날씨 데이터를 활용한 태양광 발전량 예측

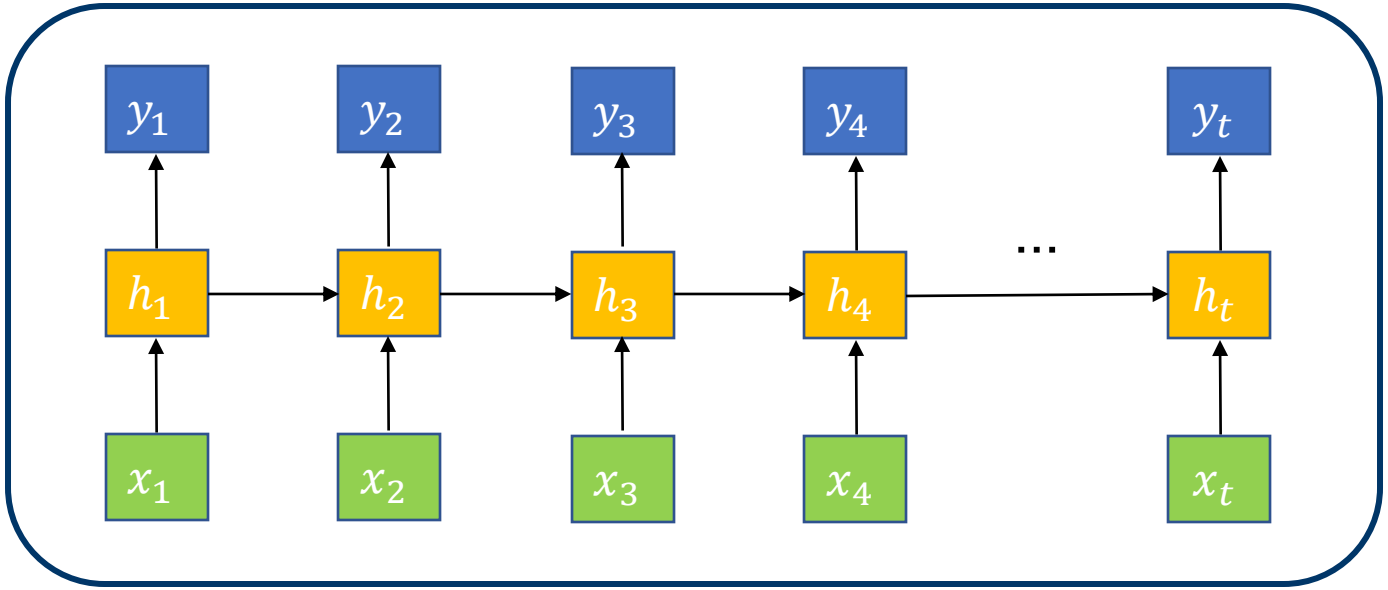
7강. 태양광 발전량 예측 모델 구현

1. RNN(Recurrent Neural Network)
2. 깊은 순환 신경망(Deep RNN)
3. 양방향 순환 신경망(Bidirectional RNN)
4. LSTM(Long Short-Term Memory)
5. GRU(Gated Recurrent Unit)

1 RNN(Recurrent Neural Network)

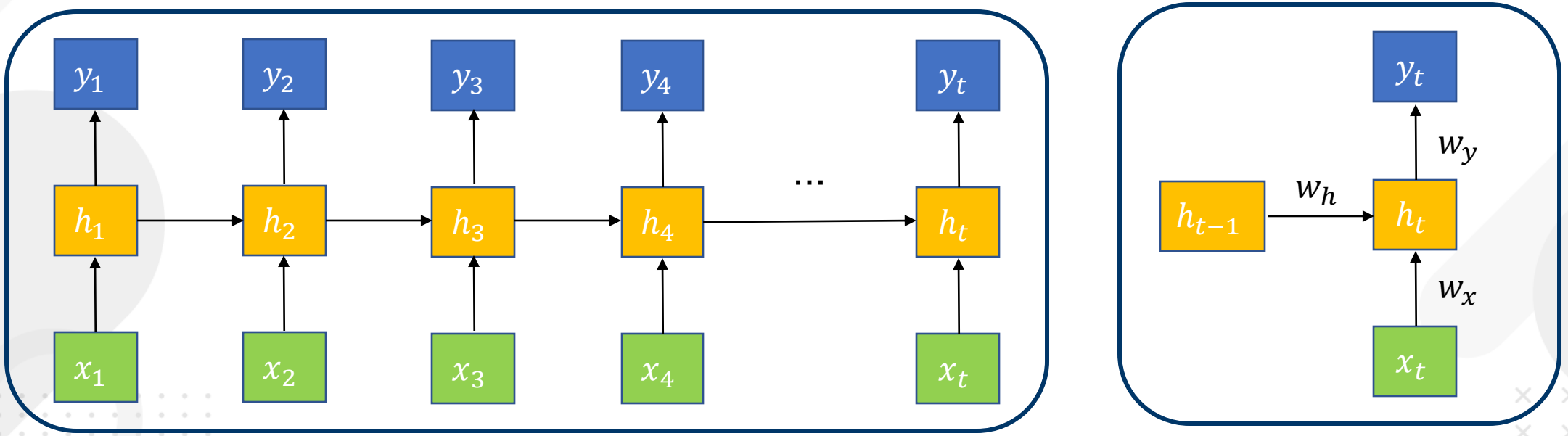
기상청 Korea Meteorological Administration

- RNN(Recurrent Neural Network)은 입력과 출력을 순차적으로 처리하는 시퀀스(Sequence) 모델입니다.
- 한 시간 단위의 날씨 데이터를 순차적으로 입력해주면 이에 대한 예측치를 순차적으로 출력해줄 수 있는 모델입니다.
- 이처럼 시계열 데이터를 처리하기 위해 고안된 모델들을 시퀀스 모델이라고 합니다.
- 그 중에서 RNN은 시계열에서 가장 기본적인 신경망 모델입니다.
- 심층 신경망의 기본 모델은 입력층 -> 은닉층 -> 출력층 방향으로만 이동하지만, RNN은 은닉층에서 나온 값을 출력층과 은닉층의 다음 노드로 보냅니다.



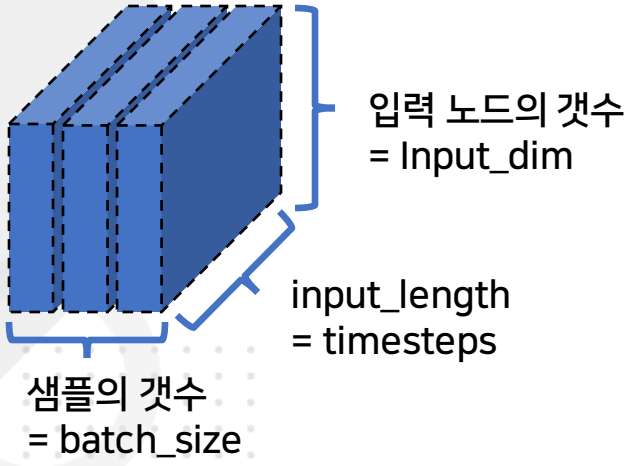
1 RNN(Recurrent Neural Network)

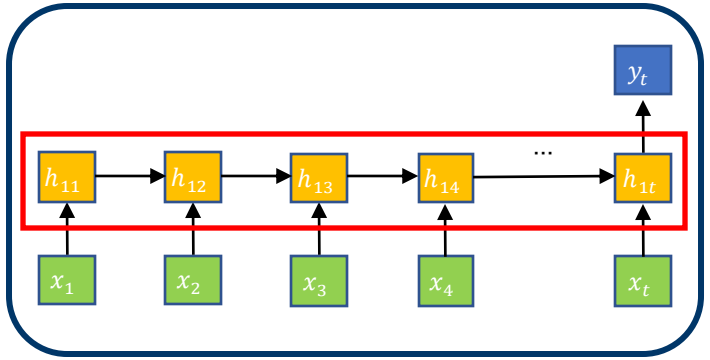
- 이 셀은 이전의 값을 기억하는 메모리 역할을 수행하므로 이를 메모리 셀(또는 RNN 셀)이라고 합니다.
- 메모리 셀은 현재 시점(h_t)에서 바로 이전 시점의 출력 값($h\_(t-1)$)을 자신의 입력으로 사용합니다.
- 현재 시점 t에서의 메모리 셀이 갖는 값은 과거의 메모리 셀들의 값에 영향을 받습니다.



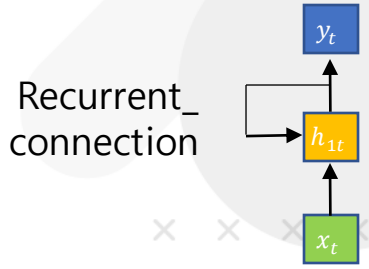
- keras로 RNN은 아래와 같이 구현할 수 있습니다.

```
from tensorflow.keras.layers import Dense, SimpleRNN

# timesteps,  input_dim
print(X_train.shape[-2:])

h_units = 100

model = models.Sequential()
model.add(SimpleRNN(h_units, recurrent_dropout=0.5, kernel_initializer='he_normal', input_shape=X_train.shape[-2:],))
#model.add(Dropout(0.5))
model.add(Dense(1))

model.compile(loss='mse', metrics=['mae'], optimizer=Adam(learning_rate = 0.0006))
```





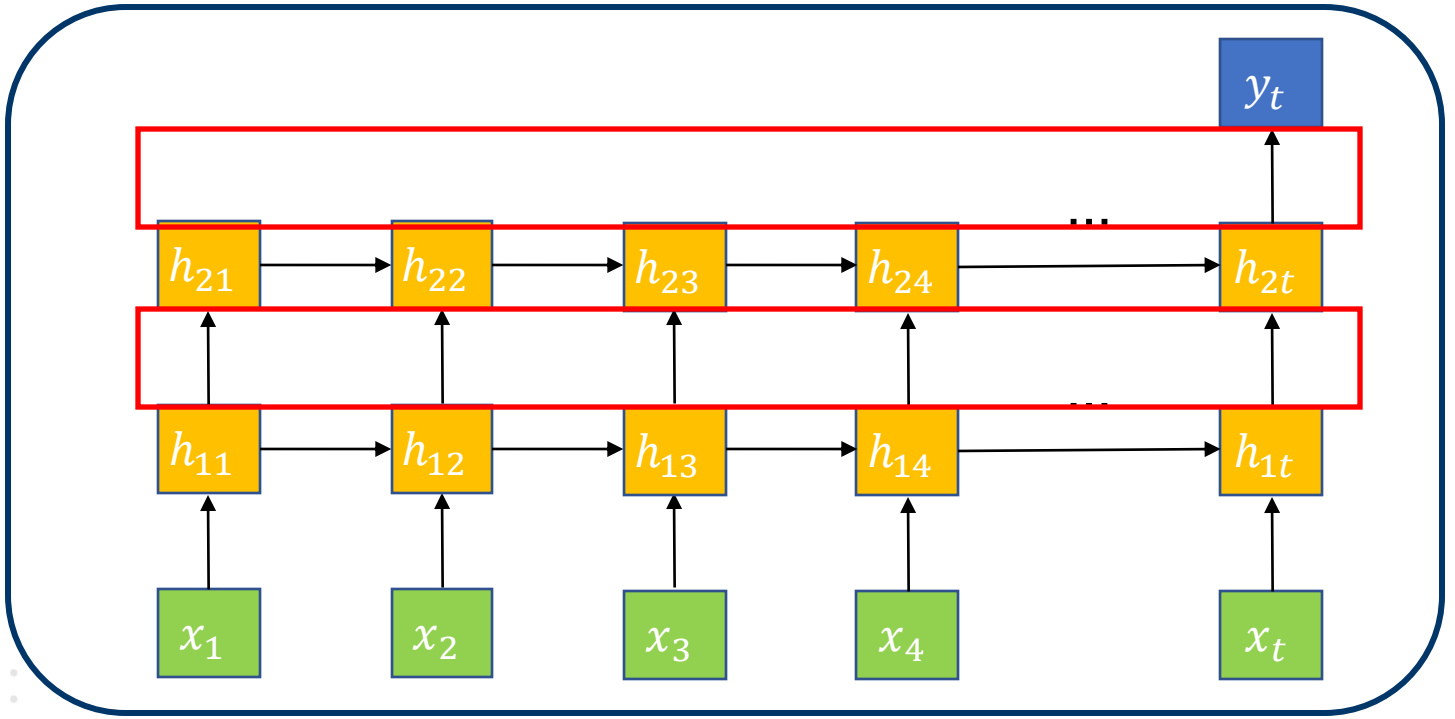
h_units = 100



2 깊은 순환 신경망(Deep RNN)

기상청 Korea Meteorological Administration

- RNN도 은닉층(Hidden Layer)를 여러 개 갖을 수 있습니다. 아래 그림은 2개의 은닉층을 갖는 RNN을 보여주고 있습니다.
- 여기서 중요한 점은 첫 번째 은닉층과 두 번째 은닉층이 같은 노드의 갯수이며, 첫 번째 은닉층은 return_sequences=True이지만 두 번째 은닉층은 return_sequences=False 입니다.



return_sequences = False

return_sequences = True

2 깊은 순환 신경망(Deep RNN)

기상청 Korea Meteorological Administration

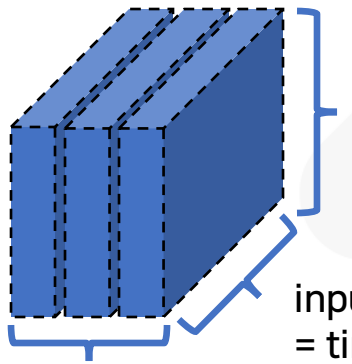
- keras로 Deep RNN은 아래와 같이 구현할 수 있습니다.

```
from tensorflow.keras.layers import Dense, SimpleRNN, Bidirectional

# timesteps,  input_dim
print(X_train.shape[-2:])

h_units = 100

model = models.Sequential()
model.add(SimpleRNN(h_units, return_sequences=True, recurrent_dropout=0.5, kernel_initializer='he_normal', input_shape=X_train.shape[-2:],))
model.add(SimpleRNN(h_units))
model.add(Dense(1))

model.compile(loss='mse', metrics=['mae'], optimizer=Adam(learning_rate = 0.0006))
```

```
model.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
simple_rnn_2 (SimpleRNN)     (None, 24, 100)           14600
_________________________________________________________________
simple_rnn_3 (SimpleRNN)     (None, 100)               20100
_________________________________________________________________
dense (Dense)                (None, 1)                 101
=================================================================
Total params: 34,801
Trainable params: 34,801
Non-trainable params: 0
_________________________________________________________________
```

input_shape = (timesteps, input_dim)



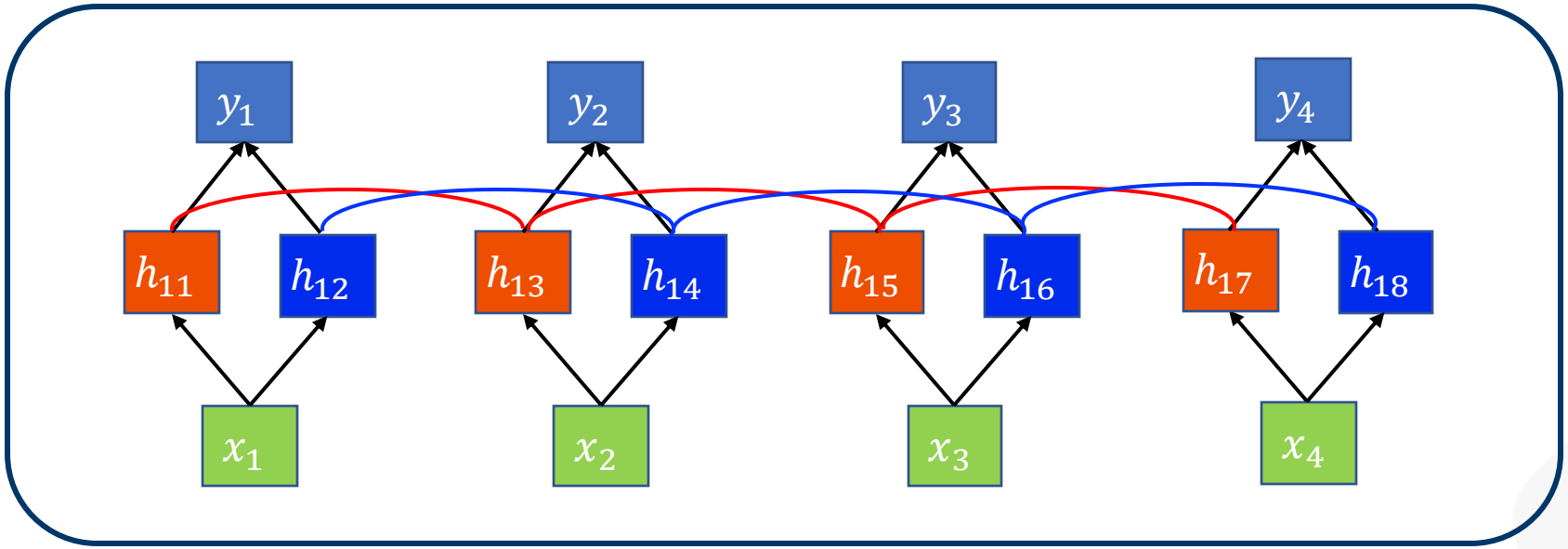
입력 노드의 갯수
= Input_dim

input_length
= timesteps

샘플의 갯수
= batch_size

3 양방향 순환 신경망(Bidirectional RNN)

- 양방향 순환 신경망은 은닉층의 출력을 이후 시점의 은닉 노드에 전달할 뿐 아니라 이전 시점의 노드에도 전달합니다.



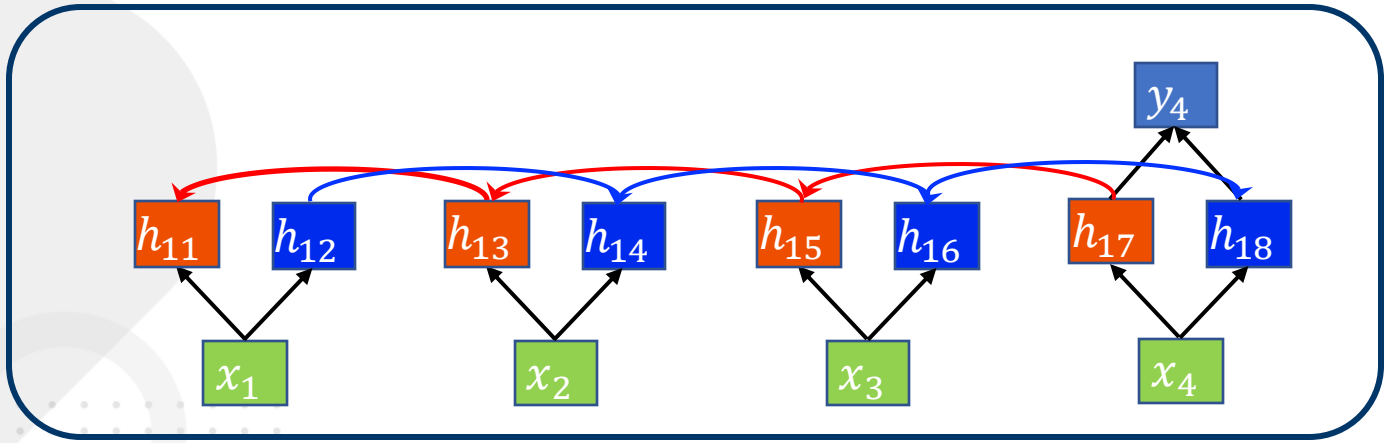
3 양방향 순환 신경망(Bidirectional RNN)

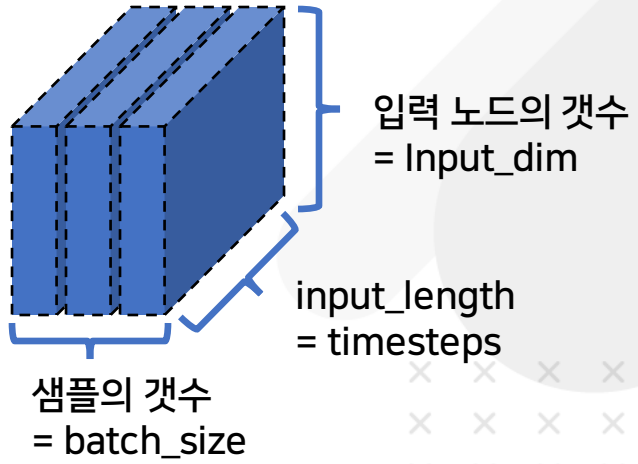
기상청 Korea Meteorological Administration

- keras로 Bidirectional RNN 은 아래와 같이 구현할 수 있습니다.

```
from tensorflow.keras import Input, Model
from tensorflow.keras.layers import Dense, SimpleRNN, Bidirectional

inputs = Input(shape=(X_train.shape[-2:]))
x = Bidirectional(SimpleRNN(h_units, return_sequences=False, return_state=False, kernel_initializer='he_normal'))(inputs)
outputs = Dense(1)(x)

model = Model(inputs, outputs)

model.compile(loss='mse', metrics=['mae'], optimizer=Adam(learning_rate = 0.0006))
```

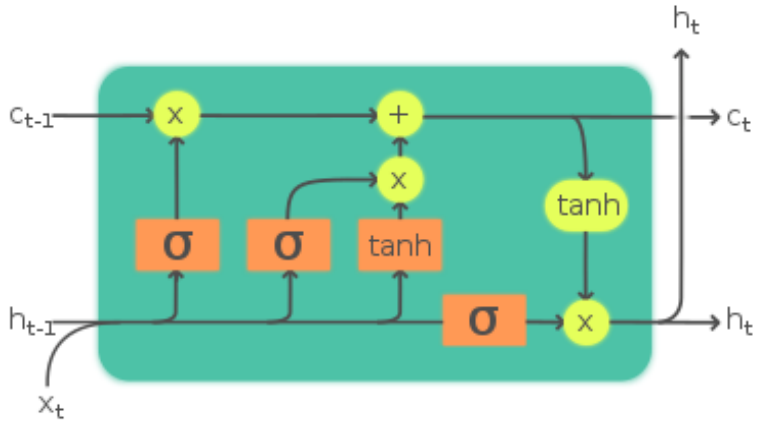
input_shape = (timesteps, input_dim)





4 장단기 메모리(Long Short-Term Memory)

- RNN은 상대적으로 짧은 시퀀스 데이터에 대해서만 좋은 성능을 보이는 단점이 있습니다.
- Timestep이 길어질 수록 앞의 시점의 정보가 뒤로 전달될수록 소실되는 현상이 있습니다.
- 이에 대한 단점을 보완하고자 제안된 방법이 LSTM입니다.
- LSTM은 은닉층의 메로리 셀에 입력 게이트, 망각 게이트, 출력 게이트를 추가하여 오래 기억해야할 것과 기억에서 지워야 할 것은 지웁니다.



LSTM의 구조

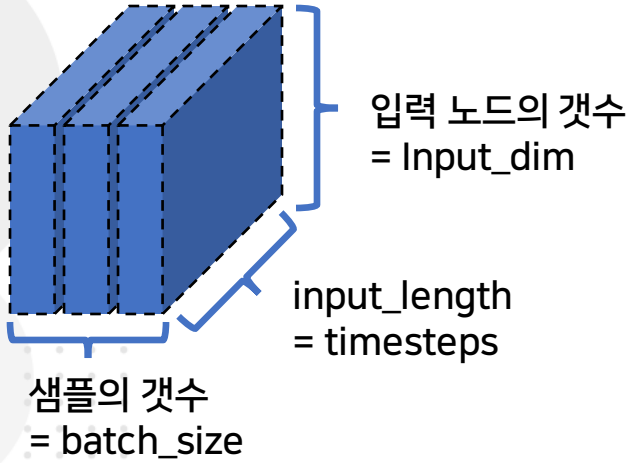
참조 : 위키피디아

4 장단기 메모리(Long Short-Term Memory)

기상청 Korea Meteorological Administration

- keras로 LSTM은 아래와 같이 구현할 수 있습니다.

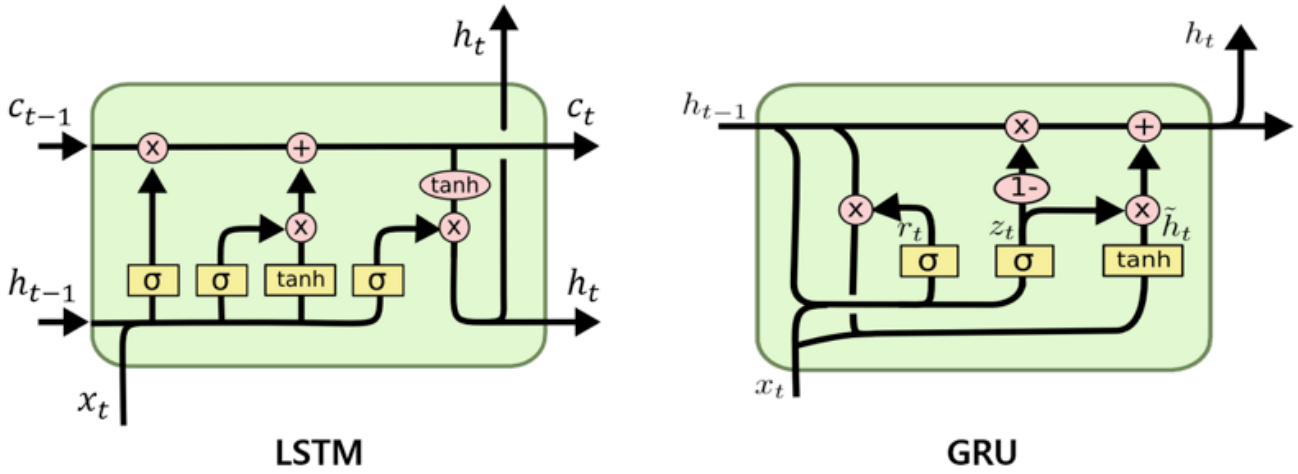
```
from tensorflow.keras import Input, Model
from tensorflow.keras.layers import Dense, SimpleRNN, Bidirectional, Dropout

inputs = Input(shape=(X_train.shape[-2:]))
x = LSTM(h_units, recurrent_dropout=0.5, kernel_initializer='he_normal')(inputs)
x = Dropout(0.5)(x)
#x = LSTM(h_units, return_sequences=False, kernel_initializer='he_normal')(x)

outputs = Dense(1)(x)

model = Model(inputs, outputs)

model.compile(loss='mse', metrics=['mae'], optimizer=Adam(learning_rate = 0.0006))
```



inputs = (timesteps, input_dim)

5 게이트 순환 유닛(Gated Recurrent Unit)

- GRU는 LSTM의 장기 의존성 문제를 해결하면서 은닉 상태를 업데이트 하는 계산을 줄였습니다.
- GRU는 성능은 LSTM과 비슷하고, 구조는 LSTM보다 단순화 시켰습니다.
- GRU는 업데이트 게이트와 리셋 게이트로 구성됩니다.



LSTM과 GRU 구조 비교

수고하셨습니다