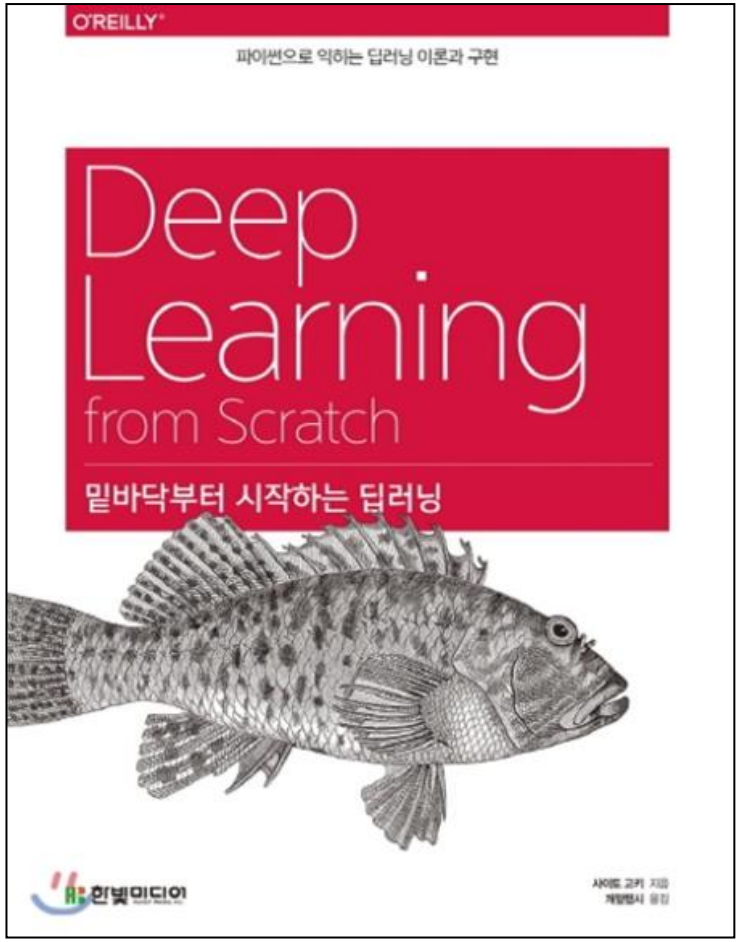
RNN, LSTM
1. 소개
2. 구현

<활용위주>

케라스 사용, 이론설명은 부족
요즘은 파이토치가 대세다.

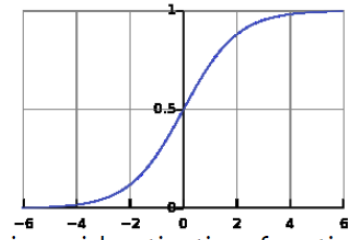


<기초, 이론위주, 1~2편>

기본원리부터 차근차근 코딩

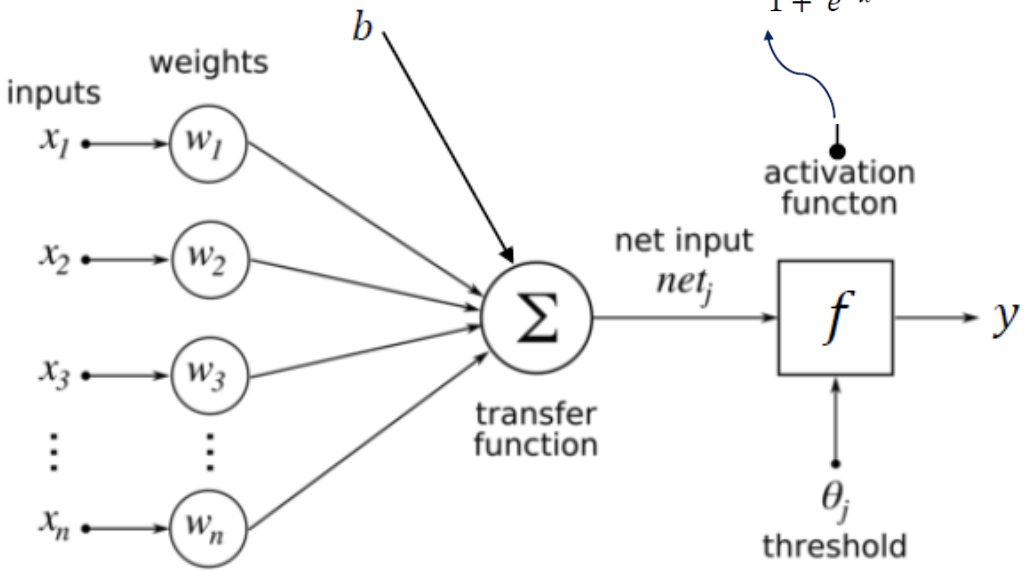
- Ref.
- 상기의 도서
- 모두를 위한 딥러닝 강좌 시즌 1
- 모두를 위한 딥러닝 강좌 시즌 2
- https://github.com/jonghkim/financial-time-series-prediction-v2

- Perceptron
- NN의 최소단위



sigmoid activation function

$$f(x) = \frac{1}{1 + e^{-x}}$$



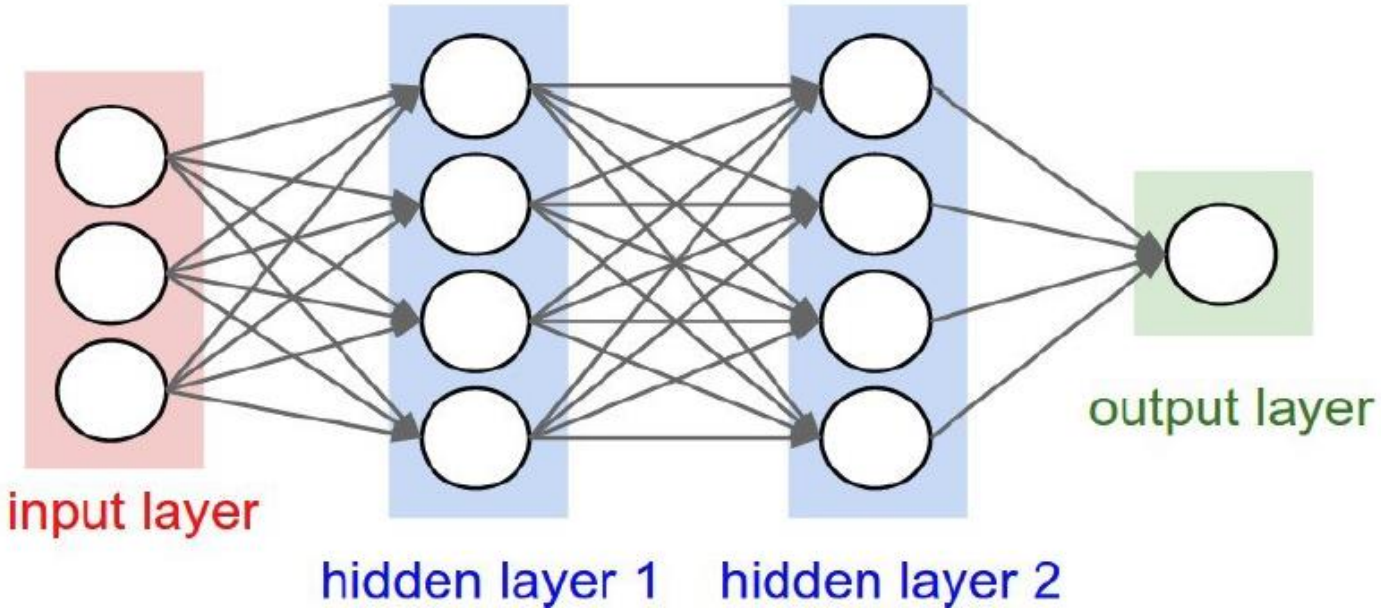
<Perceptron>

$$y = f(\mathbf{wx} + \mathrm{b})$$

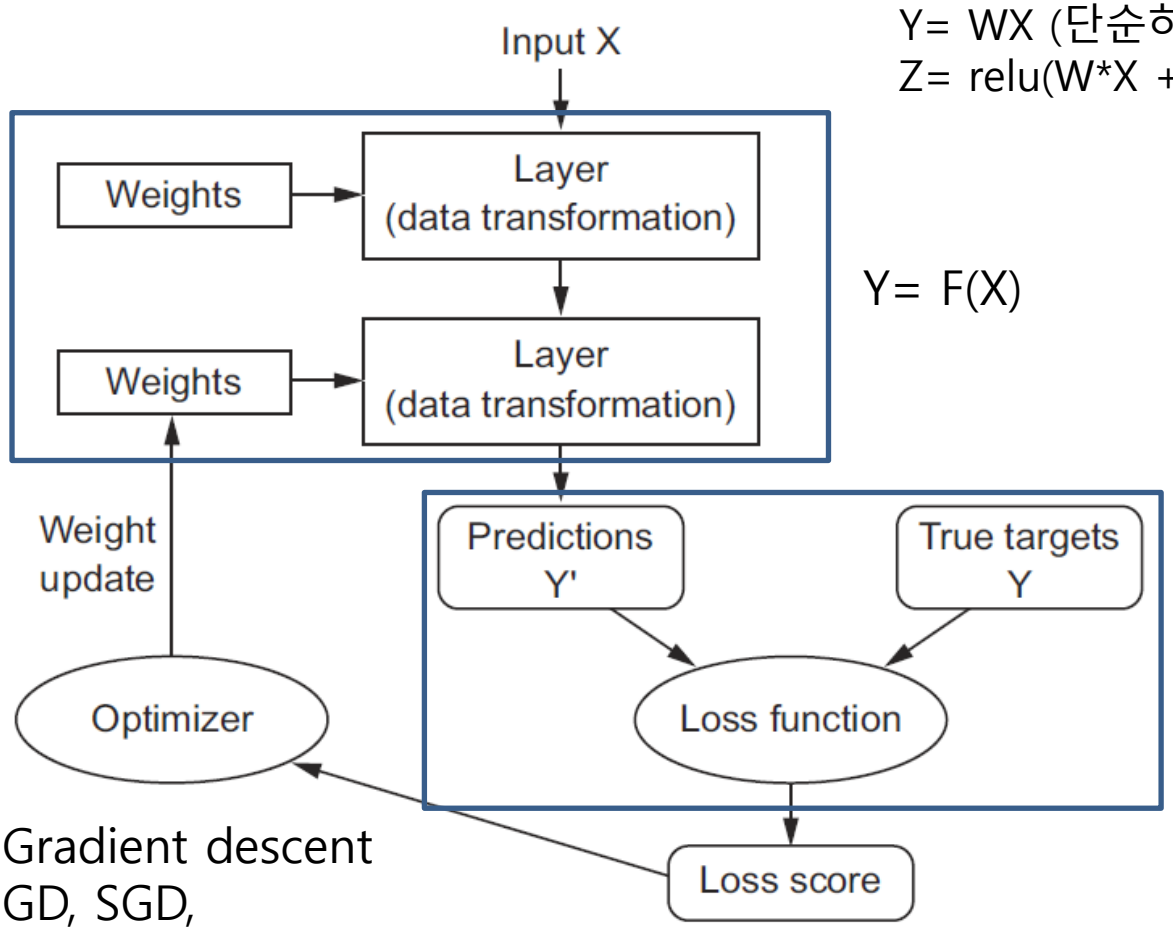
$$\boldsymbol{w} = [w_1\ w_2\ w_3\ \ldots\ w_n]$$

$$\boldsymbol{x} = [x_1\ x_2\ x_3\ \ldots\ x_n]^T$$

- DNN
- = Fully connected NN
- = Simple NN



Dataset -> X(입력값), Y(타겟, 목표값)
X와 Y의 관계를 파악하는 것이 목적
Y= F(X)
Y= WX (단순하게)
Z= relu(W*X +B) -> Y= relu(W1*Z+B1)



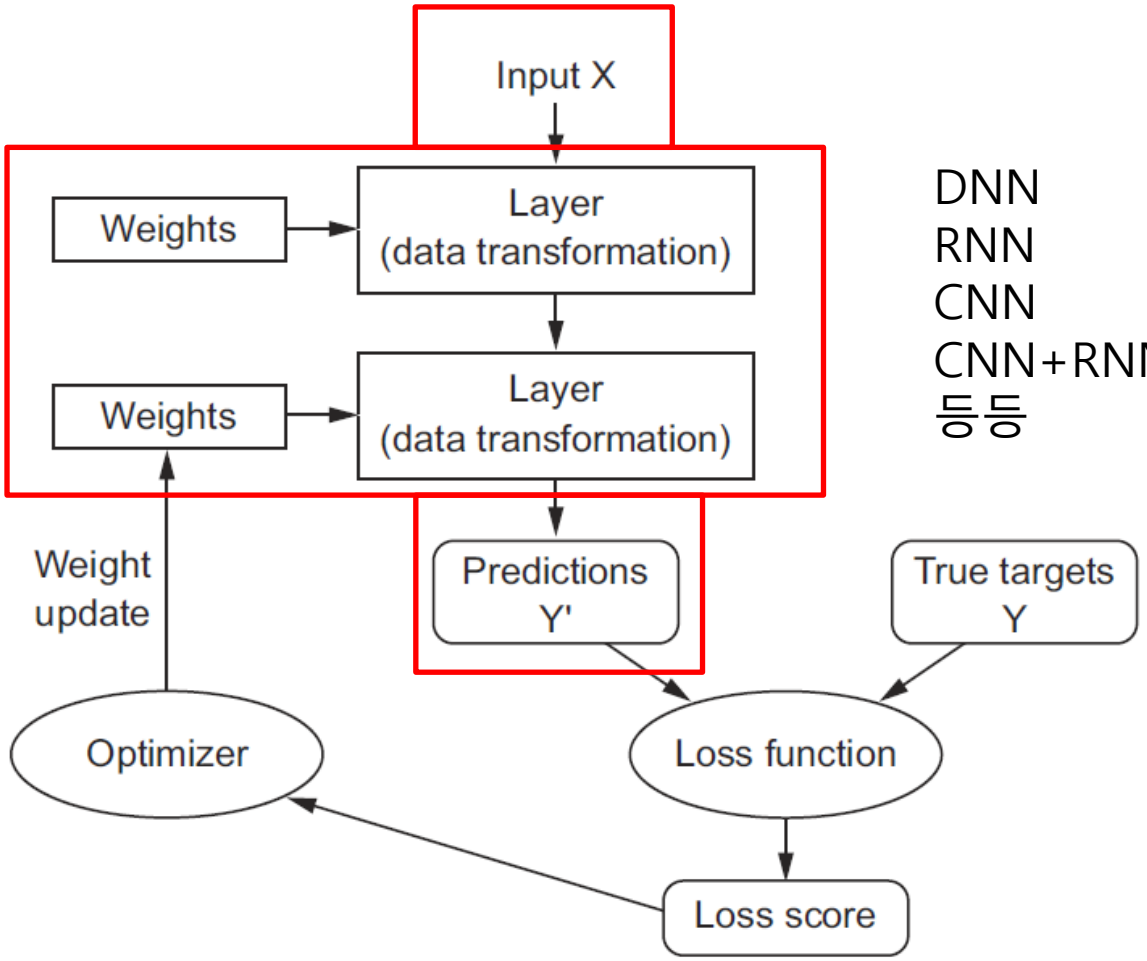
Y= F(X)

L= (f(X) – Y)^2 -> mean
mse, cross-entropy

Gradient descent
GD, SGD,

W1=W0- a* dL/dW

손실값이 최소화



DNN
RNN
CNN
CNN+RNN
등등

- RNN (Recurrent NN)
- 순환신경망
- 개선: LSTM, GRU

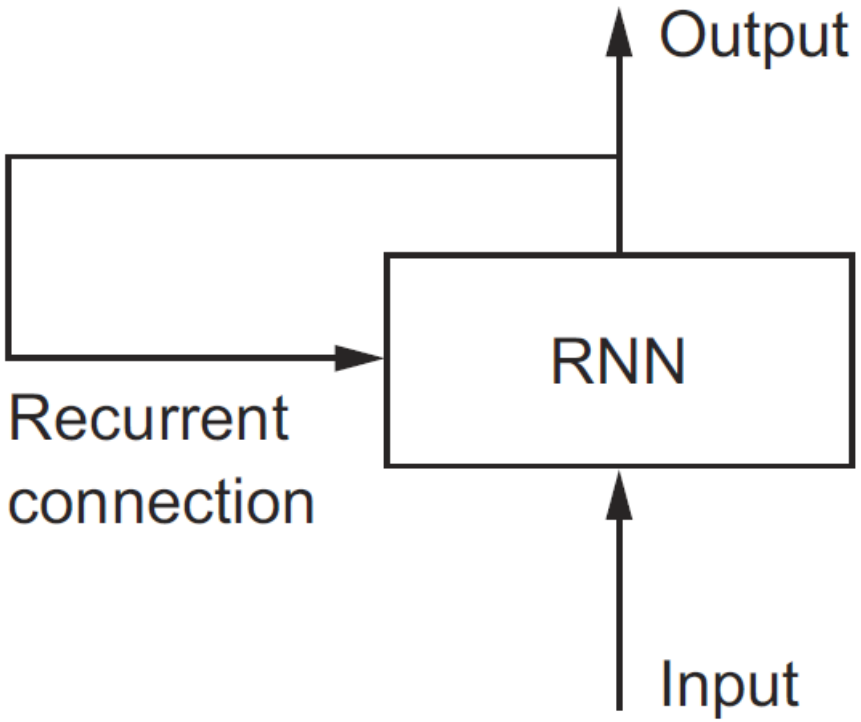


Figure 6.9 A recurrent network: a network with a loop

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

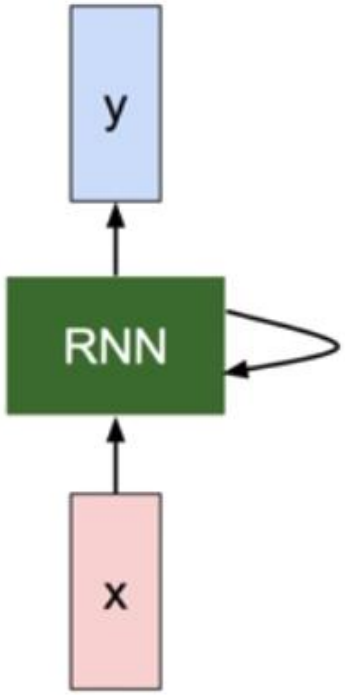
$$h_t = f_W(h_{t-1}, x_t)$$

new state

some function with parameters W

old state

input vector at some time step



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

$$y_t = W_{hy}h_t$$

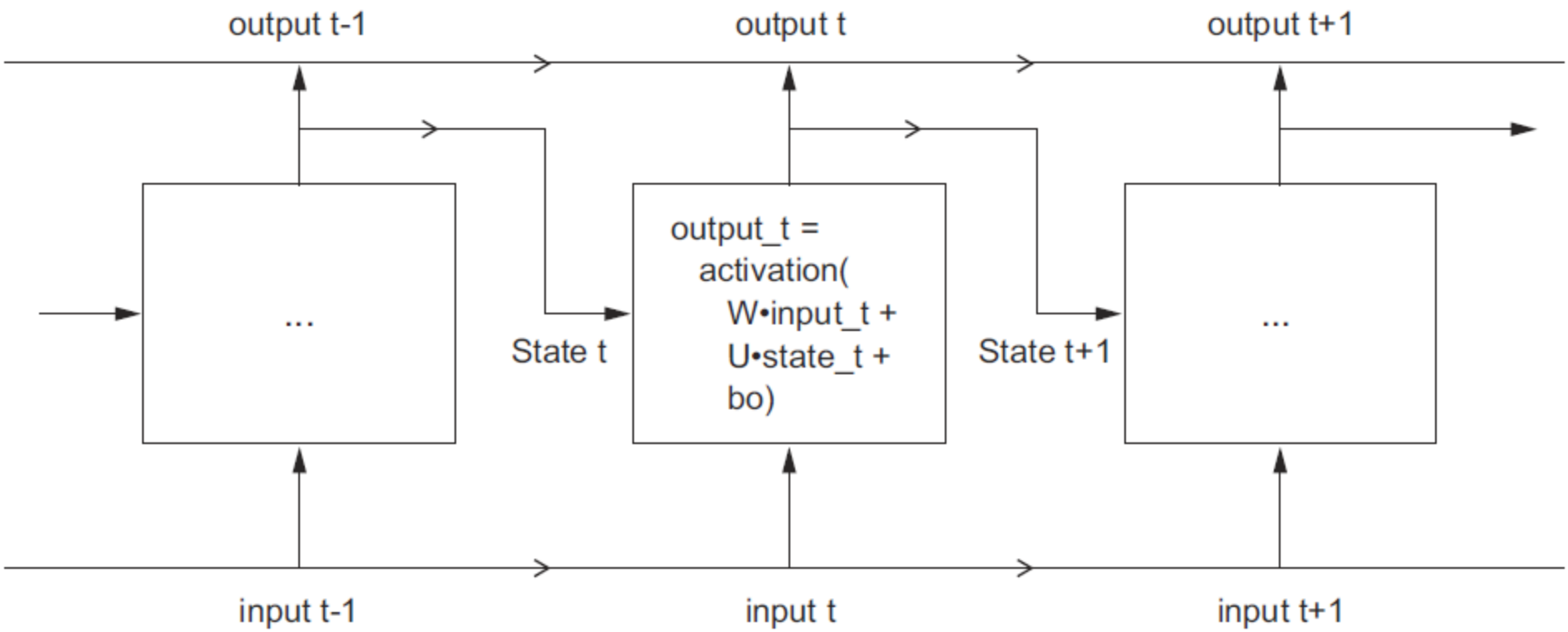


Figure 6.10 A simple RNN, unrolled over time

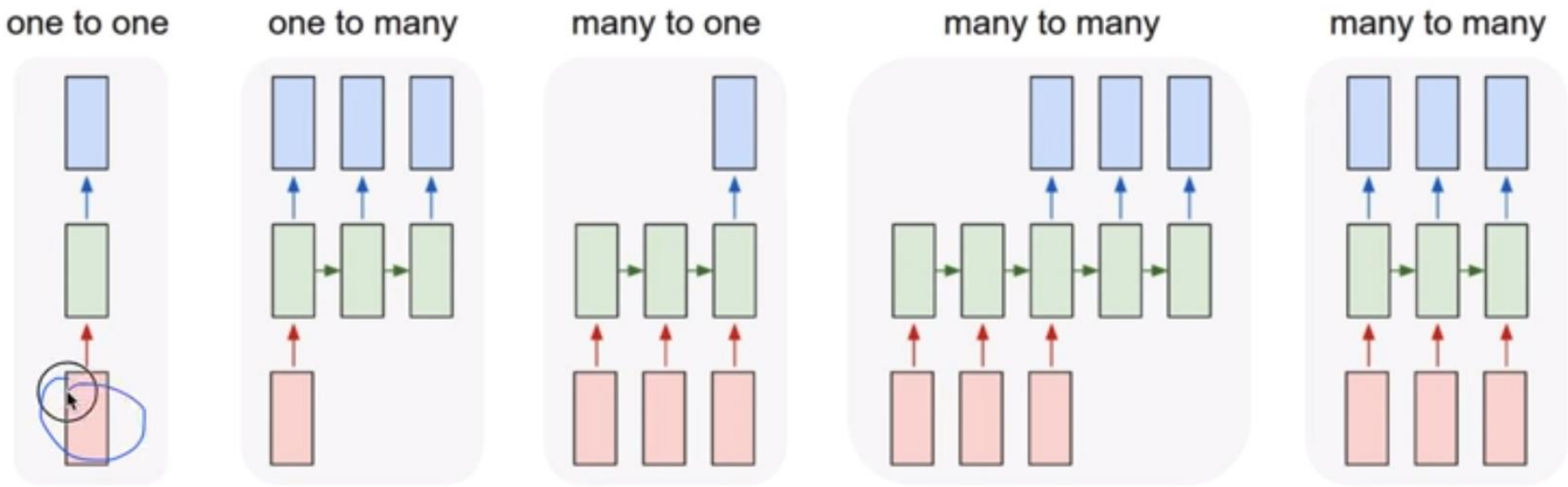
Listing 6.19 Pseudocode RNN

```
state_t = 0                          <-- The state at t
for input_t in input_sequence:       <-- Iterates over sequence elements
    output_t = f(input_t, state_t)
    state_t = output_t               <-- The previous output becomes the state for the next iteration.
```

Listing 6.20 More detailed pseudocode for the RNN

```
state_t = 0
for input_t in input_sequence:
    output_t = activation(dot(W, input_t) + dot(U, state_t) + b)
    state_t = output_t
```

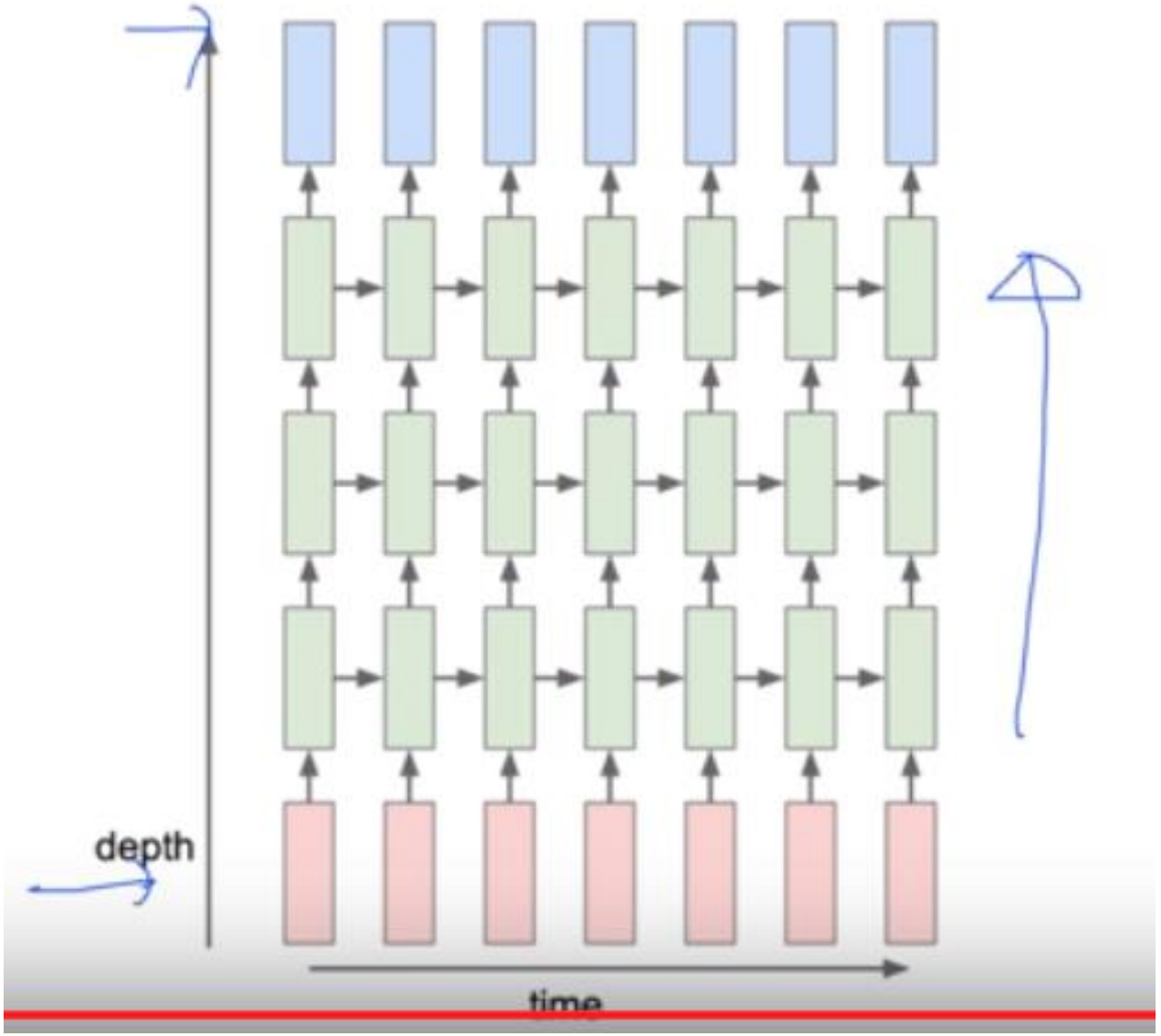
Recurrent Networks offer a lot of flexibility:

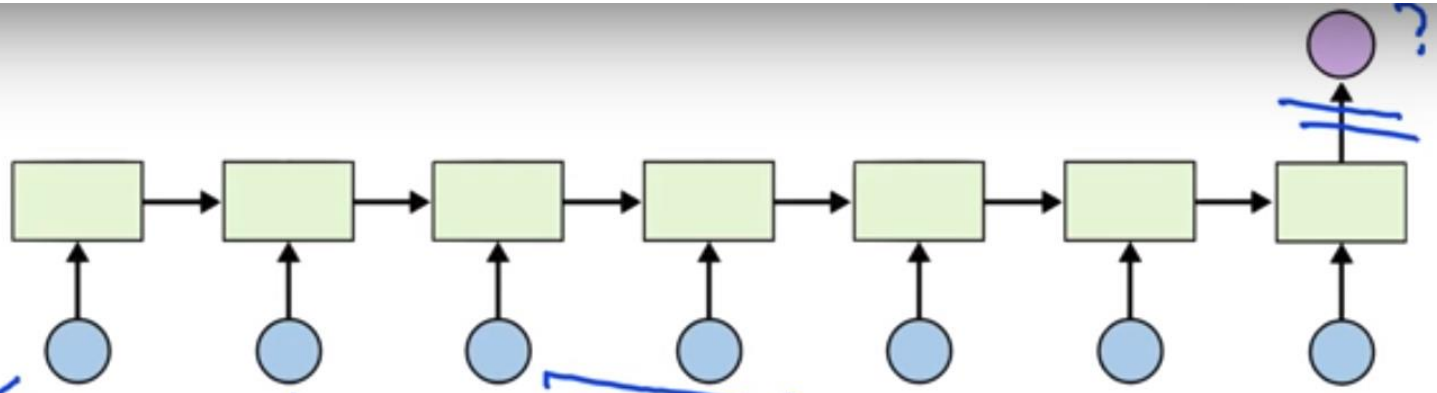


Vanilla Neural Networks

1) vanilla
2) Image Captioning(image -> sequence of words)
3) Sentiment Classification (seq of words -> sentiment)
4) Machine Translation (seq of words -> seq of words)
5) Video Classification on frame level

Multi-Layer RNN





Open	High	Low	Volume	Close
828.659973	833.450012	828.349976	1247700	**831.659973**
823.02002	828.070007	821.655029	1597800	**828.070007**
819.929993	824.400024	818.97998	1281700	**824.159973**
819.359985	823	818.469971	1304000	**818.97998**
819	823	816	1053600	**820.450012**
816	820.958984	815.48999	1198100	**819.23999**
811.700012	815.25	809.780029	1129100	**813.669983**
809.51001	810.659973	804.539978	989700	**?**

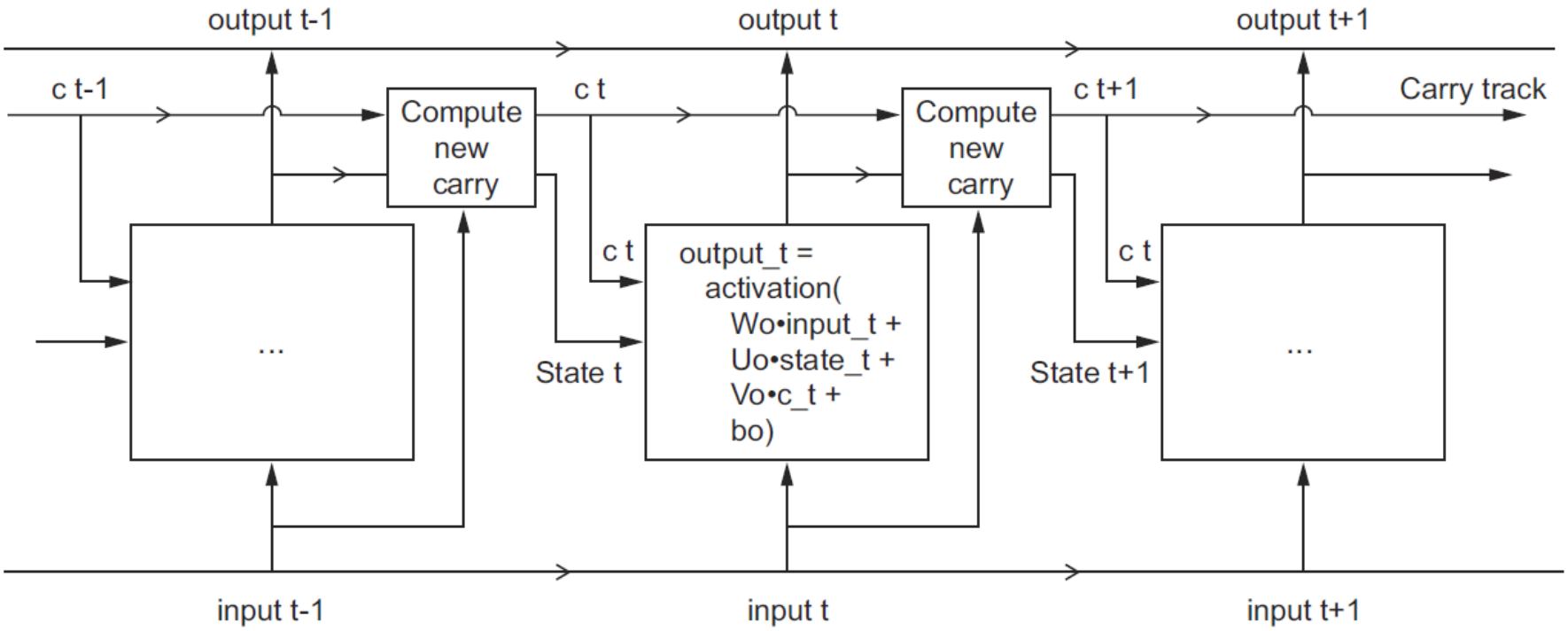


Figure 6.15 Anatomy of an LSTM

Listing 6.25 Pseudocode details of the LSTM architecture (1/2)

```
output_t = activation(dot(state_t, Uo) + dot(input_t, Wo) + dot(C_t, Vo) + bo)

i_t = activation(dot(state_t, Ui) + dot(input_t, Wi) + bi)
f_t = activation(dot(state_t, Uf) + dot(input_t, Wf) + bf)
k_t = activation(dot(state_t, Uk) + dot(input_t, Wk) + bk)
```

You obtain the new carry state (the next `c_t`) by combining `i_t`, `f_t`, and `k_t`.

Listing 6.26 Pseudocode details of the LSTM architecture (2/2)

```
c_t+1 = i_t * k_t + c_t * f_t
```

예제 실행

- https://github.com/jonghkim/financial-time-series-prediction-v2

- 구글 코랩 사용, 소스코드 가져오기
- RNN사용한 MNIST 분류
- LSTM사용한 비트코인 상승/하락 예측