Recurrent Neural Network (RNN)

순환 신경망

등장배경

- 자기 종속성
  - 이전 타임스텝의 출력이 다음 타임스텝의 입력에 영향을 준다는 속성
  - RNN은 이런 자기 종속성을 다층 퍼셉트론으로 모델링한 구조
- 음악, 동영상, 에세이, 시, 소스 코드, 주가 차트
  - 시간적인 순서를 갖는 시퀀스
  - 시퀀스의 길이는 가변적
- RNN(Recurrent Neural Network, 순환신경망)은 시퀀스 데이터를 모델링 하기 위해 등장

RNN 예시

시계열 예측 (eg. 주가 예측)

- 주가예측의 경우 이전 시간의 주가가 다음 시간의 주가에 영향을 주기 때문에 Auto regressive model인 RNN을 사용해서 문제 풀기 가능

RNN 예시

자연어 처리

- 구글의 번역기와 네이버의 파파고와 같은 자연어처리 시스템들은 RNN을 응용한 모델
- RNN 기반 모델은 기존 통계 기반 모델의 비해 우수한 성능

음성인식

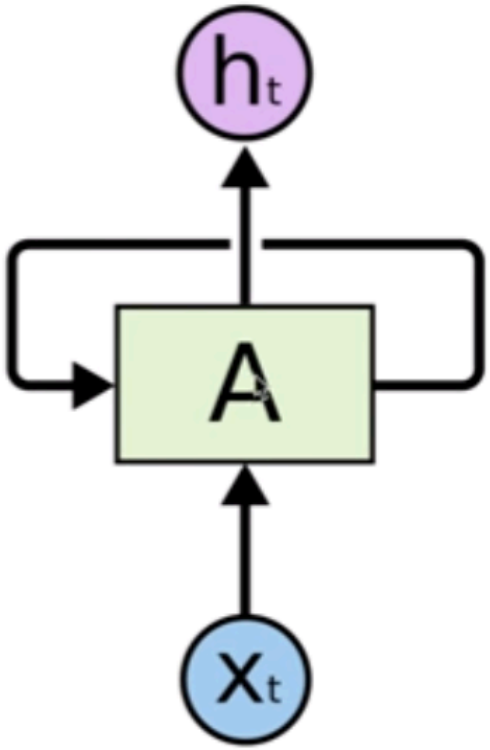
RNN

연속적인 데이터 (Sequence Data)

- 한개의 단어로는 Sequece 데이터를 이해할 수 없음
- 현재의 단어와 이전 단어의 조합으로 전체적인 문장의 흐름을 이해할 수 있음
- NN/CNN은 이전 상태에 대한 내용이 없음

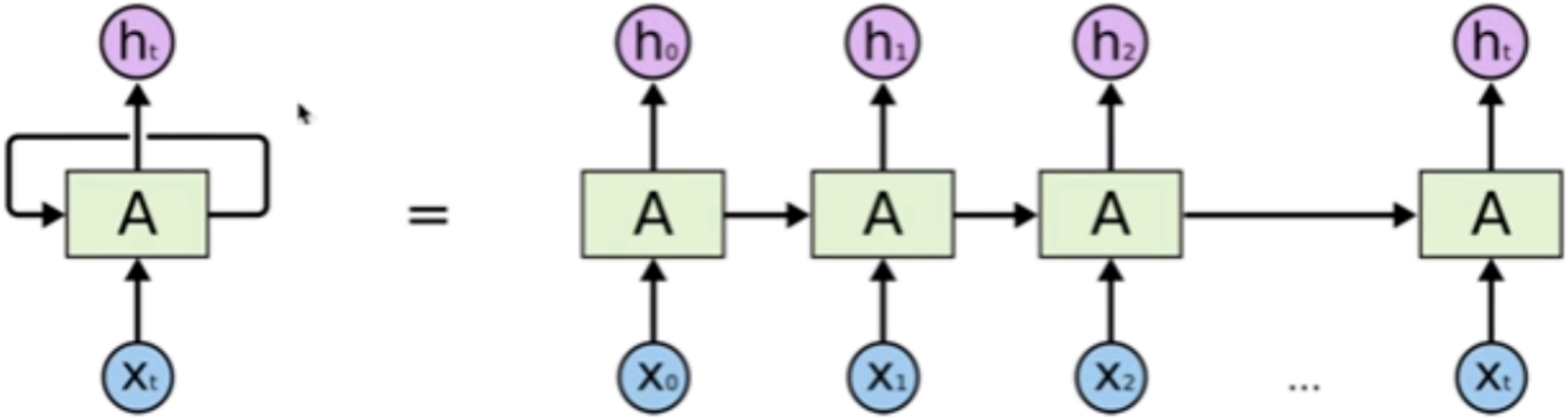
RNN은 sequence data를 처리하는 모델

- 자연어의 경우 단어 하나만 안다고 해서 처리될 수 없고, 앞뒤 문맥을 함께 이해해야 해석이 가능



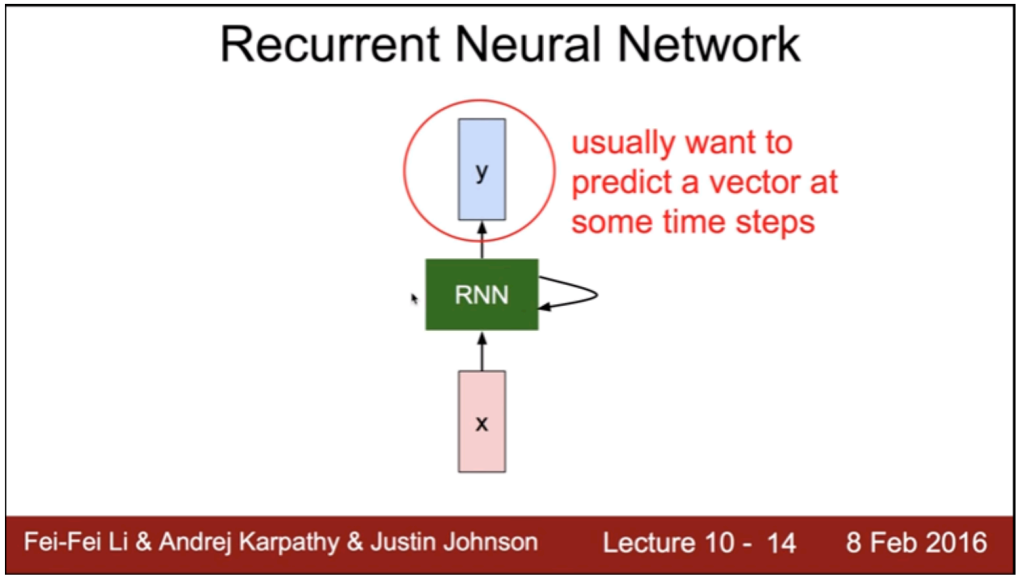
Recurrent Neural Network

- 아래 그림에서 보면 x_t 가 입력되어서 히든레이어 A를 거쳐서 최종적으로 h_t라는 출력을 반환하게 됩니다.
- 그러나 RNN의 특징은 저기에 있는 순환 구조의 화살표에 있음
- 화살표는 히든레이어인 A에서 나와서 다시 히든레이어 A로 입력

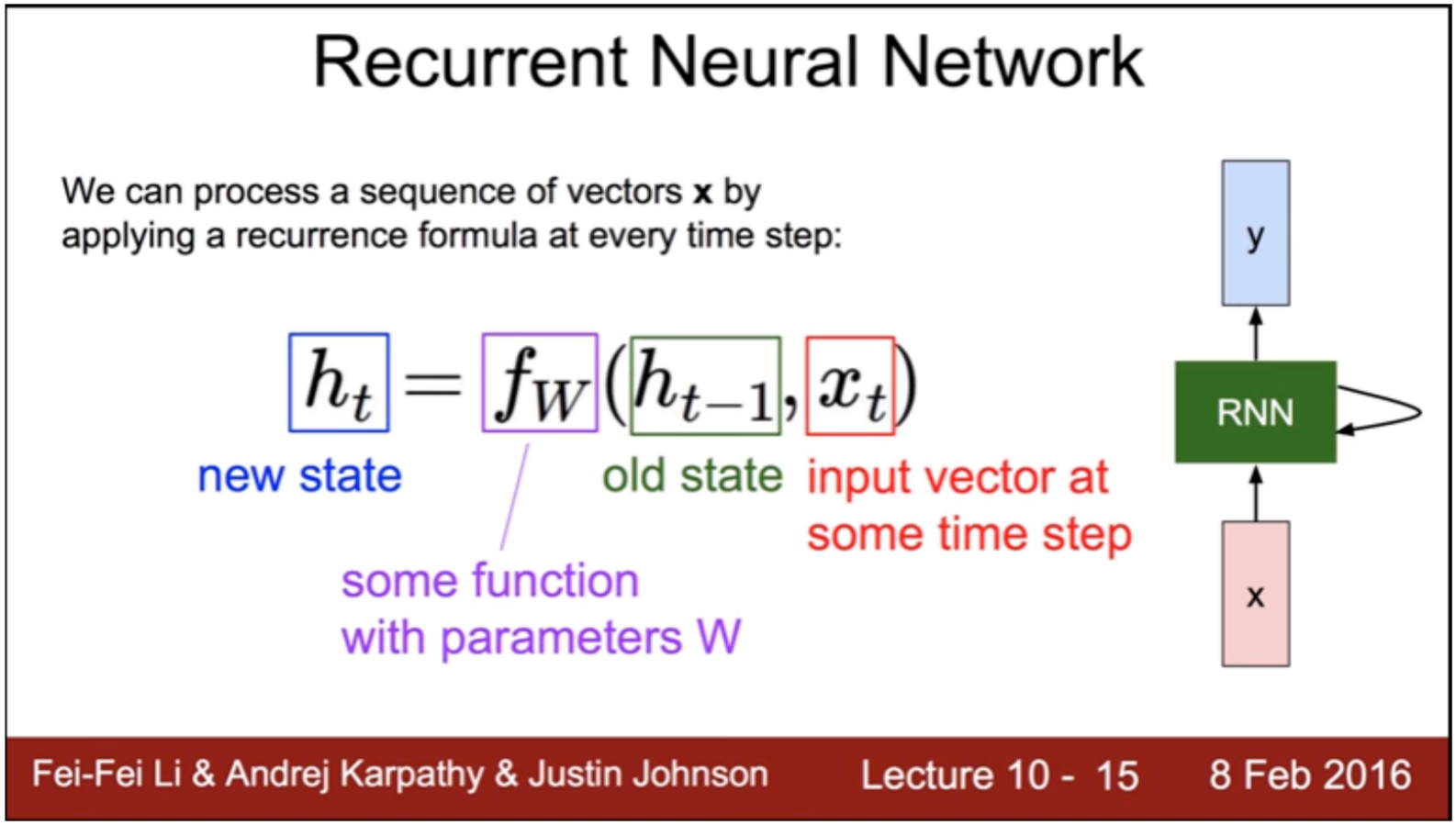


Recurrent Neural Network

- RNN(Recurrent Neural Network)은 일반적으로 step을 거칠 때마다 어떤 결과를 예측
- 예측 값을 앞에서 배웠던 것처럼 y라고 부릅니다. y = Wx + b와 같이 생각하시면 됩니다. 그리고 RNN에서는 y 대신에 h를 자주 사용

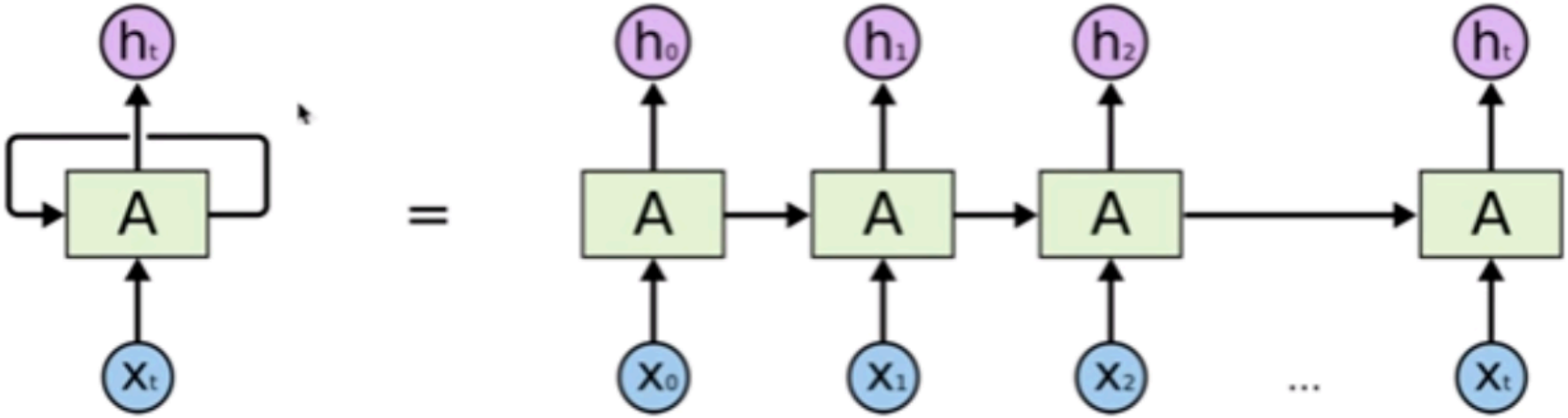


Recurrent Neural Network



Recurrent Neural Network

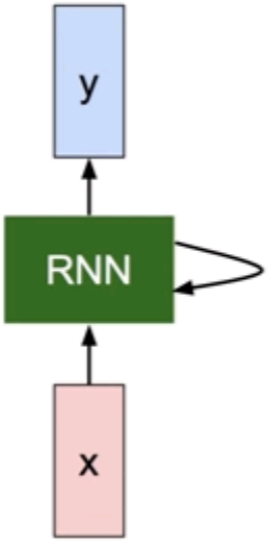
- 새로운 출력인 h_t는 이전의 출력인 h_{t-1}과 새로운 입력 x_t이 입력되어 나온 출력
- 노드 갯수는 layer에 포함된 노드(그림에서는 초록색으로 표시된 RNN) 갯수



Recurrent Neural Network

(Vanilla) Recurrent Neural Network

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



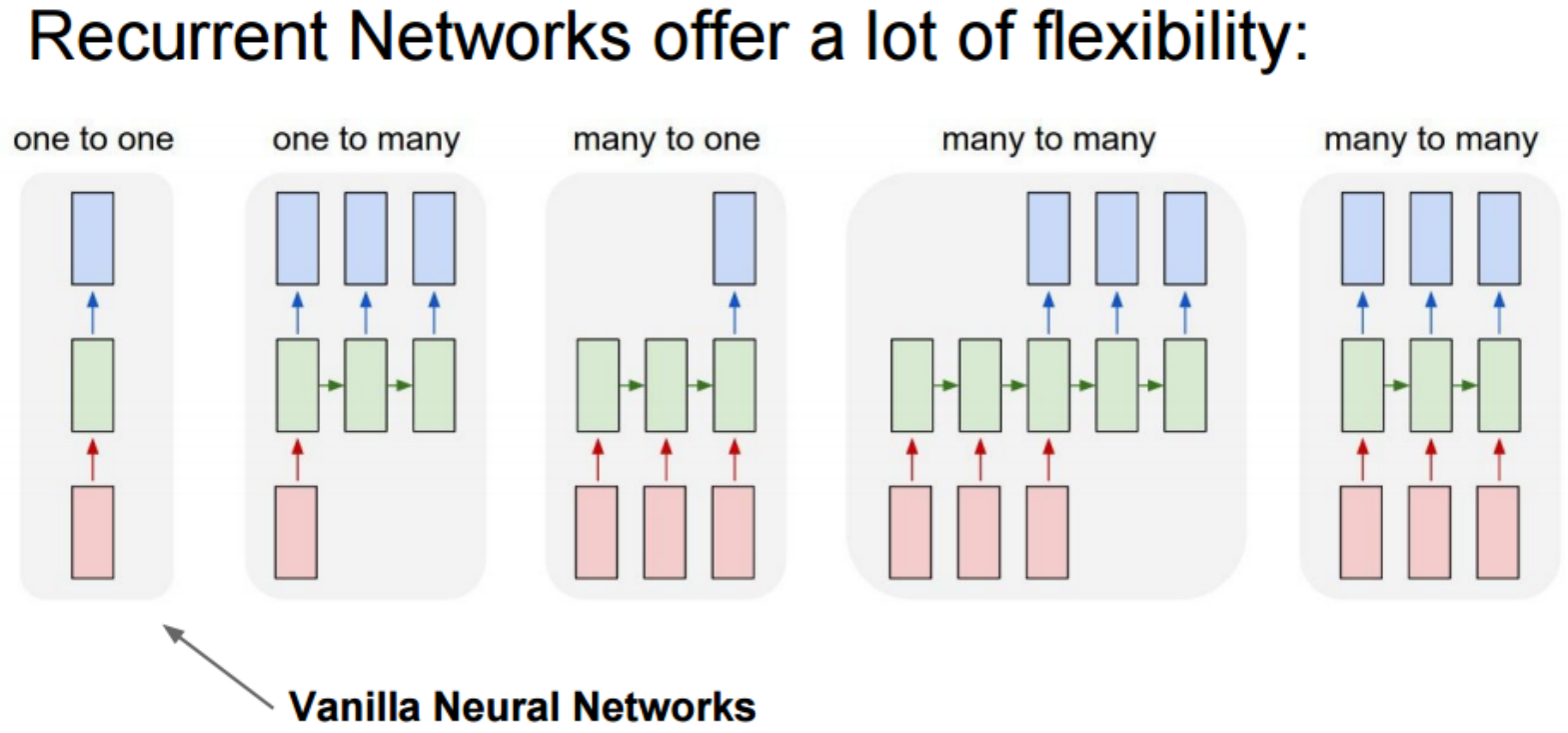
$$h_t = f_W(h_{t-1}, x_t)$$

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

$$y_t = W_{hy}h_t$$

Fei-Fei Li & Andrej Karpathy & Justin Johnson Lecture 10 - 17 8 Feb 2016

다양한 RNN 형태

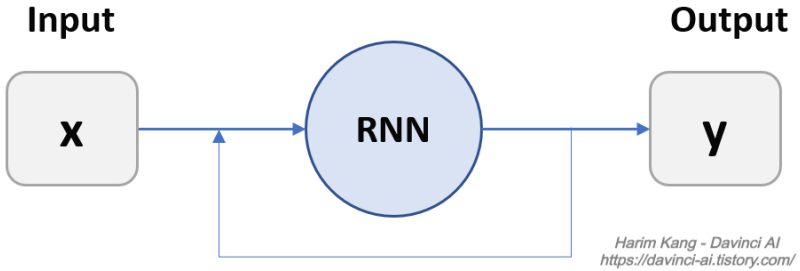

Recurrent Networks offer a lot of flexibility:
one to one
one to many
many to one
many to many
many to many
Vanilla Neural Networks

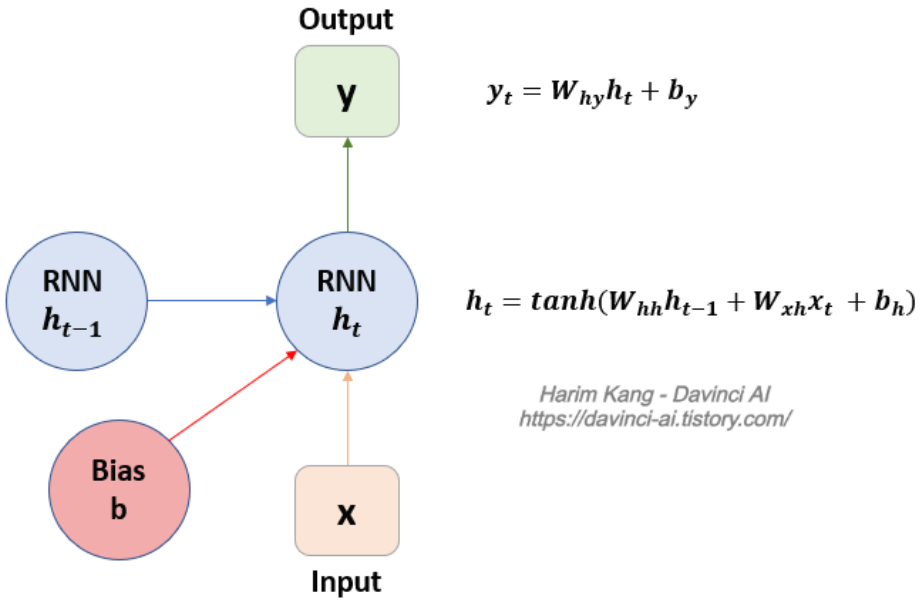
RNN Application

- https://github.com/TensorFlowKR/awesome_tensorflow_implementations
  - Language Model
  - Speech Recognition
  - Machine Translation
  - Conversation Model /Question Answering
  - Image /Vide Captioning

Vanilla RNN (Keras.SimpleRNN)

- 레이어의 출력을 다시 입력으로 받아서 사용하는 것으로서, 이전의 데이터가 함께 결과에 영향을 미침

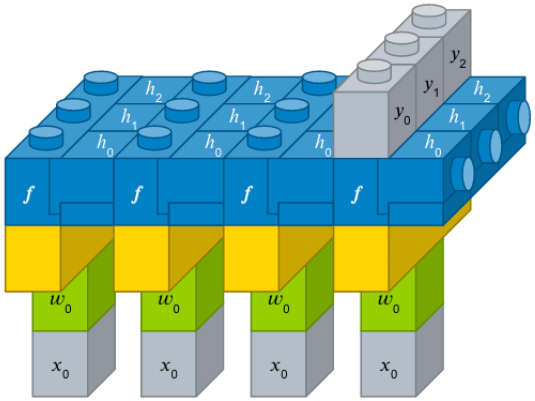




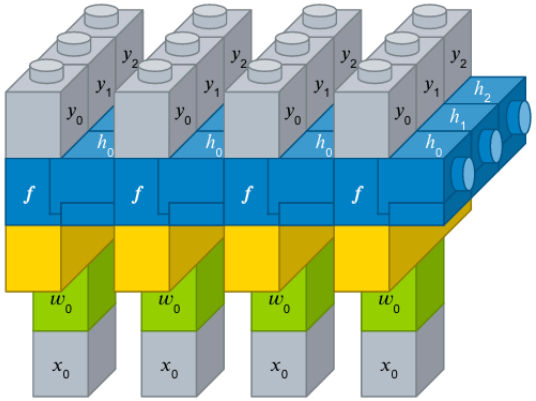
Vanilla RNN (Keras.SimpleRNN)

return_sequences

- RNN 계산 과정에 있는 hidden state를 출력할 것인지에 대한 값
  RNN 또는 one-to-many, many-to-many 출력을 위해서 사용



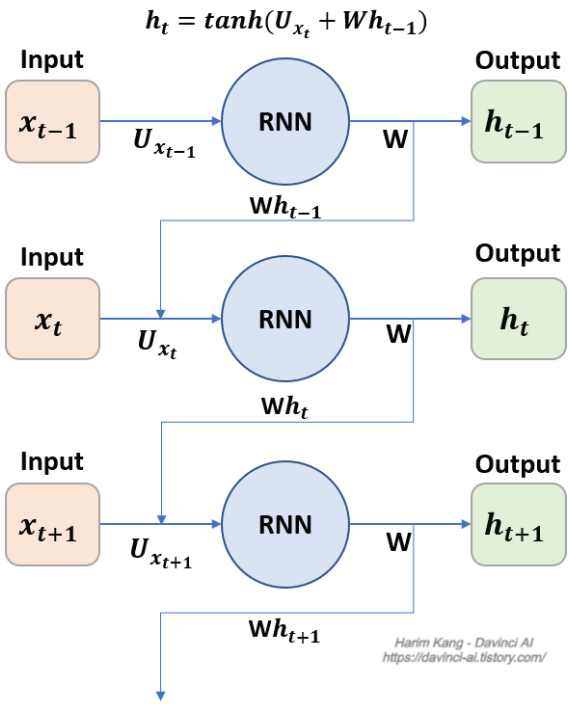
return_sequences=False



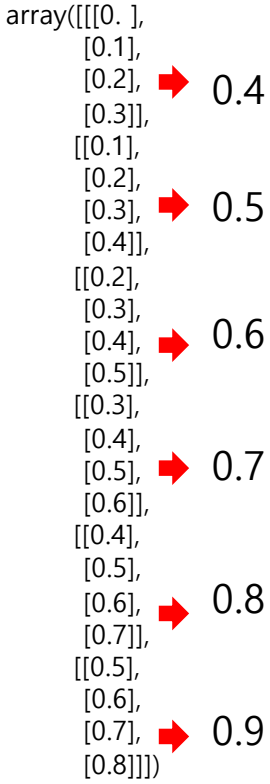
return_sequences= True

Vanilla RNN (Keras.SimpleRNN)

가장 간단한 형태의 RNN Layer



Vanilla RNN (Keras.SimpleRNN)




```
X = []
Y = []
for i in range(6):
    lst = list(range(i,i+4))
    X.append(list(map(lambda c: [c/10], lst)))
    Y.append((i+4)/10)
X = np.array(X)
Y = np.array(Y)
print(X)
print(Y)
```

Vanilla RNN (Keras.SimpleRNN)

```
model = Sequential()
model.add(SimpleRNN(50, return_sequences=False, inp
ut_shape=(4,1)))
model.add(Dense(1))
model.summary()
model.compile(loss='mse',
              optimizer='adam',
              metrics=['accuracy'])
model.fit(X,Y,epochs=200, verbose=2)
print(model.predict(X))

X_test = np.array([[[0.8],[0.9],[1.0],[1.1]]])
print(model.predict(X_test))
```