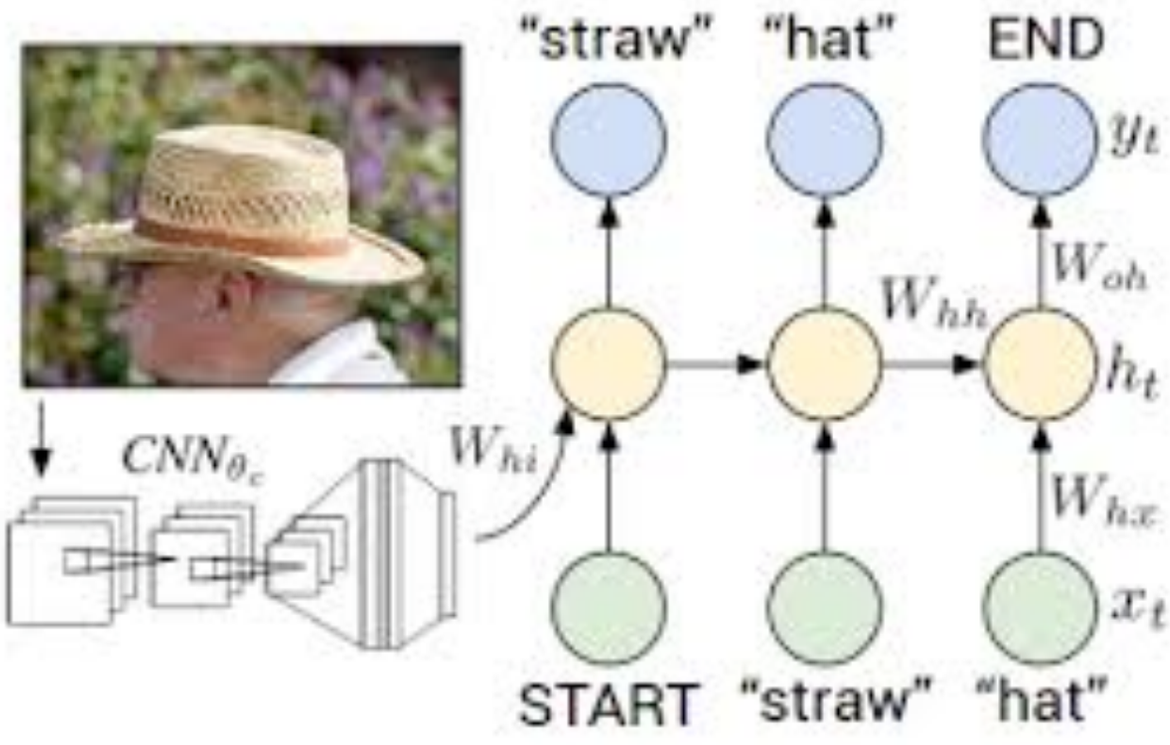


Reduced word errors by more than 30%

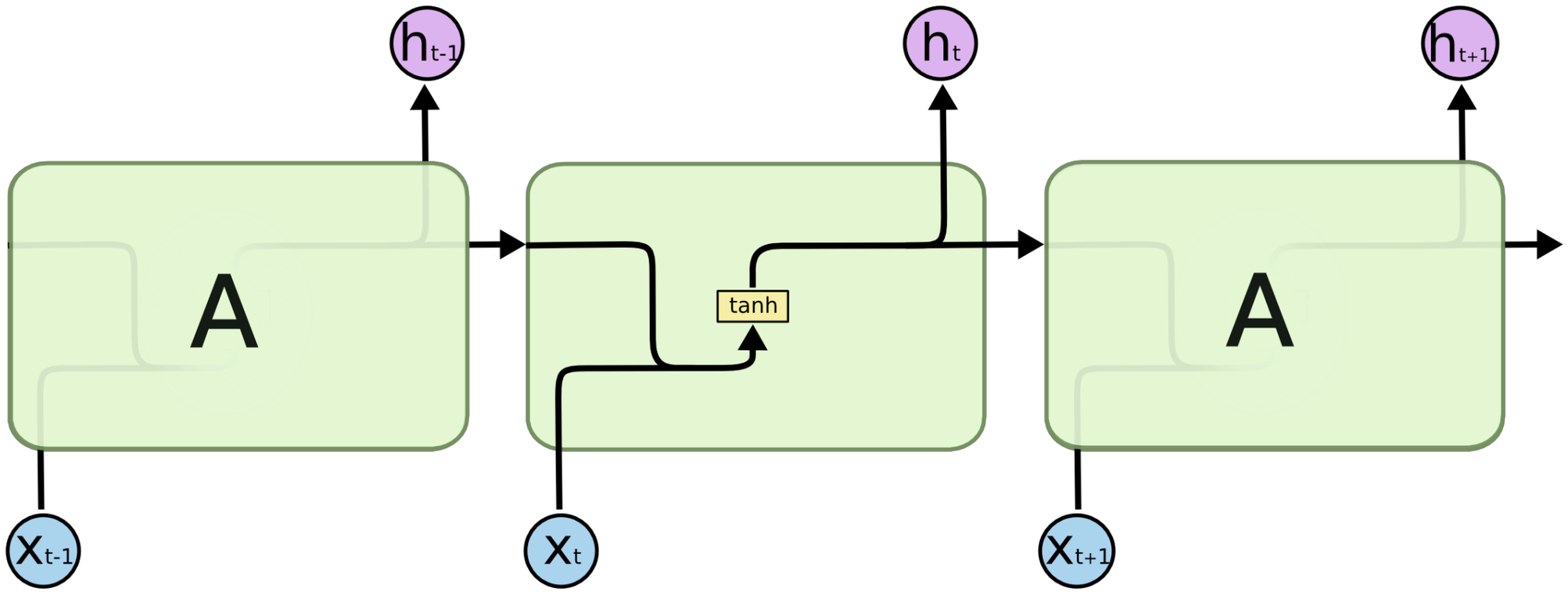
Google Research Blog - August 2012, August 2015

Research at Google

RNNs


"straw"
"hat"
END
y_t
W_{oh}
W_{hh}
h_t
W_{hi}
W_{hx}
x_t
CNN_{θ_c}
START
"straw"
"hat"

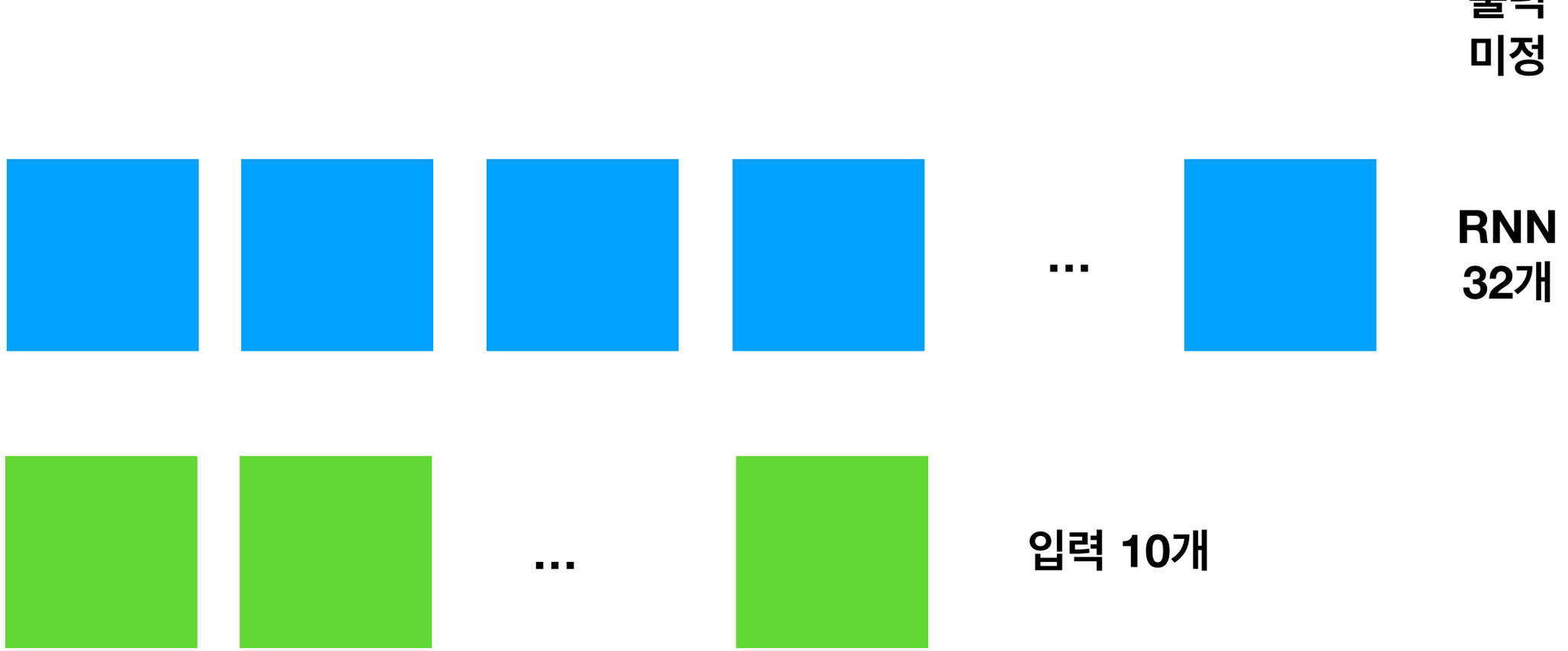
RNNs



입력 계산을 반복해서 처리하고 있는 단일 층 Standard RNN

RNNs

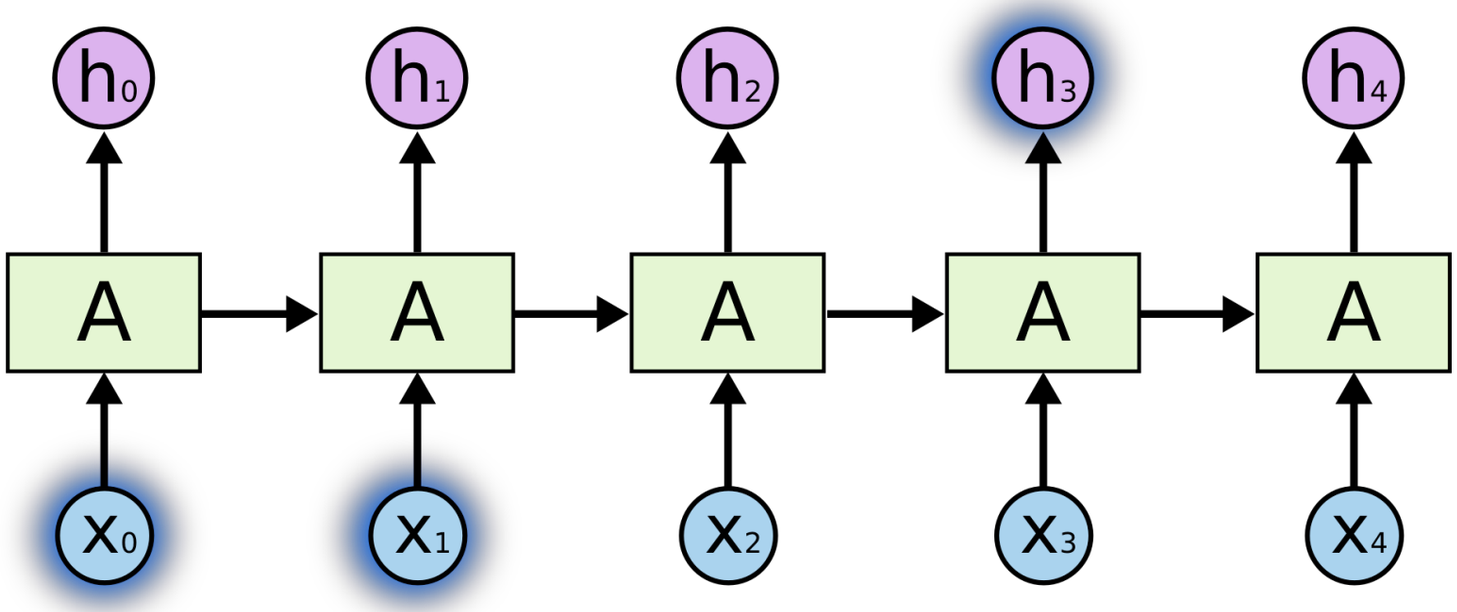
```
# Create a simple Keras model
        model = Sequential()
        model.add(SimpleRNN(32, input_dim=32, input_length=10))
```



RNNs

Dependencies 의존

RNN의 핵심적인 장점은 이전 정보를 기억하고 있다는 것.

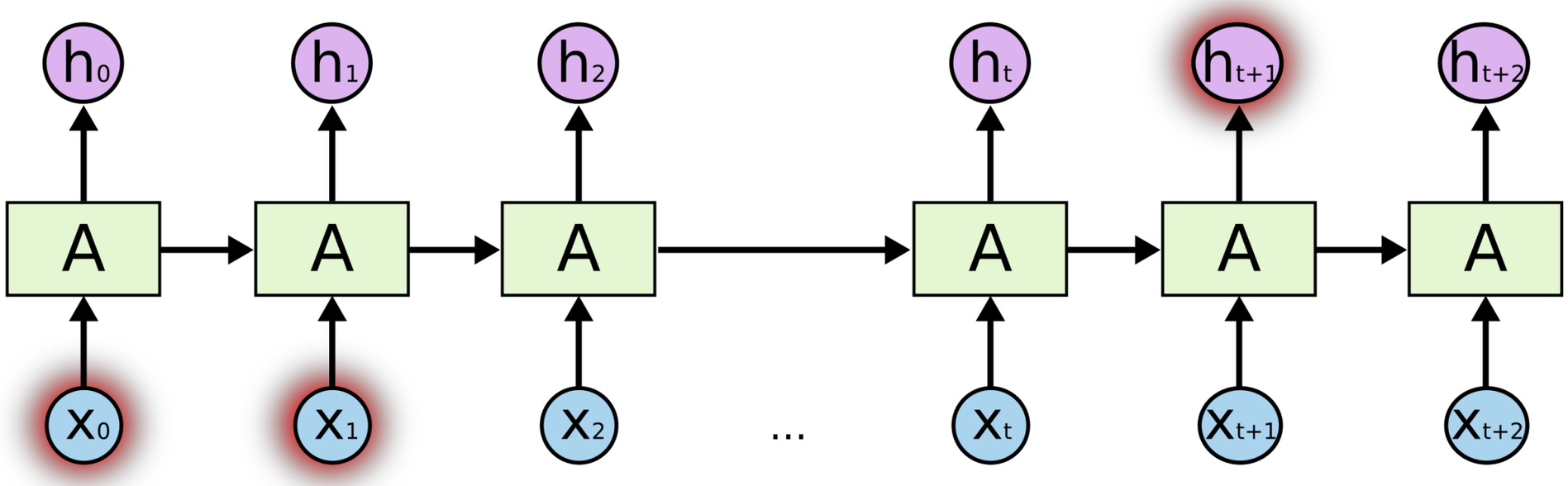


다음에 올 단어를 예측하고자 할때

the clouds are in the sky.

RNNs

하지만 쌓여있는 데이터가 많을 수록 예측이 힘들어짐.

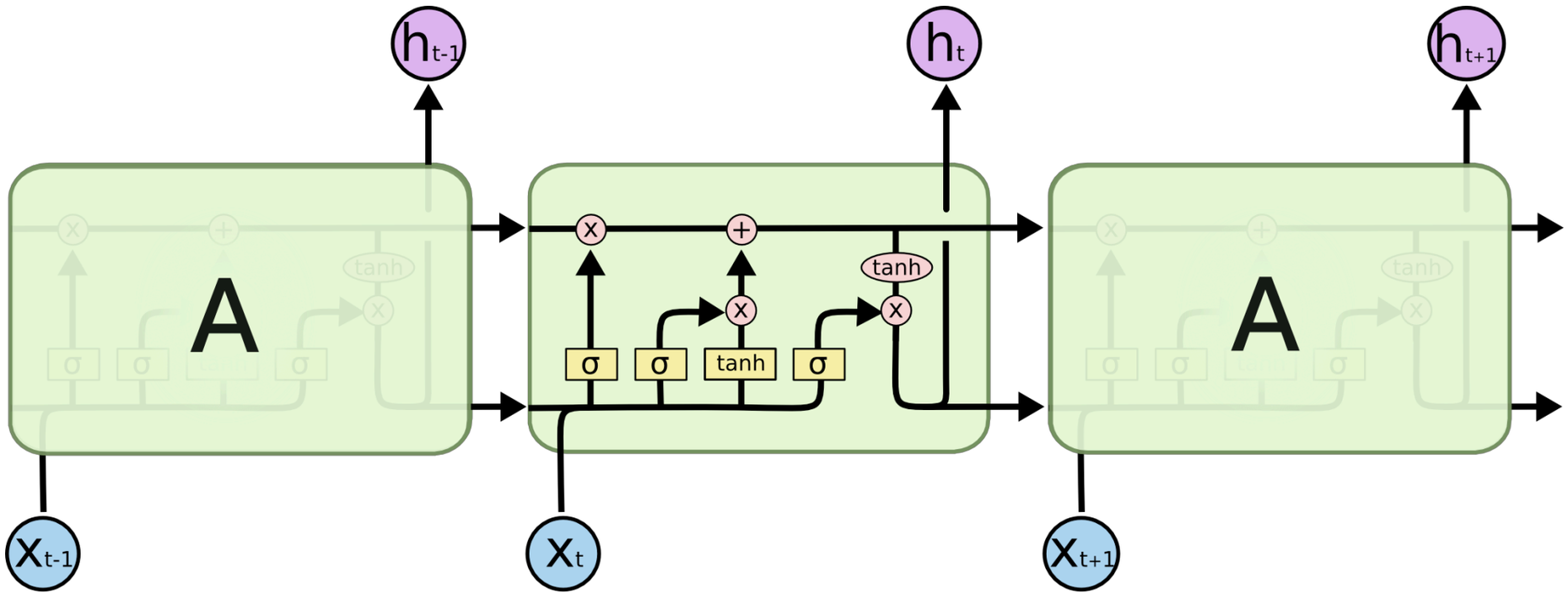


다음에 올 단어를 예측하고자 할때

I grew up in France ..... I speak fluent

Long-Term-Dependencies 문제

RNNs

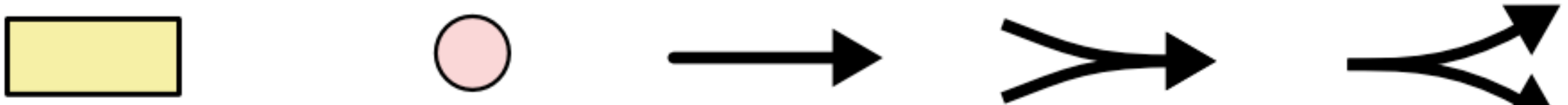


Long-Short-Term-Memory Neural Networks
LSTM

LSTM은 Long의존도가 떨어지는 현상을 방지하기 위해
4가지 Interaction을 만들었습니다.

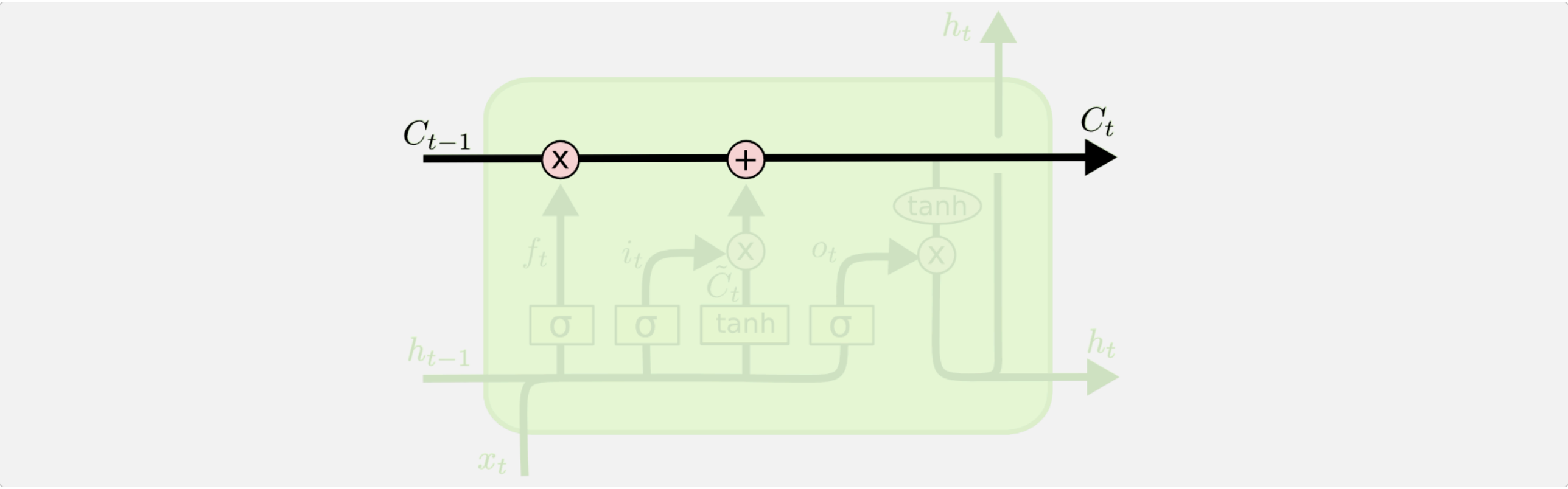
RNNs

간단한 도식 설명



Neural Network Layer | Pointwise Operation | Vector Transfer | Concatenate | Copy

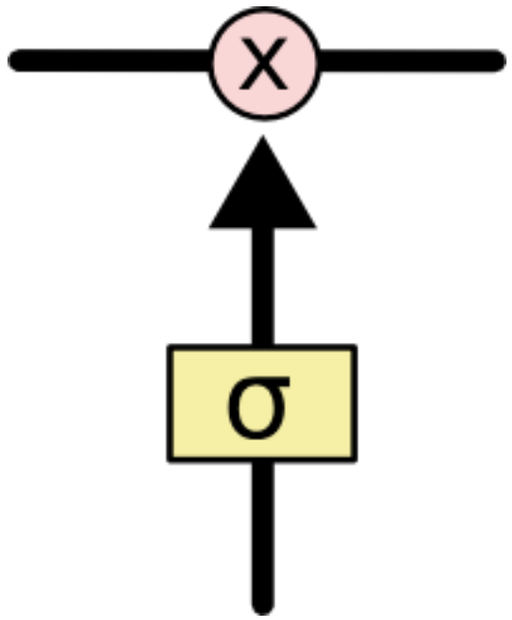
RNNs

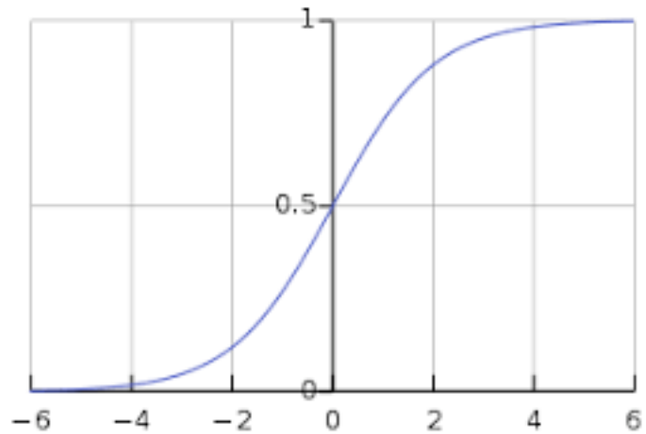


LSTM 레이어를 가로지르는 수평선 하나는 셀 상태를 나타내며
LSTM의 핵심 아이디어입니다.

이전의 상태를 받아서, 게이트를 통해 정보를 추가하거나 제거하고,
다음 LSTM에 상태를 전달해주는 것 입니다.

RNNs

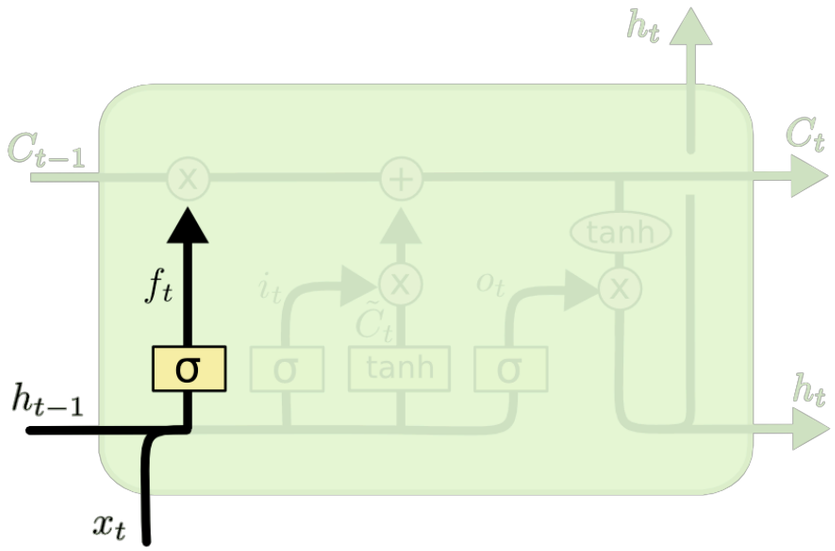




Sigmoid 함수로 데이터를 어마나 통과시킬지를 의미합니다.
0은 데이터를 없애는 것이고,
1은 데이터를 모두 통과시키는 것 입니다.

LSTM에는 셀정보를 보호하고 제어하기 위한 3가지 게이트가 있습니다.

RNNs



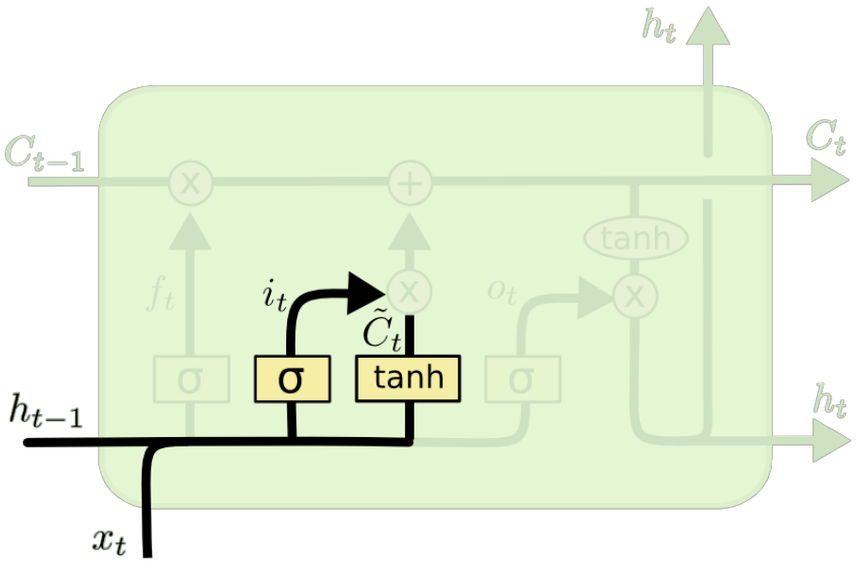
이전의 출력, 현재의 입력

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

첫번째 단계
“이전의 상태를 얼만큼 가지고 갈지!”
“forgot gate layer”
h[t-1],x[t] 의 시그모이드로 계산된 값이
이전의 상태값을 얼마만큼 잊어버리는 지를 결정합니다.

결정되었으면, 셀의 상태를 업데이트 합니다.

RNNs



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

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

두번째 단계

“input layer”

h[t-1],x[t] 의 시그모이드로 계산된 값이

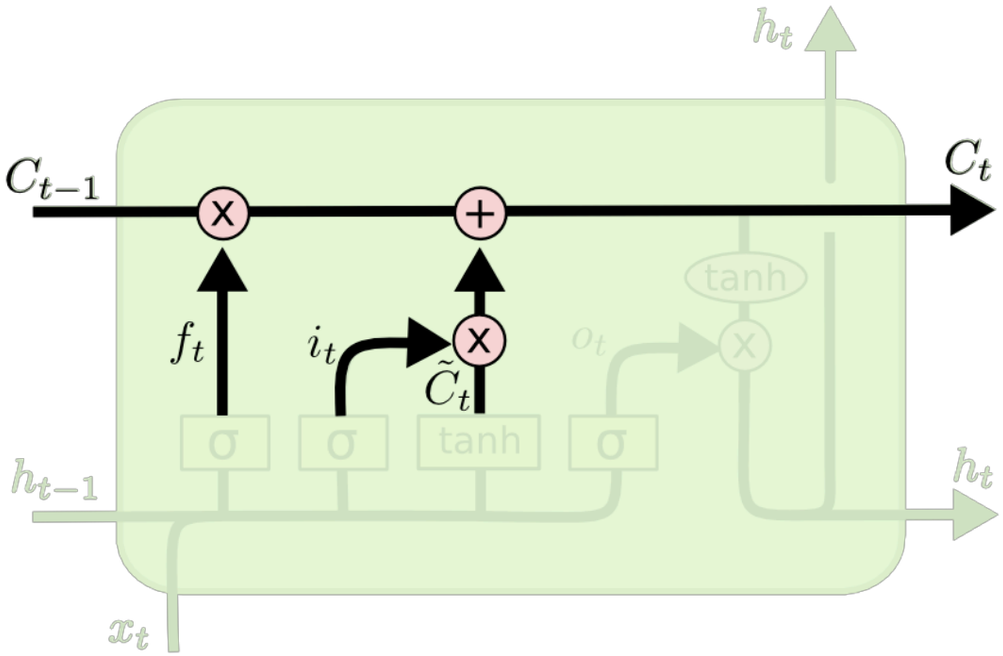
x[t]가 얼마만큼 입력될 것인지를 계산합니다 == i[t]

h[t-1],x[t] 의 탄젠트로 계산된 값이

h[t-1]으로 부터 얼마만큼 후보를 선정할 것인지를 계산합니다 == C[t]

두 후보를 합치고 셀의 상태를 업데이트 합니다.

RNNs

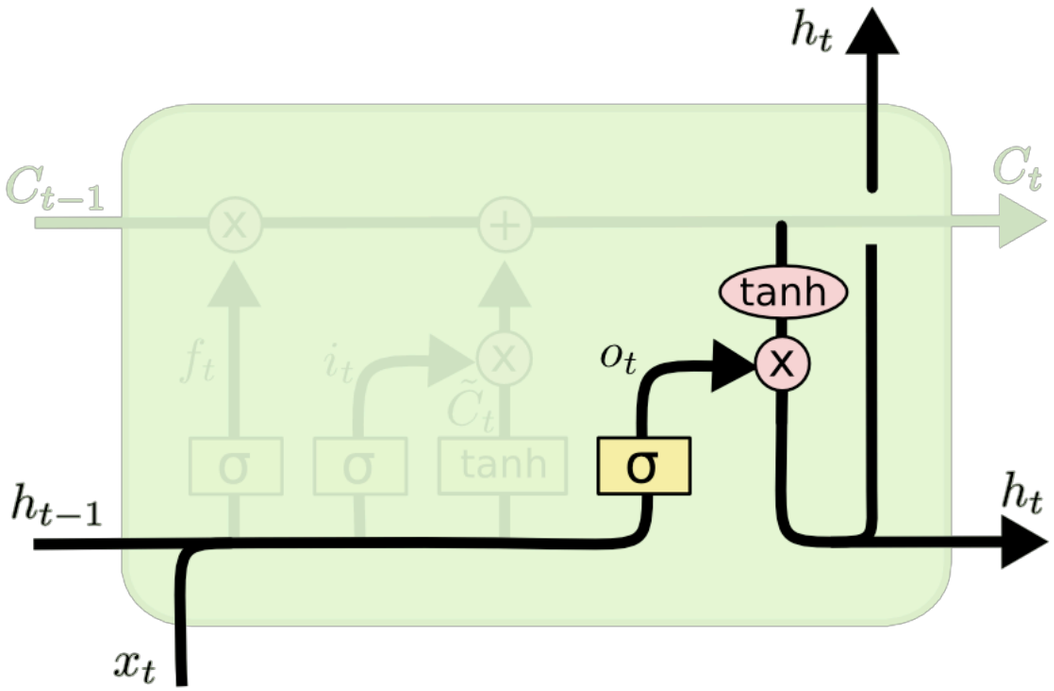


$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

셀의 상태를 업데이트 과정

언어 모델에서 비추어본다면, 첫번째 스텝으로 이전의 주제를 없애고
두번째 스텝으로 새로운 주제를 얼마만큼 선정할건지

RNNs



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

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

3번째 단계

“output layer”

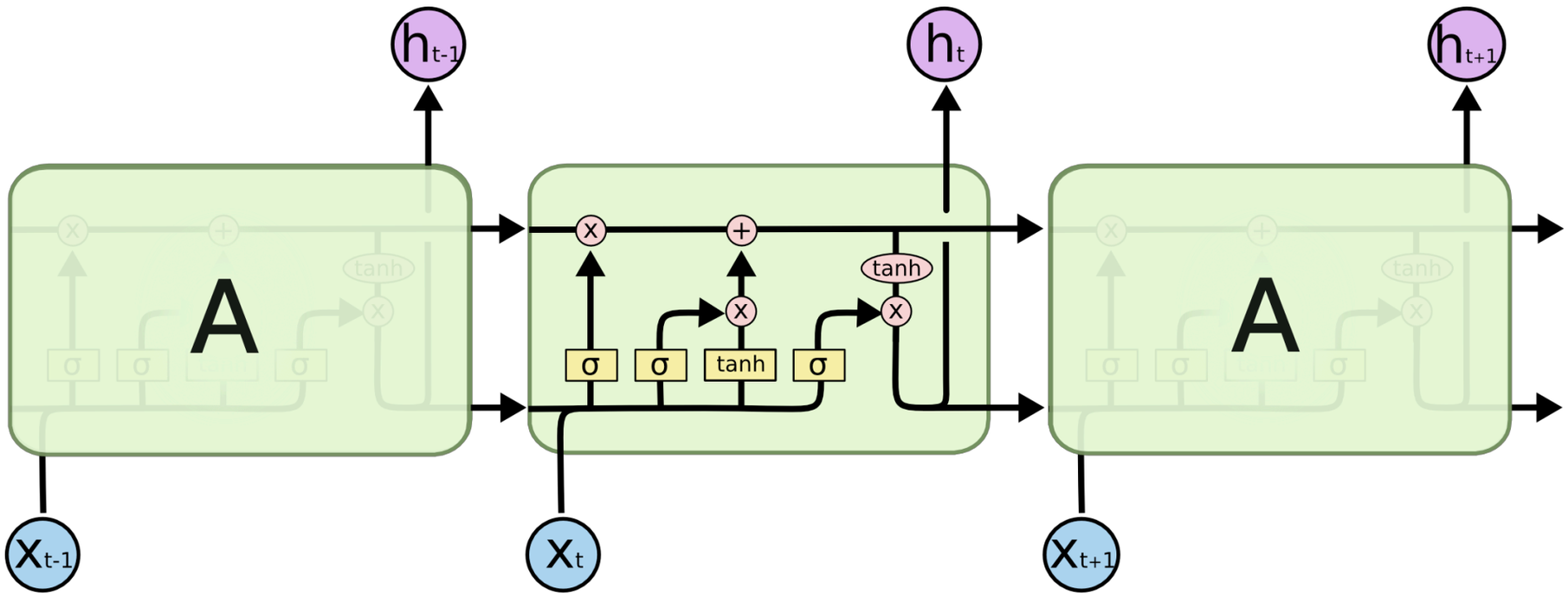
주제가 다 결정이 된 상태에서

입력 x[t],h[t-1] 중에서 다음에 올 것을 주제와 연관하여 출력.

언어모델에서 예를들면, 현재 글에서 대상이 단수인지 복수인지를 알 수 있기 때문에,

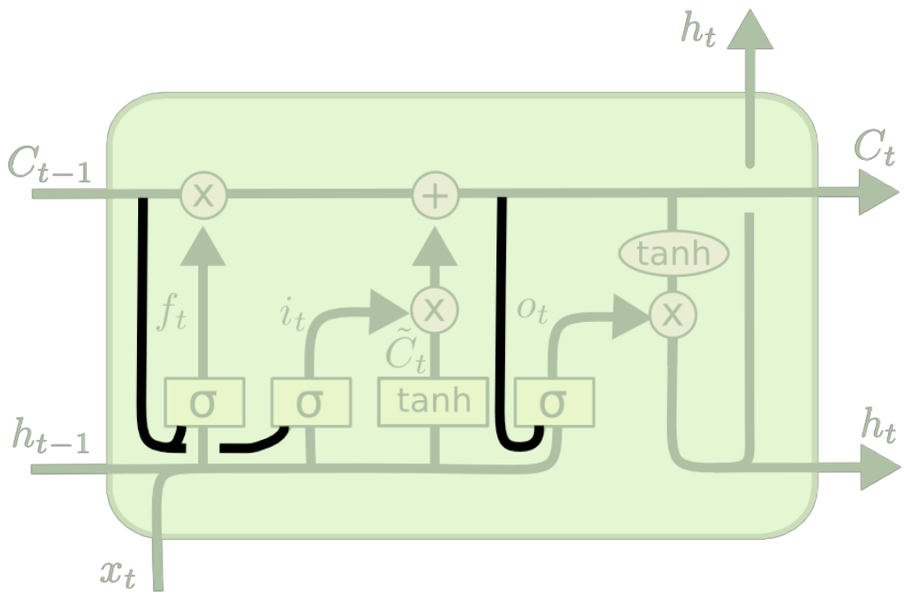
다음에 오는 단어가 어떤 형태로 결합되어야 하는지를 알 수 있습니다.

RNNs



지금까지 기본적인 LSTM의 형태 였습니다.
상태와 입력 -> 상태와 출력에서
장기적으로 상태가 유지되기 힘든점을 보완하기 위해
장기적으로 데이터를 가지고 갈지 말지를 결정하는 3가지의 내부 Interaction으로
상태와 입력 -> 상태와 출력을 만들었습니다.

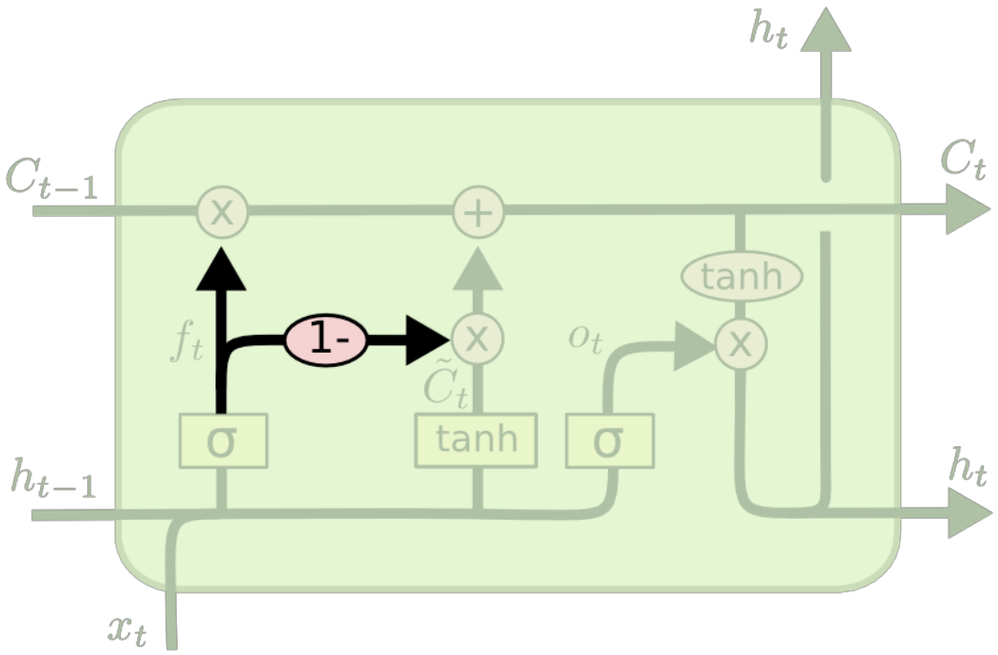
RNNs



$$f_t = \sigma\left(W_f \cdot [\boldsymbol{C_{t-1}}, h_{t-1}, x_t] \; + \; b_f\right)$$
$$i_t = \sigma\left(W_i \cdot [\boldsymbol{C_{t-1}}, h_{t-1}, x_t] \; + \; b_i\right)$$
$$o_t = \sigma\left(W_o \cdot [\boldsymbol{C_t}, h_{t-1}, x_t] \; + \; b_o\right)$$

위의 LSTM은 변형된 LSTM으로서 “peephole connection LSTM” 이라고 합니다.
각 Interaction 구간에서 단순히 LSTM의 상태를 변화 시키는 것이 아니라
LSTM의 상태 역시 변수 중 하나로 염두하는 것이 특징 입니다.

RNNs

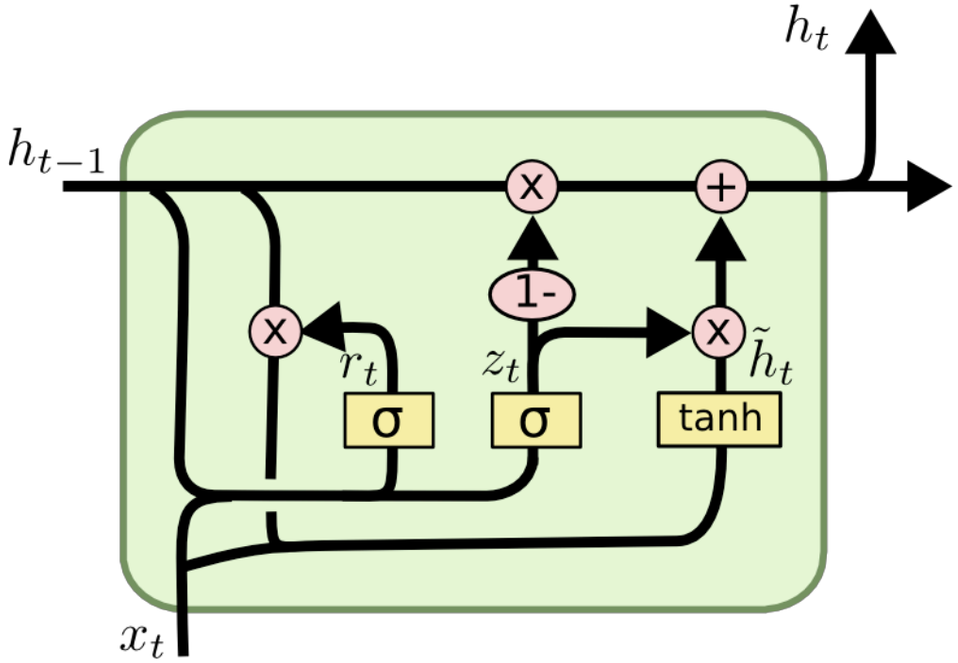


$$C_t = f_t * C_{t-1} + (1 - \boldsymbol{f_t}) * \tilde{C}_t$$

LSTM 변형 모델
망각 게이트의 출력을 입력게이트로 전이함.

이것의 효과는 잃어버린것이 많으면 많을 수록 상태를 채우는 것 입니다.

RNNs



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

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

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

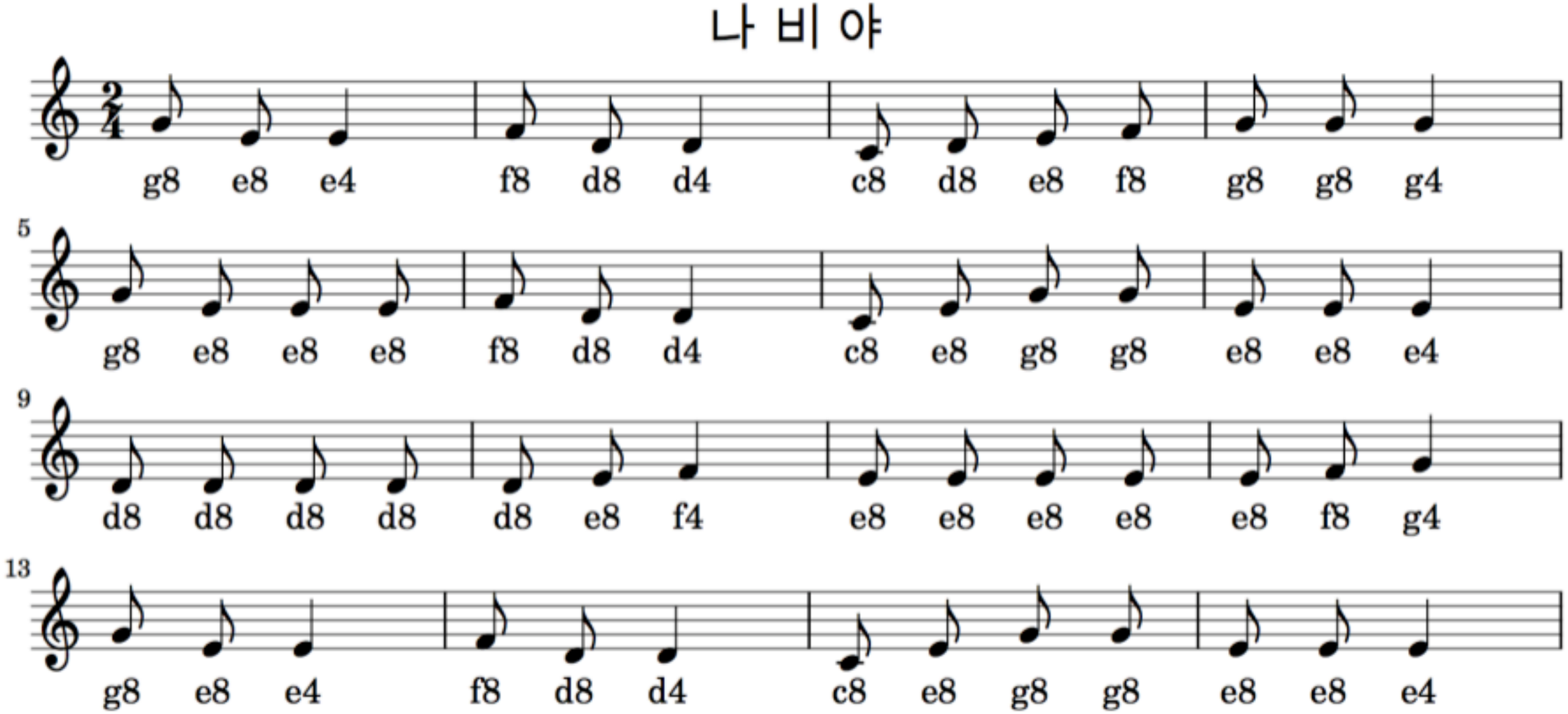
$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t$$

GRU(Gated Recurrent Unit)

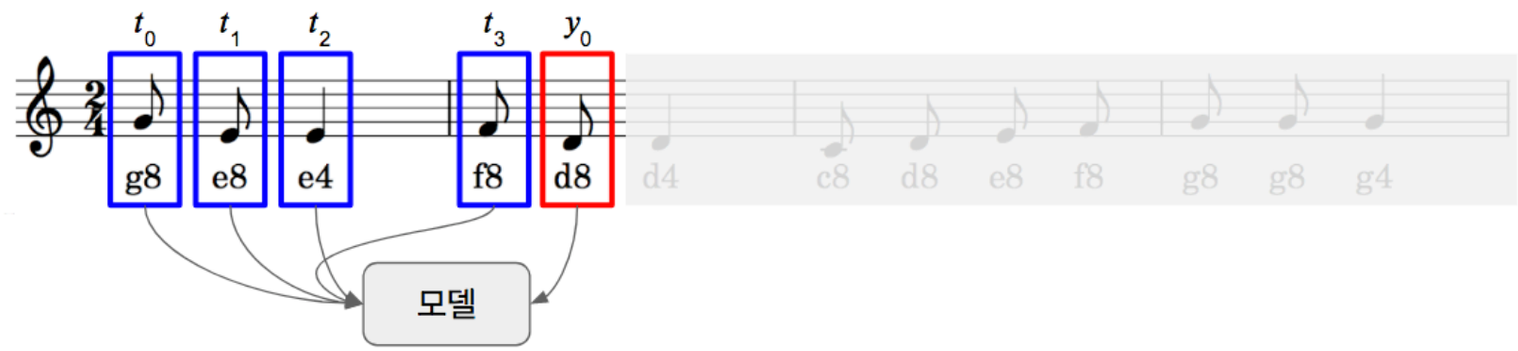
https://arxiv.org/pdf/1406.1078v3.pdf

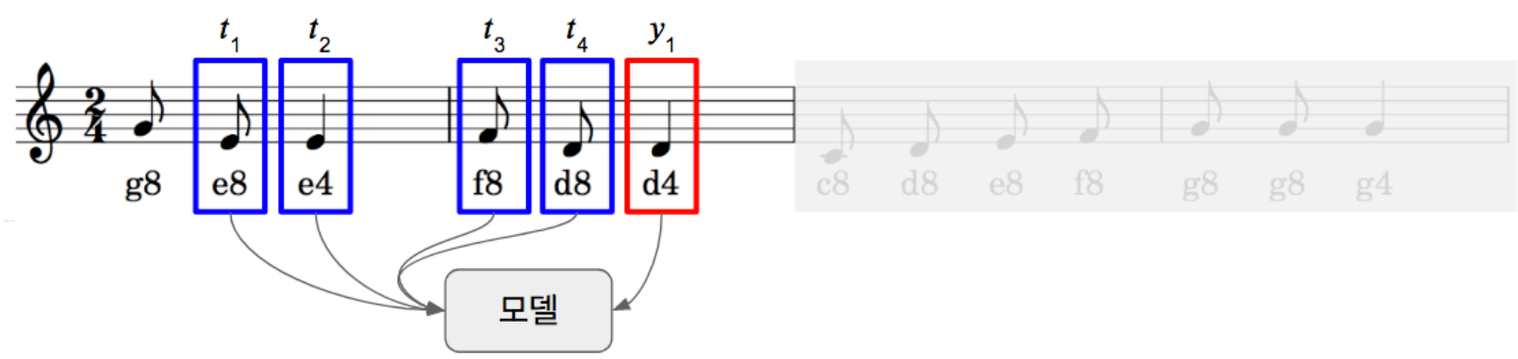
Cho Kyunghyun

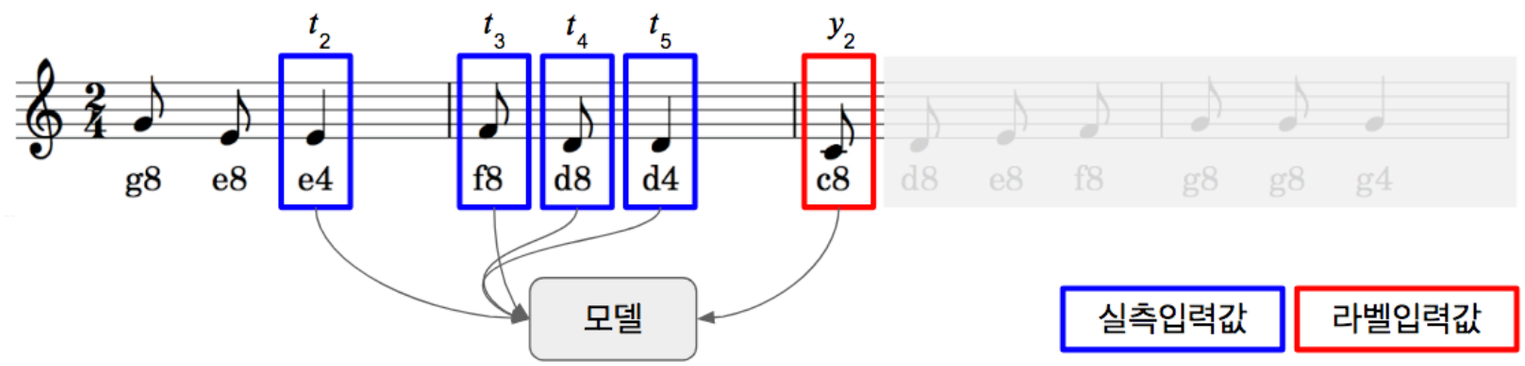
RNNs


나 비 야
g8 e8 e4 f8 d8 d4 c8 d8 e8 f8 g8 g8 g4
5
g8 e8 e8 e8 f8 d8 d4 c8 e8 g8 g8 e8 e8 e4
9
d8 d8 d8 d8 d8 e8 f4 e8 e8 e8 e8 e8 f8 g4
13
g8 e8 e4 f8 d8 d4 c8 e8 g8 g8 e8 e8 e4

RNNs


t_0 t_1 t_2 t_3 y_0
g8 e8 e4 f8 d8
d4 c8 d8 e8 f8 g8 g8 g4
모델


t_1 t_2 t_3 t_4 y_1
g8 e8 e4 f8 d8 d4
c8 d8 e8 f8 g8 g8 g4
모델


t_2 t_3 t_4 t_5 y_2
g8 e8 e4 f8 d8 d4 c8
d8 e8 f8 g8 g8 g4
모델
실측입력값
라벨입력값

RNNs

음표를 어떻게 데이터로 만들까?
One-Hot Encode

```
code2idx = {'c4':0, 'd4':1, 'e4':2, 'f4':3, 'g4':4, 'a4':5, 'b4':6,
            'c8':7, 'd8':8, 'e8':9, 'f8':10, 'g8':11, 'a8':12, 'b8':13}

idx2code = {0:'c4', 1:'d4', 2:'e4', 3:'f4', 4:'g4', 5:'a4', 6:'b4',
            7:'c8', 8:'d8', 9:'e8', 10:'f8', 11:'g8', 12:'a8', 13:'b8'}
```

RNNs

```
import numpy as np
```

Window_size = RNN 입력 개수

```
def seq2dataset(seq, window_size):
    dataset = []
    for i in range(len(seq)-window_size):
        subset = seq[i:(i+window_size+1)]
        dataset.append([code2idx[item] for item in subset])
    return np.array(dataset)
```

RNNs

실제 악보

```
seq = ['g8', 'e8', 'e4', 'f8', 'd8', 'd4', 'c8', 'd8', 'e8', 'f8', 'g8', 'g8', 'g4',
       'g8', 'e8', 'e8', 'e8', 'f8', 'd8', 'd4', 'c8', 'e8', 'g8', 'g8', 'e8', 'e8', 'e4',
       'd8', 'd8', 'd8', 'd8', 'd8', 'e8', 'f4', 'e8', 'e8', 'e8', 'e8', 'e8', 'f8', 'g4',
       'g8', 'e8', 'e4', 'f8', 'd8', 'd4', 'c8', 'e8', 'g8', 'g8', 'e8', 'e8', 'e4']

dataset = seq2dataset(seq, window_size = 4)
```

LSTM 입력 = 4

```
print(dataset.shape)
print(dataset)
```

RNNs

```
(50, 5)
[[11  9  2 10  8]
 [ 9  2 10  8  1]
 [ 2 10  8  1  7]
 [10  8  1  7  8]
 [ 8  1  7  8  9]
 [ 1  7  8  9 10]
 [ 7  8  9 10 11]
 [ 8  9 10 11 11]
 [ 9 10 11 11  4]
 [10 11 11  4 11]
```

```
# 0. 사용할 패키지 불러오기
import keras
import numpy as np
from keras.models import Sequential
from keras.layers import Dense, LSTM
from keras.utils import np_utils
```

```
# 2. 데이터셋 생성하기
dataset = seq2dataset(seq, window_size = 4)

print(dataset.shape)
```

```
# 입력(X)과 출력(Y) 변수로 분리하기
x_train = dataset[:,0:4]
y_train = dataset[:,4]

max_idx_value = 13

# 입력값 정규화 시키기
x_train = x_train / float(max_idx_value)

# 입력을 (샘플 수, 타임스텝, 특성 수)로 형태 변환
x_train = np.reshape(x_train, (50, 4, 1))

# 라벨값에 대한 one-hot 인코딩 수행
y_train = np_utils.to_categorical(y_train)

one_hot_vec_size = y_train.shape[1]

print("one hot encoding vector size is ", one_hot_vec_size)
```

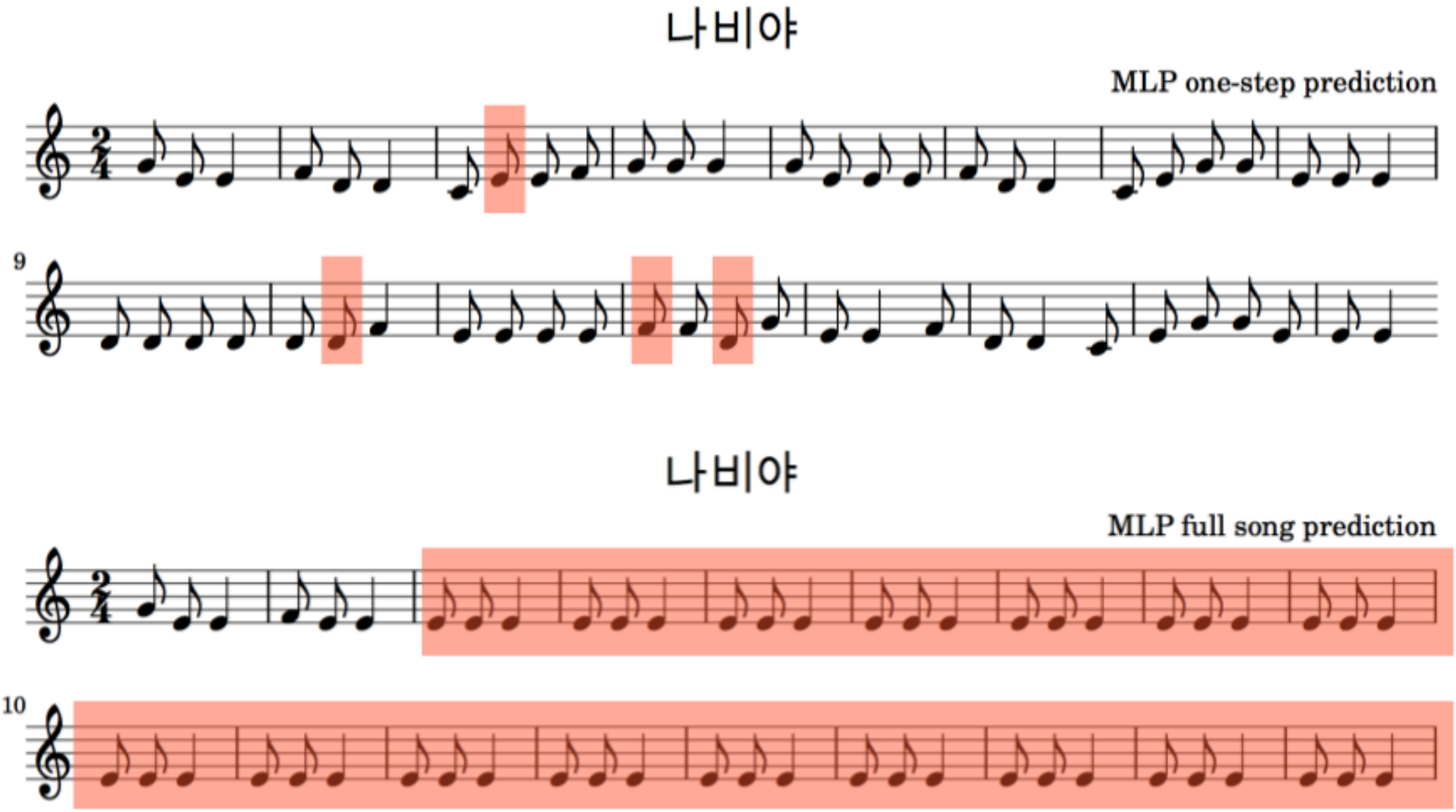
```
# 3. 모델 구성하기
model = Sequential()
model.add(LSTM(128, batch_input_shape = (1, 4, 1), stateful=True))
model.add(Dense(one_hot_vec_size, activation='softmax'))
```

```
# 4. 모델 학습과정 설정하기
model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])

# 5. 모델 학습시키기
num_epochs = 2000

history = LossHistory() # 손실 이력 객체 생성

history.init()

for epoch_idx in range(num_epochs):
    print ('epochs : ' + str(epoch_idx) )
    model.fit(x_train, y_train, epochs=1, batch_size=1, verbose=2, shuffle=False, callbacks=[history])
    model.reset_states()
```

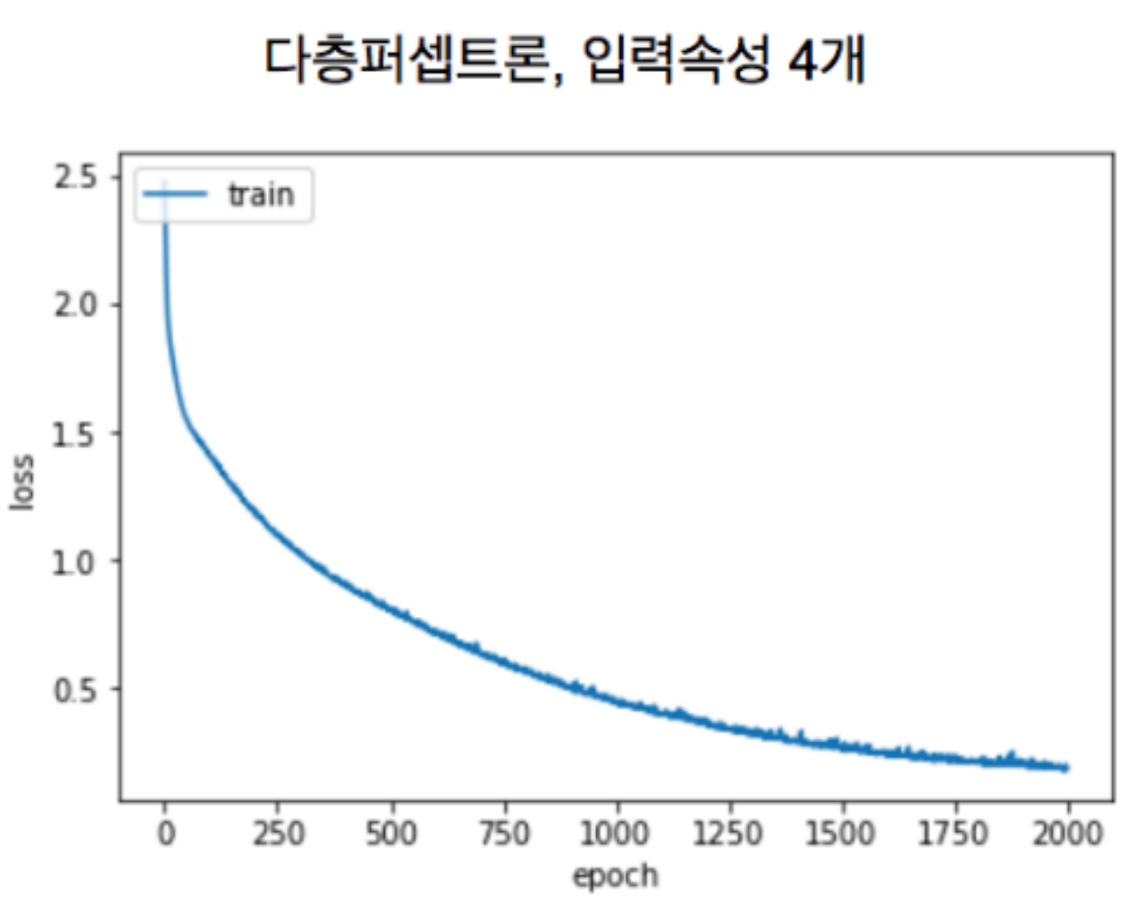
```
# 7. 모델 평가하기
scores = model.evaluate(x_train, y_train, batch_size=1)
print("%s: %.2f%%" %(model.metrics_names[1], scores[1]*100))
model.reset_states()

# 8. 모델 사용하기

pred_count = 50 # 최대 예측 개수 정의

# 곡 전체 예측

seq_in = ['g8', 'e8', 'e4', 'f8']
seq_out = seq_in
seq_in = [code2idx[it] / float(max_idx_value) for it in seq_in] # 코드를 인덱스값으로 변환

for i in range(pred_count):
    sample_in = np.array(seq_in)
    sample_in = np.reshape(sample_in, (1, 4, 1)) # 샘플 수, 타입스텝 수, 속성 수
    pred_out = model.predict(sample_in)
    idx = np.argmax(pred_out)
    seq_out.append(idx2code[idx])
    seq_in.append(idx / float(max_idx_value))
    seq_in.pop(0)

model.reset_states()

print("full song prediction : ", seq_out)
```

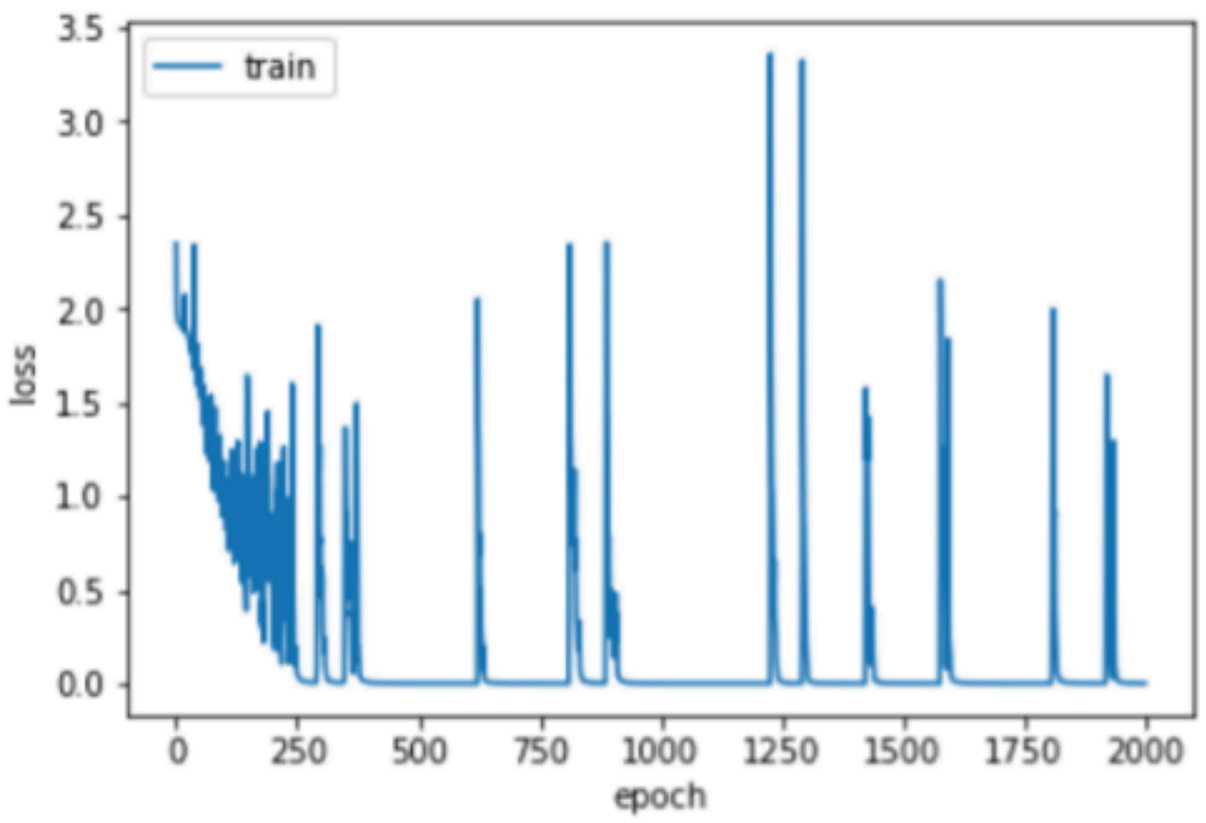
```
# 3. 모델 구성하기
model = Sequential()
model.add(Dense(128, input_dim=4, activation='relu'))
model.add(Dense(128, activation='relu'))
model.add(Dense(one_hot_vec_size, activation='softmax'))
```

Dense Layer


나비야
MLP one-step prediction
9
나비야
MLP full song prediction
10

Dense Layer


다층퍼셉트론, 입력속성 4개
train
loss
epoch
2.5
2.0
1.5
1.0
0.5
0
250
500
750
1000
1250
1500
1750
2000


상태유지 LSTM, 입력속성 1개
train
loss
epoch
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
0
250
500
750
1000
1250
1500
1750
2000