- Basic Architecture -

박성호 (neowizard2018@gmail.com)

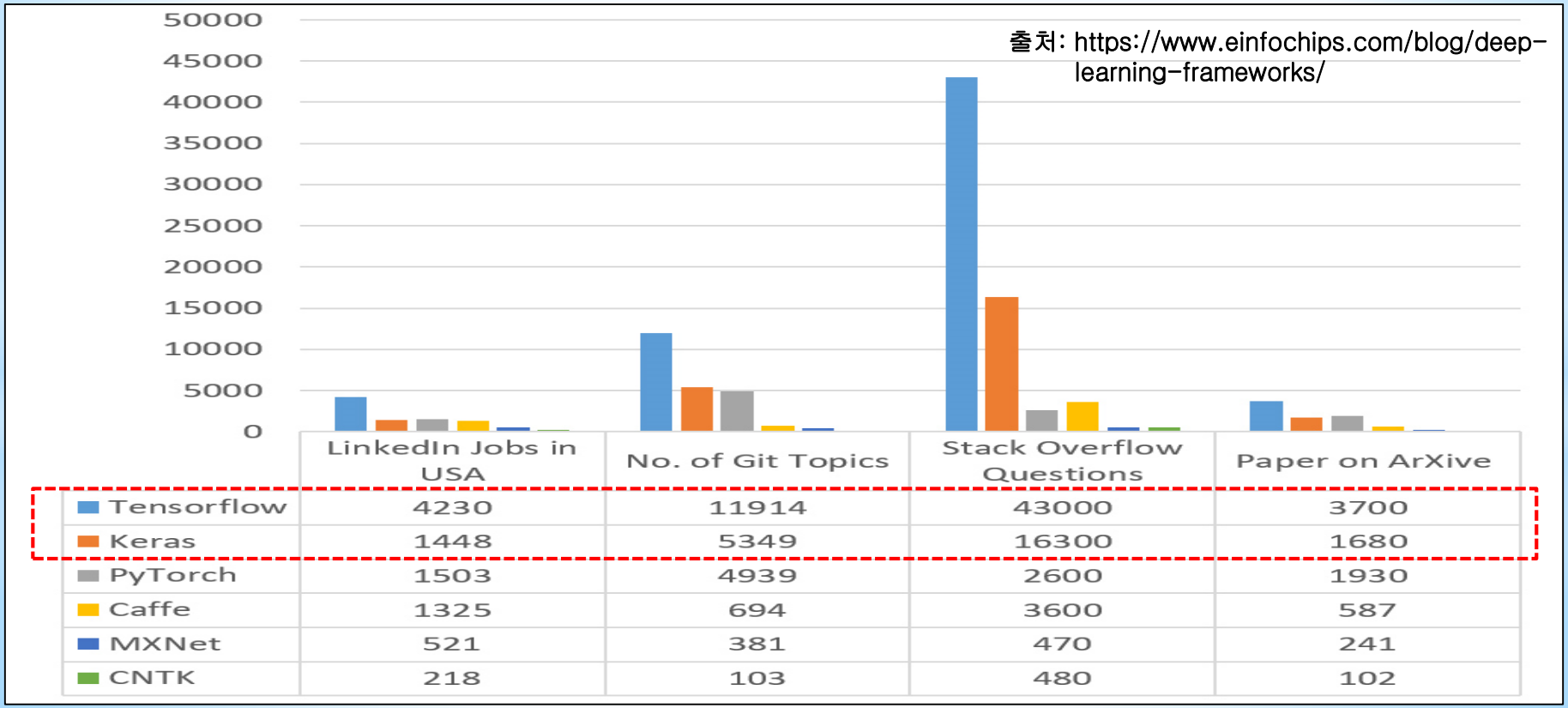
Overview – TensorFlow 2.x

➢ TensorFlow는 텐서(Tensor)를 흘려보내면서(Flow) 딥러닝 알고리즘을 수행하는 프레임워크.

– 2019년 9월 30일에 TensorFlow 2.0 정식 Release 되었으며 1.x 버전과 비교하면

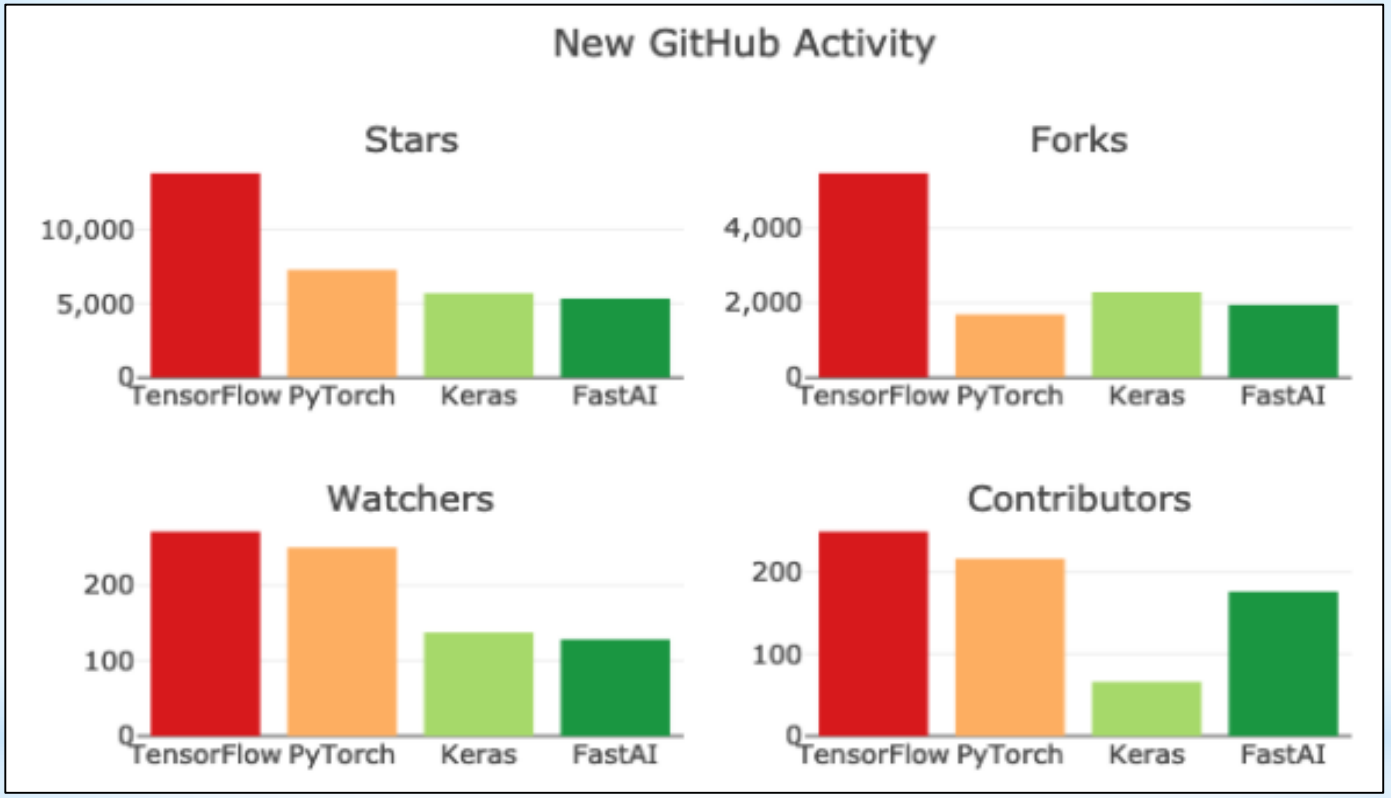
① 사용자 친화적(Keras as High Level API) ② 코드 가독성과 직관성을 높이는 Eager Execution 적용

"If you have started with TensorFlow 2.0 and have never seen TensorFlow 1.x, then you are lucky.", Deep Learning with TensorFlow 2 and Keras, 2nd Edition, Packt, 2020.04



	LinkedIn Jobs in USA	No. of Git Topics	Stack Overflow Questions	Paper on ArXive
Tensorflow	4230	11914	43000	3700
Keras	1448	5349	16300	1680
PyTorch	1503	4939	2600	1930
Caffe	1325	694	3600	587
MXNet	521	381	470	241
CNTK	218	103	480	102

Overview – GitHub Activity (Developer Trend)



출처: https://dschloe.github.io/python/tensorflow2.0/googlecolab/#fn:1

Overview – Keras vs TensorFlow vs PyTorch

Keras is most suitable for:

- Rapid Prototyping
- Small Dataset
- Multiple back-end support

TensorFlow is most suitable for:

- Large Dataset
- High Performance
- Functionality
- Object Detection

PyTorch is most suitable for:

- Flexibility
- Short Training Duration
- Debugging capabilities

출처: https://www.edureka.co/blog/keras-vs-tensorflow-vs-pytorch/

Keras as High Level API

➢ Keras in TensorFlow 2.0

- Keras 창시자 프랑소와 숄레(François Chollet)가 TF 2.0 개발에 참여하였고, TF 2.0 에서 공식적이고 유일한 High-Level API 로서 Keras가 선정되었음

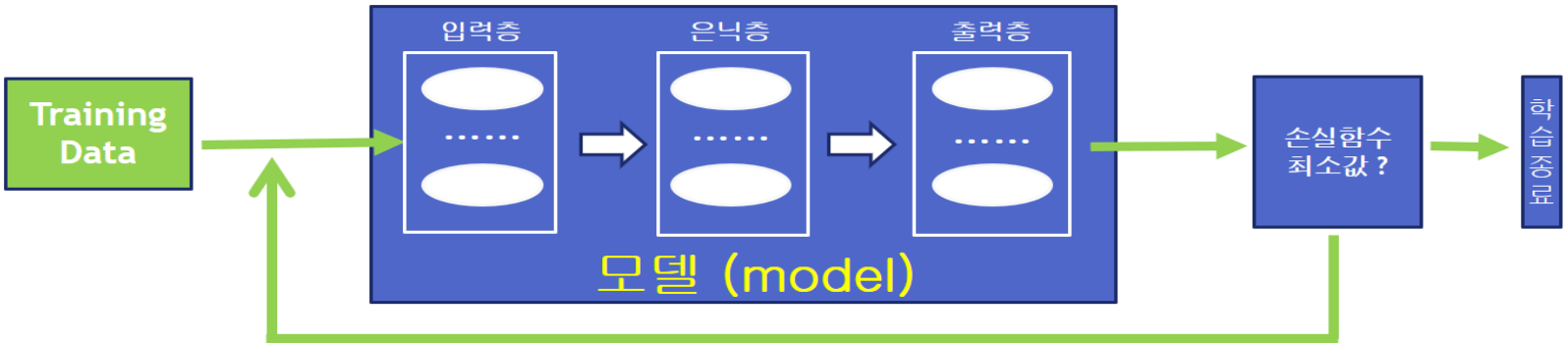
- 또한 프랑소와 숄레는 앞으로 native Keras 보다는 tf.keras를 사용할 것을 권장하고 있음

➢ Keras 특징

- 사용자 친근성 (User Friendliness) : 직관적인 API를 이용하면 ANN, CNN, RNN 또는 이를 조합한 딥러닝 모델을 쉽게 구축 할 수 있음

- 모듈성 (Modularity) : Keras에서 제공하는 모듈은 독립적으로 설정 가능함. 즉 신경망 층, 손실함수, 활성화 함수, 최적화 알고리즘, 정규화 기법 등은 모두 독립적인 모듈이기 때문에 이러한 모듈을 서로 조합하기만 하면 새로운 딥러닝 모델을 쉽게 만들고 학습시킬 수 있음

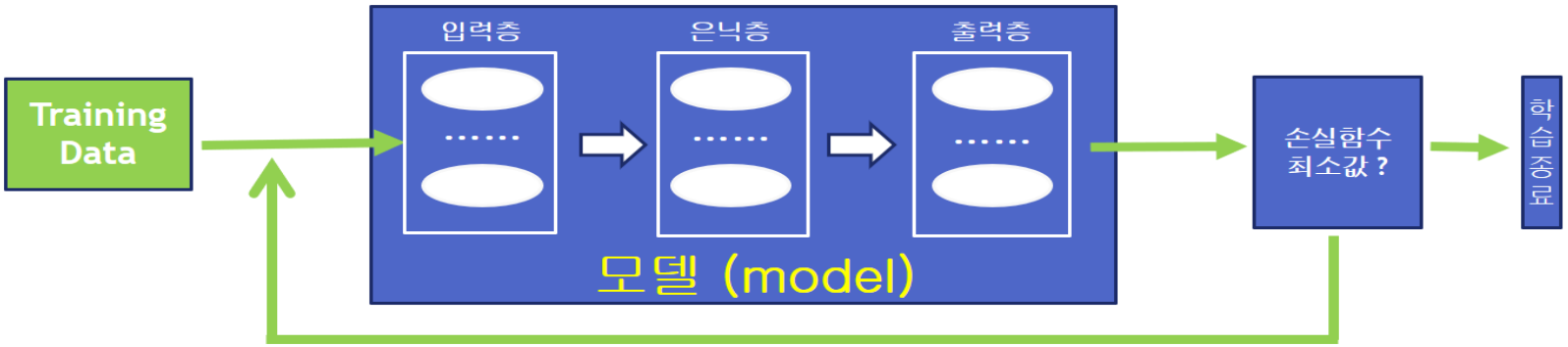
Keras – model / layer (https://youtu.be/Ke70Xxj2EJw)



- 케라스의 모델(model)은 신경망 자체이며, 모든 모델의 기본 단위는 층(layer)으로 나타냄

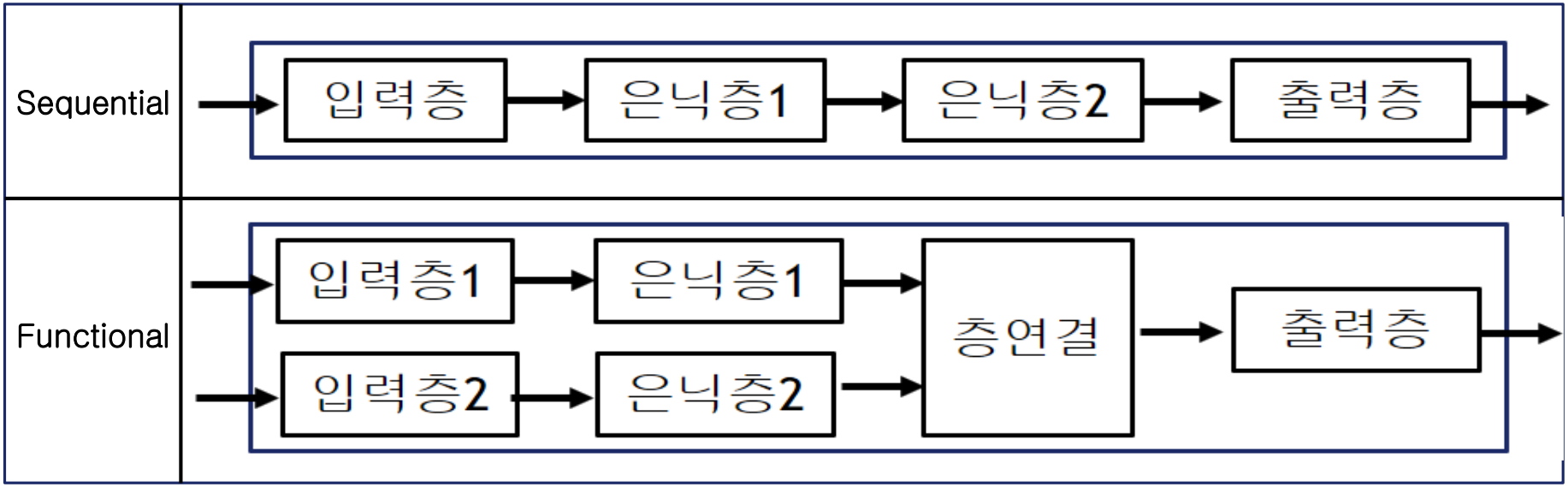
tf.keras.layers.Flatten()	입력데이터(텐서)를 1차원 vector로 만들어주는 역할을 수행함
tf.keras.layers.Dense(100, activation='relu') tf.keras.layers.Dense(100, activation='sigmoid') tf.keras.layers.Dense(10, activation='softmax')	은닉층과 출력층을 의미하는 완전 연결층을 나타내며, 1st 파라미터는 출력 노드 수 이며, 활성화 함수는 activation='...' 나타냄

Keras – Sequential API / Functional API

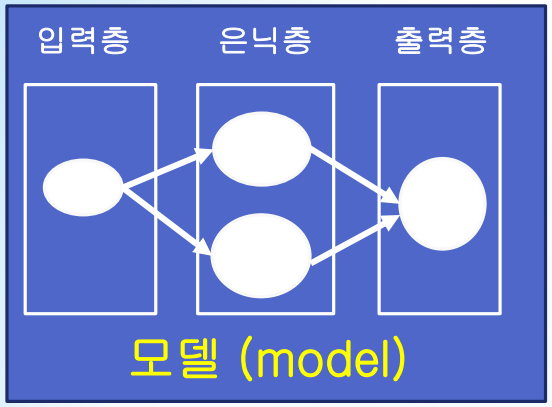


- 케라스 모델은 다양한 층(layer)로 이루어지는데, 이러한 층(layer)을 구성하는 방식에는 크게 Sequential API, Functional API, Subclassing API 등으로 정의할 수 있음

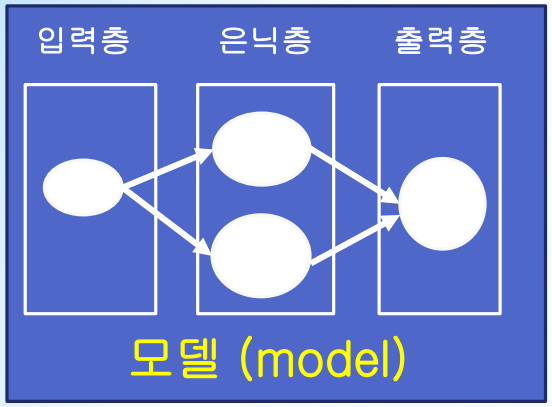
- 초급 개발자 또는 간단한 모델은 일반적으로 Sequential API , 전문가 또는 복잡한 모델은 Functional API 기반의 모델이 주로 사용됨



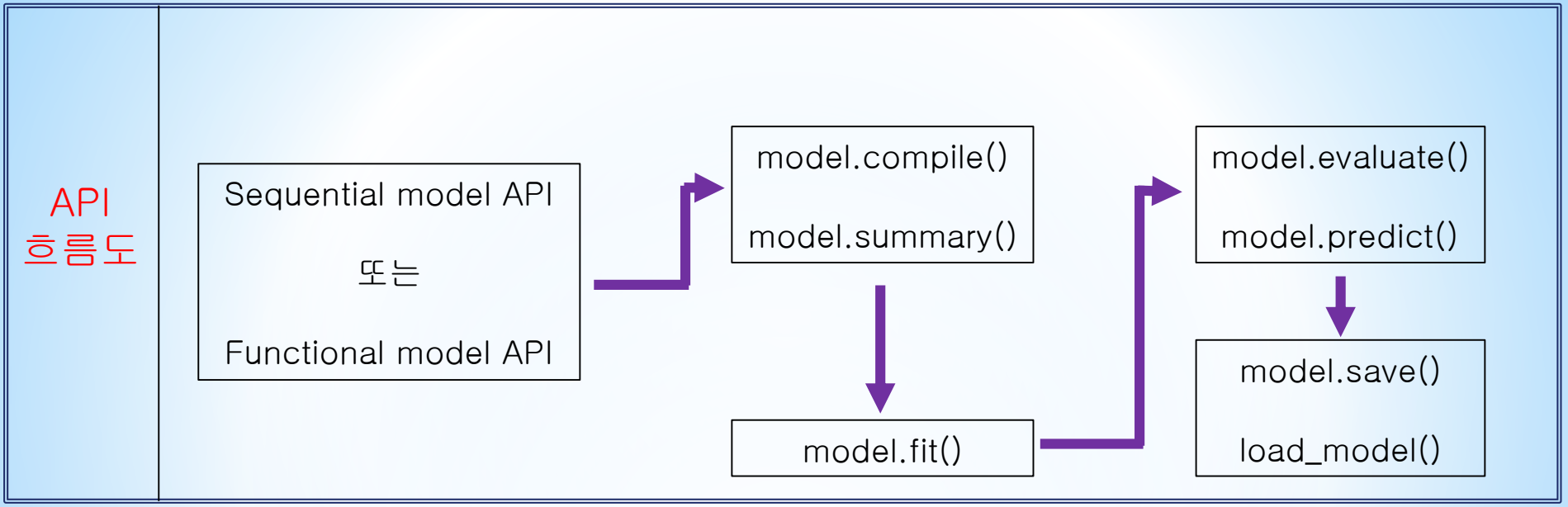
Keras Sequential model

모델 구축	 model = Sequential() model.add(Flatten(input_shape=(1,)) model.add(Dense(2, activation='sigmoid') model.add(Dense(1, activation='sigmoid') ① ② model.add(Dense(2, activation='sigmoid', input_shape=(1,))
컴파일	[예1] model.compile(optimizer=SGD(learning_rate=0.1), loss='mse', metrics=['accuracy']) [예2] model.compile(optimizer=Adam(learning_rate=1e-4), loss='categorical_crossentropy) ※ 손실함수 종류는 정답이 실수 'mse', 정답이 0 또는 1 인 이항분류 'binary_crossentropy', 다중 분류시 one-hot encoding 한 후에 넣어주는 경우 'categorical_crossentropy' , 다중 분류시 one-hot encoding 하지않고 정수로 넣어주는 경우 'sparse_ categorical_crossentropy'
학습	[예] model.fit(x_train, t_train, epochs=10, batch_size=100, verbose=0, validation_split=0.2)
평가, 예측, 저장, 로드	model.evaluate(x_test, t_test) , model.predict(x_input_data) model.save("model_name.h5") , model = tensorflow.keras.models.load_model("model_name.h5")

Keras Functional model (https://youtu.be/KugOEdth3iE)

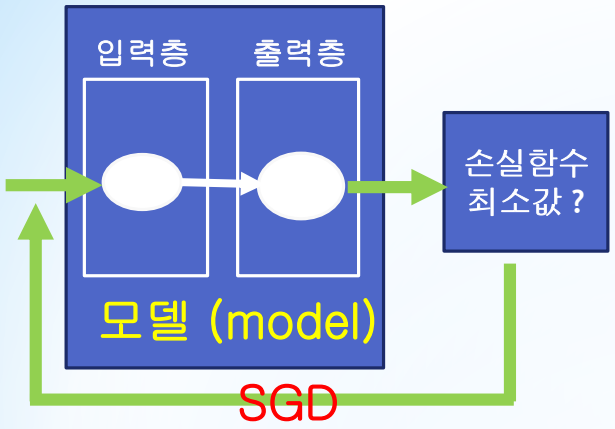
모델 구축	 in_ = Input(shape=(1,) x = Dense(2, activation='sigmoid')(in_) out_ = Dense(1, activation='sigmoid')(x) model = Model(inputs=in_, outputs=out_)
컴파일	[예1] model.compile(optimizer=SGD(learning_rate=0.1), loss='mse', metrics=['accuracy']) [예2] model.compile(optimizer=Adam(learning_rate=1e-4), loss='categorical_crossentropy) ※ 손실함수 종류는 정답이 실수 'mse', 정답이 0 또는 1 인 이항분류 'binary_crossentropy', 다중 분류시 one-hot encoding 한 후에 넣어주는 경우 'categorical_crossentropy' , 다중 분류시 one-hot encoding 하지않고 정수로 넣어주는 경우 'sparse_ categorical_crossentropy'
학습	[예] model.fit(x_train, t_train, epochs=10, batch_size=100, verbose=0, validation_split=0.2)
평가, 예측, 저장, 로드	model.evaluate(x_test, t_test) , model.predict(x_input_data) model.save("model_name.h5") , model = tensorflow.keras.models.load_model("model_name.h5")

Keras – 딥러닝 모델 구축 (API 흐름도)



Keras - Simple LinearRegression Exercise (Sequential)

입력	정답
1	3
2	4
3	5
4	6
5	7
6	8




```
import tensorflow as tf

from tensorflow.keras.models import Sequential, Model
from tensorflow.keras.layers import Flatten, Dense, Input
from tensorflow.keras.optimizers import SGD

import numpy as np
```

```
x_data = np.array([1, 2, 3, 4, 5, 6])
t_data = np.array([3, 4, 5, 6, 7, 8])
```

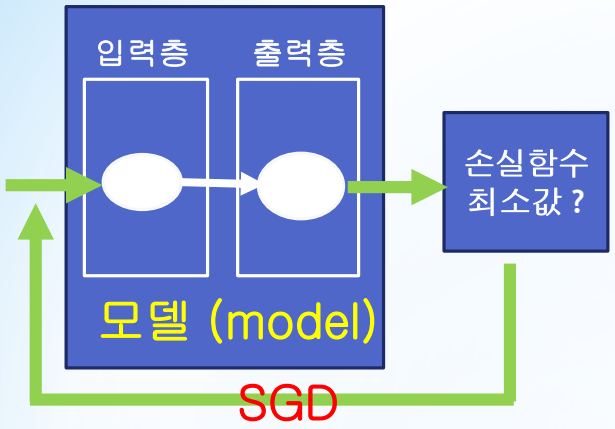
```
model = Sequential()    # 모델

model.add(Flatten(input_shape=(1,)))       # 입력층

model.add(Dense(1, activation='linear'))   # 출력층

# model.add(Dense(1, input_shape=(1,), activation='linear'))
```

```
model.compile(optimizer=SGD(), loss='mse')

model.summary()
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
flatten (Flatten)            (None, 1)                 0
_________________________________________________________________
dense (Dense)                (None, 1)                 2
=================================================================
Total params: 2
Trainable params: 2
Non-trainable params: 0
_________________________________________________________________
```

Keras - Simple LinearRegression Exercise (Sequential)

입력	정답
1	3
2	4
3	5
4	6
5	7
6	8




```
import tensorflow as tf

from tensorflow.keras.models import Sequential, Model
from tensorflow.keras.layers import Flatten, Dense, Input
from tensorflow.keras.optimizers import SGD

import numpy as np
```

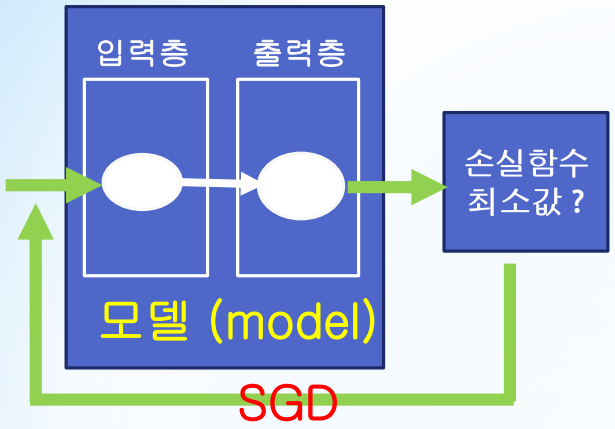
```
hist = model.fit(x_data, t_data, epochs=1000)
```

```
test_input_data = np.array([-3.1, 3.0, 3.5, 15.0, 20.1])
label_data = test_input_data + 2.0

result = model.predict(test_input_data)

print(result)
print(label_data.reshape(5,1))
```

```
[[-1.1591684]
 [ 4.98973  ]
 [ 5.4937377]
 [17.085922 ]
 [22.226805 ]]
[[-1.1]
 [ 5. ]
 [ 5.5]
 [17. ]
 [22.1]]
```

Keras - Simple LinearRegression Exercise (Functional)

입력	정답
1	3
2	4
3	5
4	6
5	7
6	8




```
import tensorflow as tf

from tensorflow.keras.models import Sequential, Model
from tensorflow.keras.layers import Flatten, Dense, Input
from tensorflow.keras.optimizers import SGD

import numpy as np
```

```
x_data = np.array([1, 2, 3, 4, 5, 6])
t_data = np.array([3, 4, 5, 6, 7, 8])
```

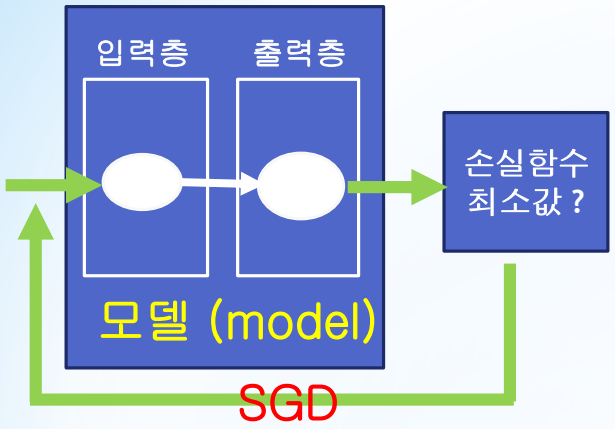
```
input_ = Input(shape=(1,))

#x = Flatten()(input_)

output_ = Dense(1, activation='linear')(input_)

model = Model(inputs=input_, outputs=output_)
```

```
model.compile(optimizer=SGD(), loss='mse')

model.summary()
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
flatten (Flatten)            (None, 1)                 0
_________________________________________________________________
dense (Dense)                (None, 1)                 2
=================================================================
Total params: 2
Trainable params: 2
Non-trainable params: 0
_________________________________________________________________
```

Keras - Simple LinearRegression Exercise (Functional)

입력	정답
1	3
2	4
3	5
4	6
5	7
6	8




```
import tensorflow as tf

from tensorflow.keras.models import Sequential, Model
from tensorflow.keras.layers import Flatten, Dense, Input
from tensorflow.keras.optimizers import SGD

import numpy as np
```

```
hist = model.fit(x_data, t_data, epochs=1000)
```

```
test_input_data = np.array([-3.1, 3.0, 3.5, 15.0, 20.1])
label_data = test_input_data + 2.0

result = model.predict(test_input_data)

print(result)
print(label_data.reshape(5,1))
```

```
[[-1.1783116]
 [ 4.9864073]
 [ 5.4917116]
 [17.113722 ]
 [22.267832 ]]
[[-1.1]
 [ 5. ]
 [ 5.5]
 [17. ]
 [22.1]]
```