

AI 프로그래밍

- 2024



14주차

케라스 신경망 실습 - 타이타닉 생존자 예측

인하공전 컴퓨터 정보공학과

과제 1 titanic_test.ipynb test.csv, train.csv

모델을 다음과 같이 변경 하고 code (모델 생성 부분만)과 정확도(accuracy)를 제출 하시오.

1)

입력 층 : 변동 없음
은닉층 1 : 16 (relu)
은닉층 2 : 8 (relu)
은닉층 3 : 8 (relu)
출력 층 변동 없음.

2)

입력 층 : 변동 없음
은닉층 1 : 16 (relu)
은닉층 2 : 16 (relu)
은닉층 3 : 8 (relu)
출력 층 변동 없음.

```
model.add(tf.keras.layers.Dense(16, activation='relu', input_shape=(2,)))
model.add(tf.keras.layers.Dense(8, activation='relu'))
model.add(tf.keras.layers.Dense(8, activation='relu'))
model.add(tf.keras.layers.Dense(1, activation='sigmoid'))
```

Model: "sequential_2"

입력 층 : 2
은닉층 1 : 16 (relu)
은닉층 2 : 8 (relu)
은닉층 3 : 8 (relu)
출력 층 : 1

Layer (type)	Output Shape	Param #
=====		
dense_6 (Dense)	(None, 16)	48
dense_7 (Dense)	(None, 8)	136
dense_8 (Dense)	(None, 8)	72
dense_9 (Dense)	(None, 1)	9
=====		

Total params: 265
Trainable params: 265
Non-trainable params: 0

```
model.add(Dense(30, input_dim=12, activation='relu'))
model.add(Dense(12, activation='relu'))
model.add(Dense(8, activation='relu'))
model.add(Dense(1, activation='sigmoid'))
```

입력 층 : 12
은닉층 1 : 30 (relu)
은닉층 2 : 12 (relu)
은닉층 3 : 8 (relu)
출력 층 : 1 (sigmoid)

Layer (type)	Output Shape	Param #
dense (Dense)	(None, 30)	390
dense_1 (Dense)	(None, 12)	372
dense_2 (Dense)	(None, 8)	104
dense_3 (Dense)	(None, 1)	9

케라스 신경망 실습 - 주택 가격 예측

과제5 (house.ipynb,house_train.csv)

1) 아래와 같이 모델과 학습 방법을 변경 한 후 모델을 house1.hdf5로 저장 하시오.

입력 층 : 변동 없음
은닉층 1 : 20 (relu)
은닉층 2 : 20 (relu)
은닉층 3 : 40 (relu)
출력 층 변동 없음.

17회 이상 결과가 향상 되지 않으면 자동으로 학습이 중단되게
Batch size 16

2) 아래와 같이 모델과 학습 방법을 변경 한 후 모델을 house2.hdf5로 저장 하시오.

입력 층 : 변동 없음
은닉층 1 : 10 (relu)
은닉층 2 : 20 (relu)
은닉층 3 : 20 (relu)
출력 층 변동 없음.

15회 이상 결과가 향상 되지 않으면 자동으로 학습이 중단되게
Batch size 20

code (.ppt에 직접 붙여도 되고 .py 파일로 제출 가능) 와 model 제출 (house1.hdf5, hoise2.hdf5) 제출

```
model.add(tf.keras.layers.Dense(16, activation='relu', input_shape=(2,)))
model.add(tf.keras.layers.Dense(8, activation='relu'))
model.add(tf.keras.layers.Dense(8, activation='relu'))
model.add(tf.keras.layers.Dense(1, activation='sigmoid'))
```

Model: "sequential_2"

입력 층 : 2
은닉층 1 : 16 (relu)
은닉층 2 : 8 (relu)
은닉층 3 : 8 (relu)
출력 층 : 1

Layer (type)	Output Shape	Param #
=====		
dense_6 (Dense)	(None, 16)	48
dense_7 (Dense)	(None, 8)	136
dense_8 (Dense)	(None, 8)	72
dense_9 (Dense)	(None, 1)	9
=====		

Total params: 265
Trainable params: 265
Non-trainable params: 0

```
model.add(tf.keras.layers.Dense(16, activation='relu', input_shape=(2,)))
model.add(tf.keras.layers.Dense(8, activation='relu'))
model.add(tf.keras.layers.Dense(8, activation='relu'))
model.add(tf.keras.layers.Dense(1, activation='sigmoid'))
```

Model: "sequential_2"

입력 층 : 2
은닉층 1 : 16 (relu)
은닉층 2 : 8 (relu)
은닉층 3 : 8 (relu)
출력 층 : 1



Layer (type)	Output Shape	Param #
=====		
dense_6 (Dense)	(None, 16)	48
dense_7 (Dense)	(None, 8)	136
dense_8 (Dense)	(None, 8)	72
dense_9 (Dense)	(None, 1)	9
=====		

Total params: 265
Trainable params: 265
Non-trainable params: 0

- 저장된 모델로 확인
- house1.hdf5, house2.hdf5를 load 하여

test loss 확인

house_model_test.ipynb

 house1.hdf5	2022-05-14 오후 ...	HDF5 파일	69KB
 house2.hdf5	2022-05-14 오후 ...	HDF5 파일	61KB

모델 파일에는 모델의 구조와 가중치(w) bias 값이 들어있다.

과제1) house_testonly.ipynb

```
X_train, X_test, y_train, y_test = train_test_split(X_train_pre, y, test_size=0.2, shuffle=False)
```

```
model=load_model ('/content/house1.hdf5')  
print('테스트 loss1', model.evaluate(X_test,y_test)/10000000)
```

```
model=load_model ('/content/house2.hdf5')  
print('테스트 loss2', model.evaluate(X_test,y_test)/10000000)
```

house1.hdf5

house2.hdf5

test loss 값 과제 설명에 입력

1. Quality 예측
2. 테스트 데이터, 훈련 데이터 분리
3. 검증 set 사용
4. Early stopping 적용
5. Model 저장
6. Metric은 mse
7. Test 데이터 이용한 성능 측정 (mse)

성능 기준 0.6 보다 작게

wineqr_assign.ipynb code 채우기

*.py로 upload

```
from sklearn.preprocessing import MinMaxScaler

# 데이터 세트를 읽어들인다.
df = pd.read_csv("/content/wineqrev.csv", sep=',')

X=df.iloc[:,0:11]
y=df.iloc[:,11]

### 0과 1사이로 정규화
mms=MinMaxScaler()
mms.fit(X)
print(X.head())
X_mms=mms.transform(X)
X = pd.DataFrame(X_mms, columns=X.columns,
index=list(X.index.values))
```

```
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.3,shuffle=True)
print(df.quality)
```

```
# 케라스 모델을 생성한다.
```

```
model = keras.models.Sequential()
model.add(keras.layers.Dense(16, activation='relu', input_shape=(11,)))
model.add(keras.layers.Dense(8, activation='relu'))
model.add(keras.layers.Dense(1))
```

```
# 케라스 모델을 컴파일한다.
```

```
model.compile(loss='mse',
              optimizer='adam',
              metrics=['mse'])
```

20회 이상 결과가 향상되지 않으면 자동으로 중단되게끔 합니다.

```
early_stopping_callback = EarlyStopping(monitor='val_loss', patience=20)
```

모델의 이름을 정합니다.

```
modelpath='/content/wine_modelr.hdf5'
```

최적화 모델을 업데이트하고 저장합니다.

```
checkpointer = ModelCheckpoint(filepath=modelpath, monitor='val_loss', verbose=1,  
save_best_only=True)
```

실행 관련 설정을 하는 부분입니다. 전체의 20%를 검증셋으로 설정합니다.

```
history = model.fit(X_train, y_train, validation_split=0.25, epochs=2000, batch_size=32,  
callbacks=[early_stopping_callback, checkpointer])
```

테스트 결과를 출력합니다.

```
score=model.evaluate(X_test, y_test)  
print('Test accuracy:', score[1])
```

와인 품질 예측 - 분류

인하공전 컴퓨터 정보공학과

1. 0~9 등으로 분류 (10 카테고리: 원 핫 인코딩)
2. 테스트 데이터, 훈련 데이터 분리
3. 검증 set 사용
4. Early stopping 적용
5. Model 저장
6. Metric은 accuracy
7. Test 데이터 이용한 성능 측정 (accuracy)

성능 기준 accuracy 0.54 보다 크게

성능 기준

wineqc_assign.ipynb code 채우기

*.py로 upload

데이터 세트를 읽어들인다.

```
df = pd.read_csv("/content/wineqrev.csv", sep=',')
```

```
X=df.iloc[:,0:11]
```

```
y=df.iloc[:,11]
```

one hot encoding으로 label 형태 변환

```
y=keras.utils.to_categorical(y)
```

0~ 1 사이로 정규화

```
mms=MinMaxScaler()
```

```
mms.fit(X)
```

```
print(X.head())
```

```
X_mms=mms.transform(X)
```

```
X = pd.DataFrame(X_mms, columns=X.columns,  
index=list(X.index.values))
```

```
print(X.head())
```

code 작성

```
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.3,shuffle=True)
print(df.quality)
```

케라스 모델을 생성한다.

```
model = keras.models.Sequential()
model.add(keras.layers.Dense(16, activation='relu',
input_shape=(11,)))
model.add(keras.layers.Dense(32, activation='relu'))
model.add(keras.layers.Dense(32, activation='relu'))
model.add(keras.layers.Dense(32, activation='relu'))
model.add(keras.layers.Dense(10,activation='softmax'
))
```

케라스 모델을 컴파일한다.

```
model.compile(loss='categorical_crossentropy',
optimizer='adam',
metrics=['accuracy'])
```


20회 이상 결과가 향상되지 않으면 자동으로 중단되게끔 합니다.

```
early_stopping_callback = EarlyStopping(monitor='val_loss', patience=20)
```

모델의 이름을 정합니다.

```
modelpath='/content/wine_modelr.hdf5'
```

최적화 모델을 업데이트하고 저장합니다.

```
checkpointer = ModelCheckpoint(filepath=modelpath, monitor='val_loss', verbose=1,  
save_best_only=True)
```

실행 관련 설정을 하는 부분입니다. 전체의 20%를 검증셋으로 설정합니다.

```
history = model.fit(X_train, y_train, validation_split=0.25, epochs=2000, batch_size=32,  
callbacks=[early_stopping_callback, checkpointer])
```

테스트 결과를 출력합니다.

```
score=model.evaluate(X_test, y_test)  
print('Test accuracy:', score[1])
```

Convolution Neural Network (CNN)

(과제 1)

인하공저 컴퓨터 정보과
인하공전 컴퓨터 정보공학과

- 모델 디자인 (mnist_paratest.ipynb) **1x1 kernel(filter)은 사용할 수 없음.**

예제 code를 이용하여 model.summar가 다음과 같이 출력되도록 code를 수정하고 제출

Layer (type)	Output Shape	Param #
=====		
=====		
conv2d_3 (Conv2D)	(None, 26, 26, 16)	160
max_pooling2d_2 (MaxPooling 2D)	(None, 13, 13, 16)	0
conv2d_4 (Conv2D)	(None, 13, 13, 32)	4640
max_pooling2d_3 (MaxPooling 2D)	(None, 6, 6, 32)	0
conv2d_5 (Conv2D)	(None, 4, 4, 64)	18496
flatten_1 (Flatten)	(None, 1024)	0
dense_2 (Dense)	(None, 64)	65600
dense_3 (Dense)	(None, 10)	650
=====		
=====		

code

Convolution Neural Network (CNN)^{과제2)}

인하공전 컴퓨터 정보공학과

■ 모델 디자인 (mnist_paratest.ipynb) **1x1 kernel(filter)은 사용할 수 없음.**

예제 code를 이용하여 model.summar가 다음과 같이 출력되도록 code를 수정하고 제출
code

Layer (type)	Output Shape	Param #
=====		
conv2d (Conv2D)	(None, 28, 28, 32)	320
max_pooling2d (MaxPooling2D)	(None, 14, 14, 32)	0
conv2d_1 (Conv2D)	(None, 10, 10, 64)	51264
max_pooling2d_1 (MaxPooling 2D)	(None, 5, 5, 64)	0
conv2d_2 (Conv2D)	(None, 3, 3, 128)	73856
flatten (Flatten)	(None, 1152)	0
dense (Dense)	(None, 64)	73792
dense_1 (Dense)	(None, 10)	650

Convolution Neural Network (CNN)

■ 모델 디자인 (mnist_paratest.ipynb)

(과제1)

```
input_shape=(28,28,1)

model.add(layers.Input(shape=input_shape))
model.add(layers.Conv2D(16, (3, 3), activation='relu'))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Conv2D(32, (3, 3), activation='relu',padding='same'))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Conv2D(64, (3, 3), activation='relu'))

model.add(layers.Flatten())
model.add(layers.Dense(64, activation='relu'))
model.add(layers.Dense(10, activation='softmax'))
```

Layer (type)	Output Shape	Param #
=====		
==		
conv2d_3 (Conv2D)	(None, 26, 26, 16)	160
max_pooling2d_2 (MaxPooling 2D)	(None, 13, 13, 16)	0
conv2d_4 (Conv2D)	(None, 13, 13, 32)	4640
max_pooling2d_3 (MaxPooling 2D)	(None, 6, 6, 32)	0
conv2d_5 (Conv2D)	(None, 4, 4, 64)	18496
flatten_1 (Flatten)	(None, 1024)	0
dense_2 (Dense)	(None, 64)	65600
dense_3 (Dense)	(None, 10)	650
=====		
==		

Convolution Neural Network (CNN)

■ 모델 디자인 (mnist_paratest.ipynb)

(과제2)

```
input_shape=(28,28,1)

model.add(layers.Input(shape=input_shape))
model.add(layers.Conv2D(32, (3, 3),
activation='relu',padding='same'))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Conv2D(64, (5, 5), activation='relu'))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Conv2D(128, (3, 3), activation='relu'))

model.add(layers.Flatten())
model.add(layers.Dense(64, activation='relu'))
model.add(layers.Dense(10, activation='softmax'))
```

Layer (type)	Output Shape	Param #
=====		
conv2d (Conv2D)	(None, 28, 28, 32)	320
max_pooling2d (MaxPooling2D)	(None, 14, 14, 32)	0
conv2d_1 (Conv2D)	(None, 10, 10, 64)	51264
max_pooling2d_1 (MaxPooling 2D)	(None, 5, 5, 64)	0
conv2d_2 (Conv2D)	(None, 3, 3, 128)	73856
flatten (Flatten)	(None, 1152)	0
dense (Dense)	(None, 64)	73792
dense_1 (Dense)	(None, 10)	650

컨볼루션 신경망 (CNN: Convolutional Neural Network)

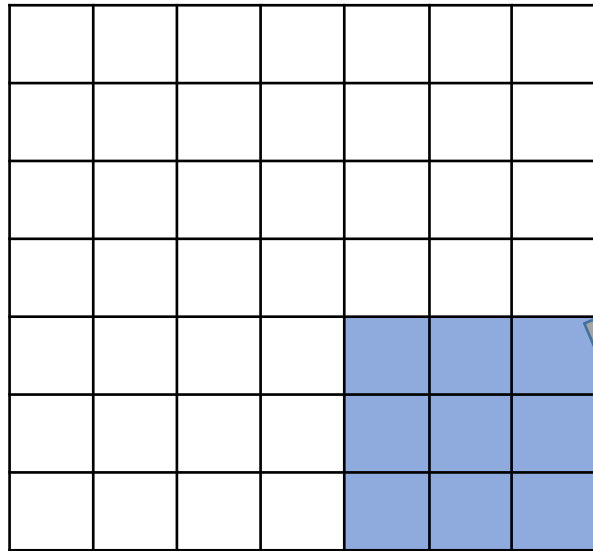
인공지능 컴퓨터 정보공학과

- Convolution 신경망 활용 예제
- 순환 신경망
- 시험 범위 review

컨볼루션 신경망 (CNN: Convolutional Neural Network)

인공지능 컴퓨터 정보공학과

7x7 input image



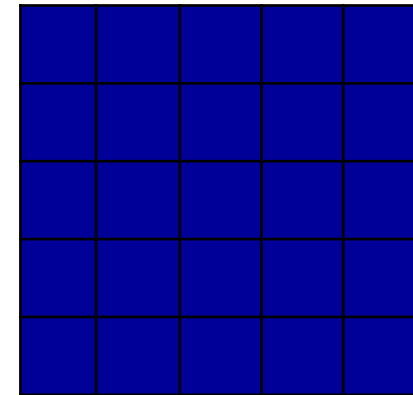
*

3x3 kernel

w00	w01	w02
w10	w11	w21
w20	w21	w22

output image

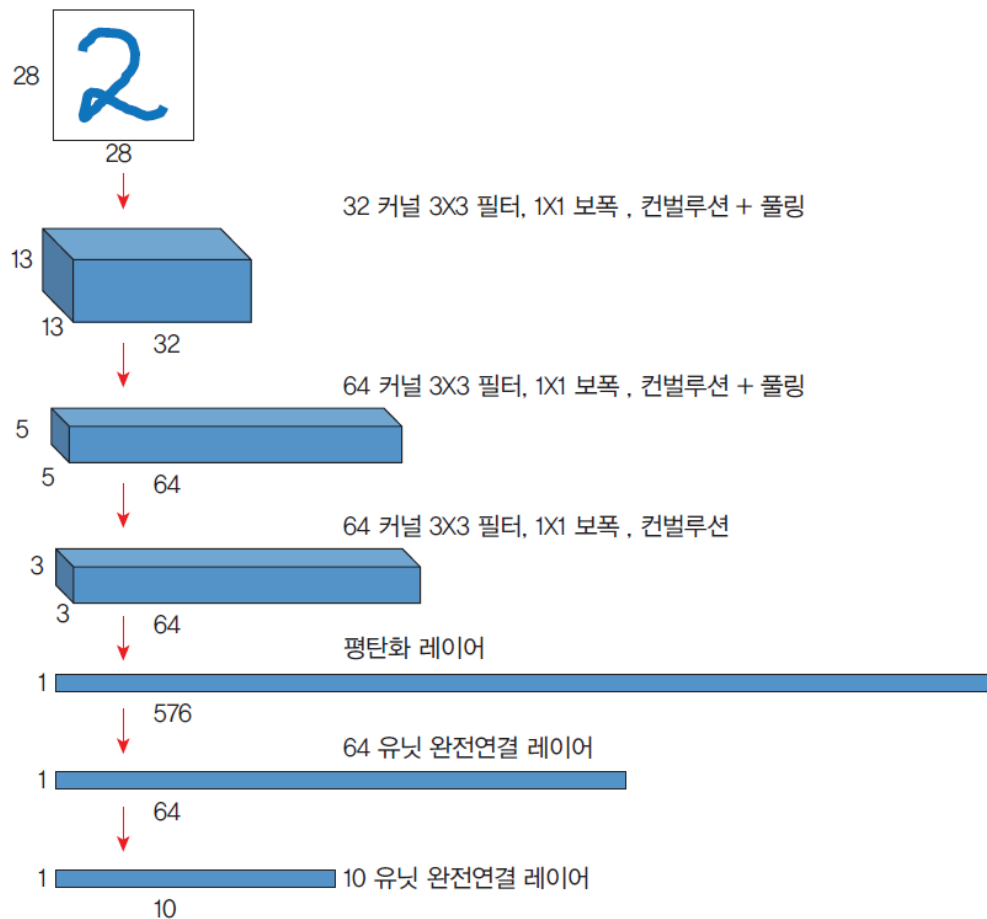
5X5 image



출력 사이즈 = (입력 사이즈 - 커널 사이즈) + 1 = (7 - 3) + 1 = 5



Convolution Neural Network (CNN)



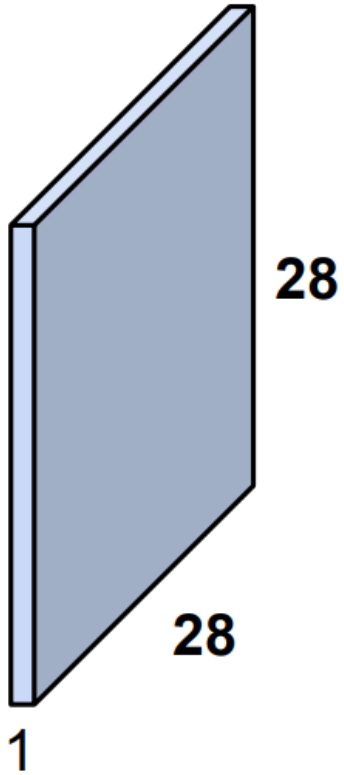
Model: "sequential"

Layer (type)	Output Shape	Param #
=====Model: "sequential"=====		
conv2d (Conv2D)	(None, 26, 26, 32)	320
max_pooling2d (MaxPooling2D)	(None, 13, 13, 32)	0
conv2d_1 (Conv2D)	(None, 11, 11, 64)	18496
max_pooling2d_1 (MaxPooling2D)	(None, 5, 5, 64)	0
flatten (Flatten)	(None, 1600)	0
dropout (Dropout)	(None, 1600)	0
dense (Dense)	(None, 10)	16010
=====		
Total params: 34,826		
Trainable params: 34,826		
Non-trainable params: 0		

Convolution Neural Network (CNN)

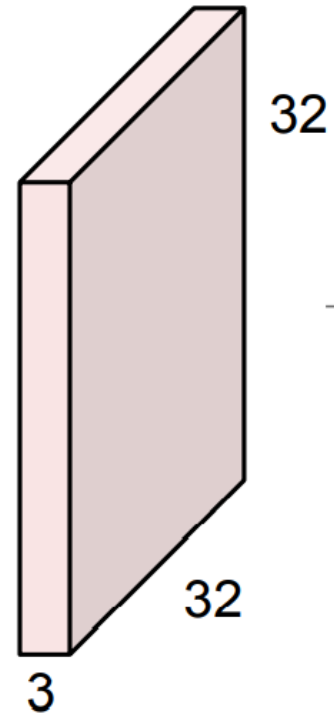
인하공저 컴퓨터 정보과
인하공전 컴퓨터 정보공학과

Input size = (28,28,1)



▪ Output Shape

Input size = (32,32,3)



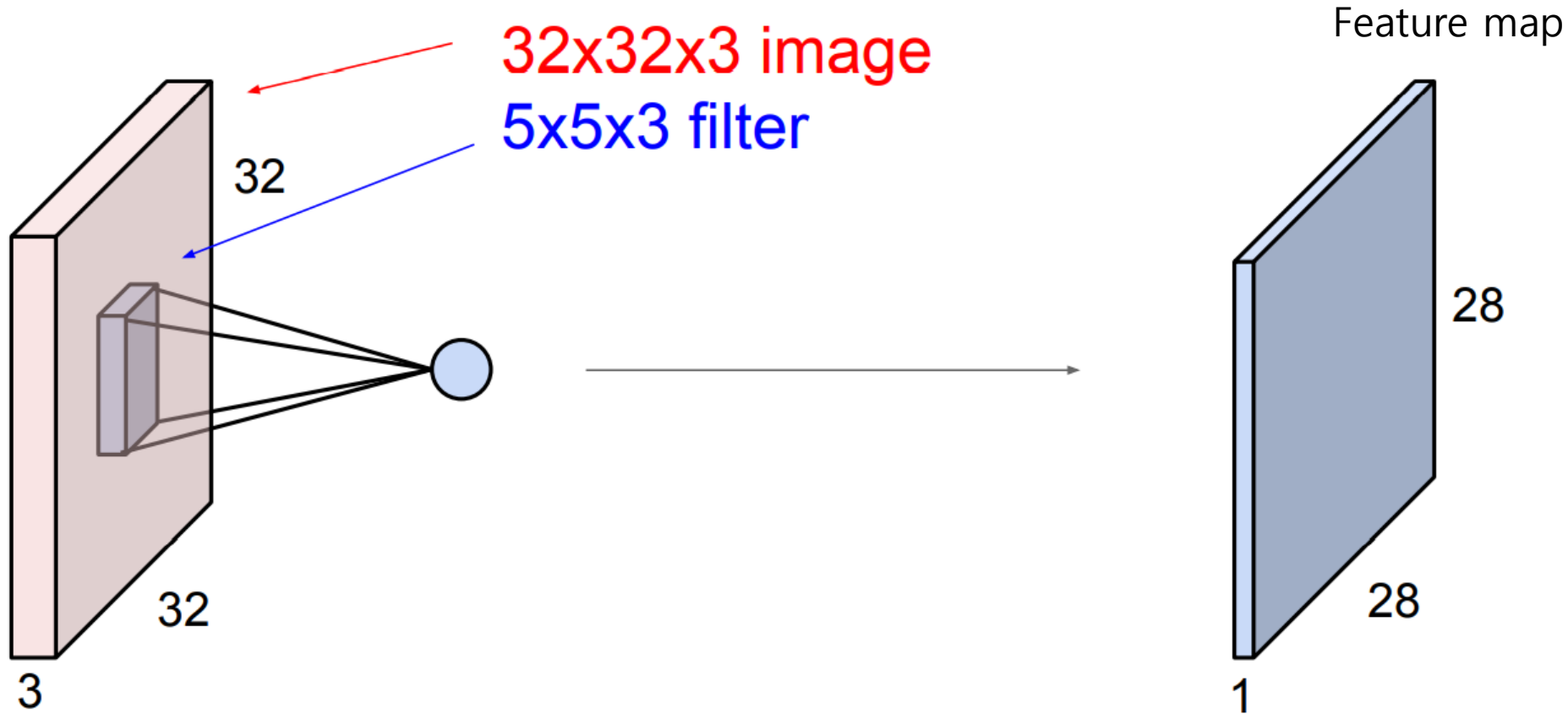
input depth

Convolution Neural Network (CNN)

- 커널(kernel)=필터(filter)
- 필터의 수 = 채널 (channel)
- `tf.keras.layers.Conv2D(filters, kernel_size, strides=(1, 1), activation=None, input_shape, padding='valid')`
 - filters: 필터의 개수이다.

Convolution Neural Network (CNN)

■ Output Shape



출력 사이즈 = (입력 사이즈 - 커널 사이즈) + 1 = 32 - 5 + 1 = 28

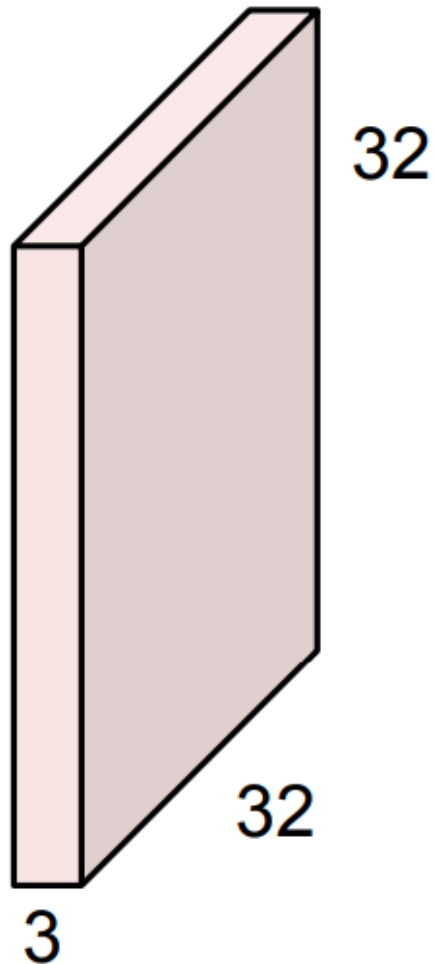
Convolution Neural Network (CNN)

■ Output Shape

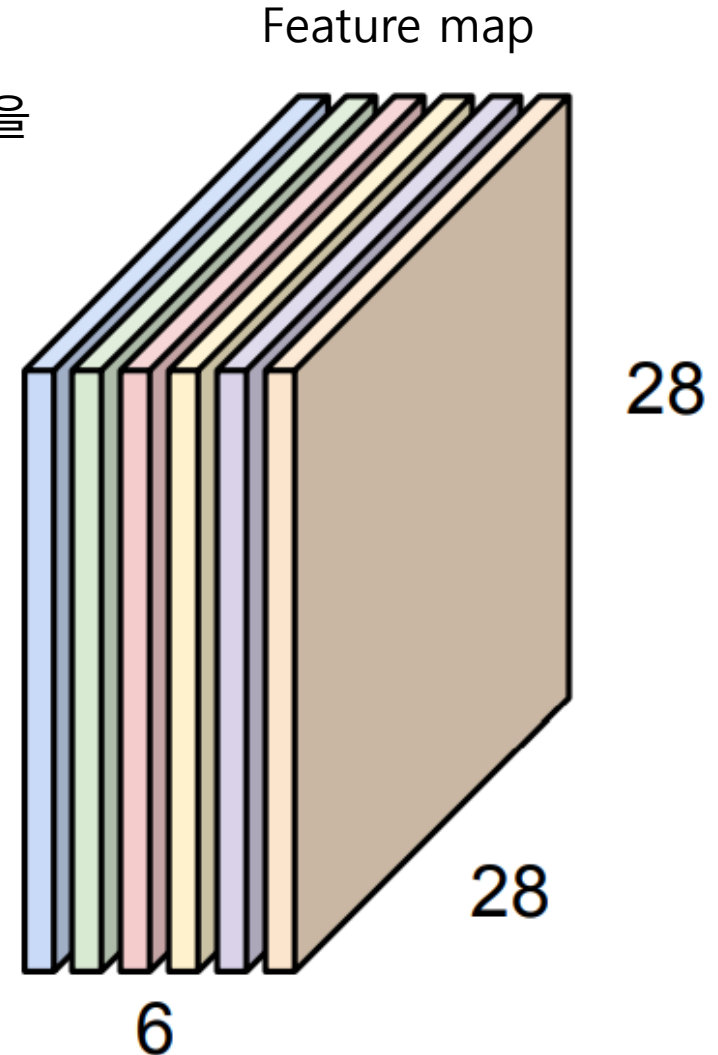


Convolution Neural Network (CNN)

인하공저 컴퓨터 정보과
인하공전 컴퓨터 정보공학과



6개의 filter가 있으면 6개의 feature map을
가지게 됨



$(32, 32, 3) \Rightarrow$ 채널 :6
 $\text{6개의 } 5 \times 5 \text{ filter} \Rightarrow (28, 28, \text{6})$

MNIST 손글씨 이미지 분류- CNN 케라스 구현

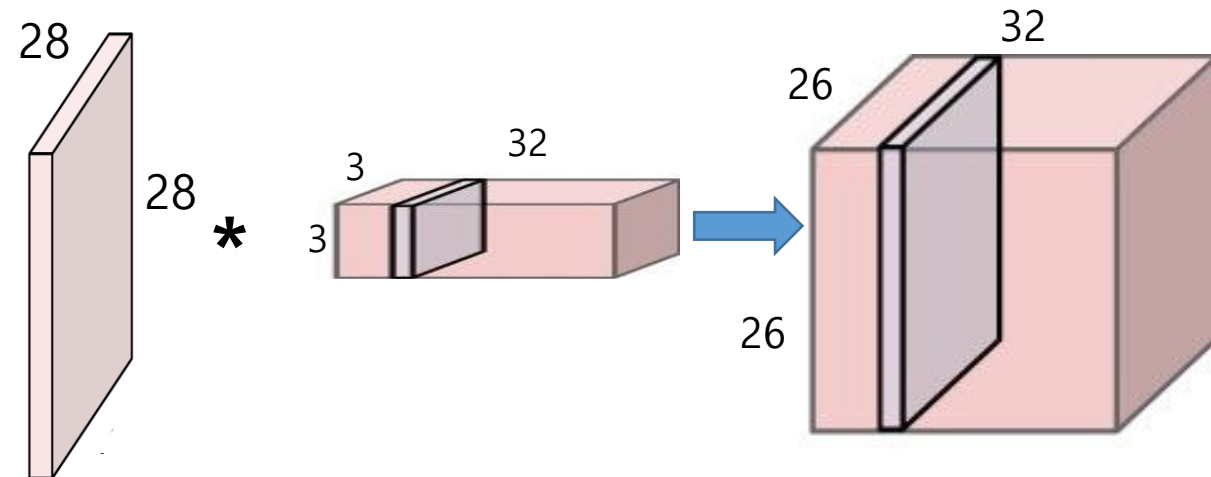
인하공전 컴퓨터 정보공학과

```
from tensorflow import keras
from tensorflow.keras import layers
```

```
input_shape = (28, 28, 1)
```

```
model = keras.Sequential(
    [
        keras.Input(shape=input_shape),
        layers.Conv2D(32, kernel_size=(3, 3), activation="relu"),
        layers.MaxPooling2D(pool_size=(2, 2)),
        layers.Conv2D(64, kernel_size=(3, 3), activation="relu"),
        layers.MaxPooling2D(pool_size=(2, 2)),
        layers.Flatten(),
        layers.Dropout(0.5),
        layers.Dense(num_classes, activation="softmax"),
    ]
)
```

Filter size Filter 개수=32
→ Input=28x28x1, kernel=3x3, channel=32
=> output size=26x26x32



MNIST 손 글씨 이미지 분류- CNN 케라스 구현

인하공전 컴퓨터 정보공학과

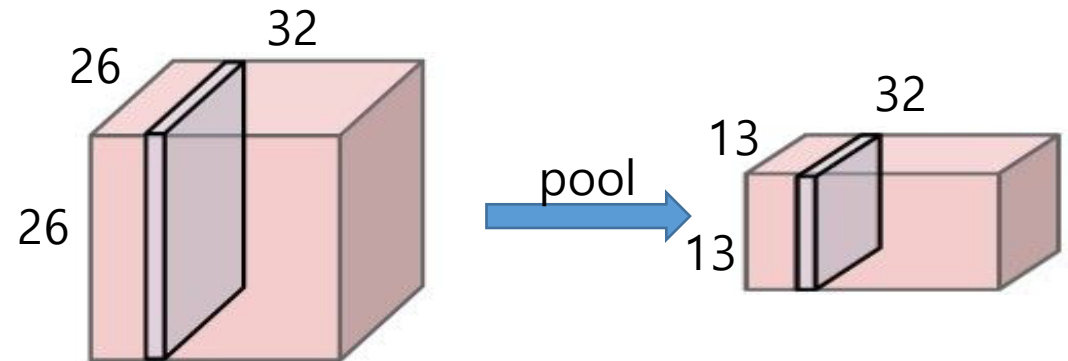
```
from tensorflow import keras  
from tensorflow.keras import layers
```

```
input_shape = (28, 28, 1)
```

```
model = keras.Sequential(  
    [  
        keras.Input(shape=input_shape),  
        layers.Conv2D(32, kernel_size=(3, 3), activation="relu"),  
        layers.MaxPooling2D(pool_size=(2, 2)),  
        layers.Conv2D(64, kernel_size=(3, 3), activation="relu"),  
        layers.MaxPooling2D(pool_size=(2, 2)),  
        layers.Flatten(),  
        layers.Dropout(0.5),  
        layers.Dense(num_classes, activation="softmax"),  
    ]  
)
```



Input=26x26x32, MaxPool =(2,2)
=> output size=13x13x32



MNIST 손 글씨 이미지 분류- CNN 케라스 구현

인하공전 컴퓨터 정보공학과

```
from tensorflow import keras  
from tensorflow.keras import layers
```

```
input_shape = (28, 28, 1)
```

```
model = keras.Sequential(  
    [  
        keras.Input(shape=input_shape),  
        layers.Conv2D(32, kernel_size=(3, 3), activation="relu"),  
        layers.MaxPooling2D(pool_size=(2, 2)),  
        layers.Conv2D(64, kernel_size=(3, 3), activation="relu"),  
        layers.MaxPooling2D(pool_size=(2, 2)),  
        layers.Flatten(),  
        layers.Dropout(0.5),  
        layers.Dense(num_classes, activation="softmax"),  
    ]  
)
```



Input=13x13x32, kernel=3x3, channel=64
=> output size=11x11x64

MNIST 손 글씨 이미지 분류- CNN 케라스 구현

인하공전 컴퓨터 정보공학과

```
from tensorflow import keras
from tensorflow.keras import layers
```

```
input_shape = (28, 28, 1)
```

```
model = keras.Sequential(
    [
        keras.Input(shape=input_shape),
        layers.Conv2D(32, kernel_size=(3, 3), activation="relu"),
        layers.MaxPooling2D(pool_size=(2, 2)),
        layers.Conv2D(64, kernel_size=(3, 3), activation="relu"),
        layers.MaxPooling2D(pool_size=(2, 2)),
        layers.Flatten(),
        layers.Dropout(0.5),
        layers.Dense(num_classes, activation="softmax"),
    ]
)
```

→ Input=11x11x64, MaxPool =(2,2)
=> output size=5x5x64

MNIST 손 글씨 이미지 분류- CNN 케라스 구현

인하공전 컴퓨터 정보공학과

```
from tensorflow import keras  
from tensorflow.keras import layers
```

```
input_shape = (28, 28, 1)
```

```
model = keras.Sequential(  
    [  
        keras.Input(shape=input_shape),  
        layers.Conv2D(32, kernel_size=(3, 3), activation="relu"),  
        layers.MaxPooling2D(pool_size=(2, 2)),  
        layers.Conv2D(64, kernel_size=(3, 3), activation="relu"),  
        layers.MaxPooling2D(pool_size=(2, 2)),  
        layers.Flatten(),  
        layers.Dropout(0.5),  
        layers.Dense(num_classes, activation="softmax"),  
    ]  
)
```

➡ Input=5x5x64, => output size=1600

MNIST 손 글씨 이미지 분류- CNN 케라스 구현

인하공전 컴퓨터 정보공학과

```
from tensorflow import keras  
from tensorflow.keras import layers
```

```
input_shape = (28, 28, 1)
```

```
model = keras.Sequential(  
    [  
        keras.Input(shape=input_shape),  
        layers.Conv2D(32, kernel_size=(3, 3), activation="relu"),  
        layers.MaxPooling2D(pool_size=(2, 2)),  
        layers.Conv2D(64, kernel_size=(3, 3), activation="relu"),  
        layers.MaxPooling2D(pool_size=(2, 2)),  
        layers.Flatten(),  
        layers.Dropout(0.5),  
        layers.Dense(num_classes, activation="softmax"),  
    ]  
)
```

➡ Input=1600, => output size=1600

MNIST 손 글씨 이미지 분류- CNN 케라스 구현

인하공전 컴퓨터 정보공학과

```
from tensorflow import keras
from tensorflow.keras import layers
```

```
input_shape = (28, 28, 1)
```

```
model = keras.Sequential(
    [
        keras.Input(shape=input_shape),
        layers.Conv2D(32, kernel_size=(3, 3), activation="relu"),
        layers.MaxPooling2D(pool_size=(2, 2)),
        layers.Conv2D(64, kernel_size=(3, 3), activation="relu"),
        layers.MaxPooling2D(pool_size=(2, 2)),
        layers.Flatten(),
        layers.Dropout(0.5),
        layers.Dense(num_classes, activation="softmax"),
    ]
)
```

Number of parameters
= input size * number of class + number of class
=> $1600 * 10 + 10 = 16010$

➡ Input=1600, => output size=10

MNIST 손 글씨 이미지 분류- CNN 케라스 구현

인하공전 컴퓨터 정보공학과

```
from tensorflow import keras
from tensorflow.keras import layers
```

```
input_shape = (28, 28, 1)
```

```
model = keras.Sequential(
    [
        keras.Input(shape=input_shape),
        layers.Conv2D(32, kernel_size=(3, 3), activation="relu"),
        layers.MaxPooling2D(pool_size=(2, 2)),
        layers.Conv2D(64, kernel_size=(3, 3), activation="relu"),
        layers.MaxPooling2D(pool_size=(2, 2)),
        layers.Flatten(),
        layers.Dropout(0.5),
        layers.Dense(num_classes, activation="softmax"),
    ]
)
```

Model: "sequential"

Layer (type)	Output Shape	Param #
=====Model: "sequential"		

Layer (type)	Output Shape	Param #
=====		
conv2d (Conv2D)	(None, 26, 26, 32)	320

max_pooling2d (MaxPooling2D)	(None, 13, 13, 32)	0

conv2d_1 (Conv2D)	(None, 11, 11, 64)	18496

max_pooling2d_1 (MaxPooling2D)	(None, 5, 5, 64)	0

flatten (Flatten)	(None, 1600)	0

dropout (Dropout)	(None, 1600)	0

dense (Dense)	(None, 10)	16010

=====

Total params: 34,826

Trainable params: 34,826

Non-trainable params: 0

Convolution Neural Network (CNN)

■ 모델 디자인 (mnist_paratest.ipynb)

(과제1)

1x1 kernel은 사용하지 않음

Layer (type)	Output Shape	Param #
=====		
==		
conv2d_3 (Conv2D)	(None, 26, 26, 16)	160
max_pooling2d_2 (MaxPooling 2D)	(None, 13, 13, 16)	0
conv2d_4 (Conv2D)	(None, 13, 13, 32)	4640
max_pooling2d_3 (MaxPooling 2D)	(None, 6, 6, 32)	0
conv2d_5 (Conv2D)	(None, 4, 4, 64)	18496
flatten_1 (Flatten)	(None, 1024)	0
dense_2 (Dense)	(None, 64)	65600
dense_3 (Dense)	(None, 10)	650
=====		
==		

Convolution Neural Network (CNN)

■ 모델 디자인 (mnist_paratest.ipynb)

(과제1)

```
input_shape=(28,28,1)

model.add(layers.Input(shape=input_shape))
model.add(layers.Conv2D(16, (3, 3), activation='relu'))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Conv2D(32, (3, 3), activation='relu',padding='same'))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Conv2D(64, (3, 3), activation='relu'))

model.add(layers.Flatten())
model.add(layers.Dense(64, activation='relu'))
model.add(layers.Dense(10, activation='softmax'))
```

Layer (type)	Output Shape	Param #
=====		
==		
conv2d_3 (Conv2D)	(None, 26, 26, 16)	160
max_pooling2d_2 (MaxPooling 2D)	(None, 13, 13, 16)	0
conv2d_4 (Conv2D)	(None, 13, 13, 32)	4640
max_pooling2d_3 (MaxPooling 2D)	(None, 6, 6, 32)	0
conv2d_5 (Conv2D)	(None, 4, 4, 64)	18496
flatten_1 (Flatten)	(None, 1024)	0
dense_2 (Dense)	(None, 64)	65600
dense_3 (Dense)	(None, 10)	650
=====		
==		

Convolution Neural Network (CNN)

■ 모델 디자인 (mnist_paratest.ipynb)

(과제2)

) 1x1 kernel(filter)은 사용할 수 없음.

Layer (type)	Output Shape	Param #
=====	=====	=====
conv2d (Conv2D)	(None, 28, 28, 32)	320
max_pooling2d (MaxPooling2D)	(None, 14, 14, 32)	0
conv2d_1 (Conv2D)	(None, 10, 10, 64)	51264
max_pooling2d_1 (MaxPooling 2D)	(None, 5, 5, 64)	0
conv2d_2 (Conv2D)	(None, 3, 3, 128)	73856
flatten (Flatten)	(None, 1152)	0
dense (Dense)	(None, 64)	73792
dense_1 (Dense)	(None, 10)	650

Convolution Neural Network (CNN)

■ 모델 디자인 (mnist_paratest.ipynb)

(과제2)

```
input_shape=(28,28,1)

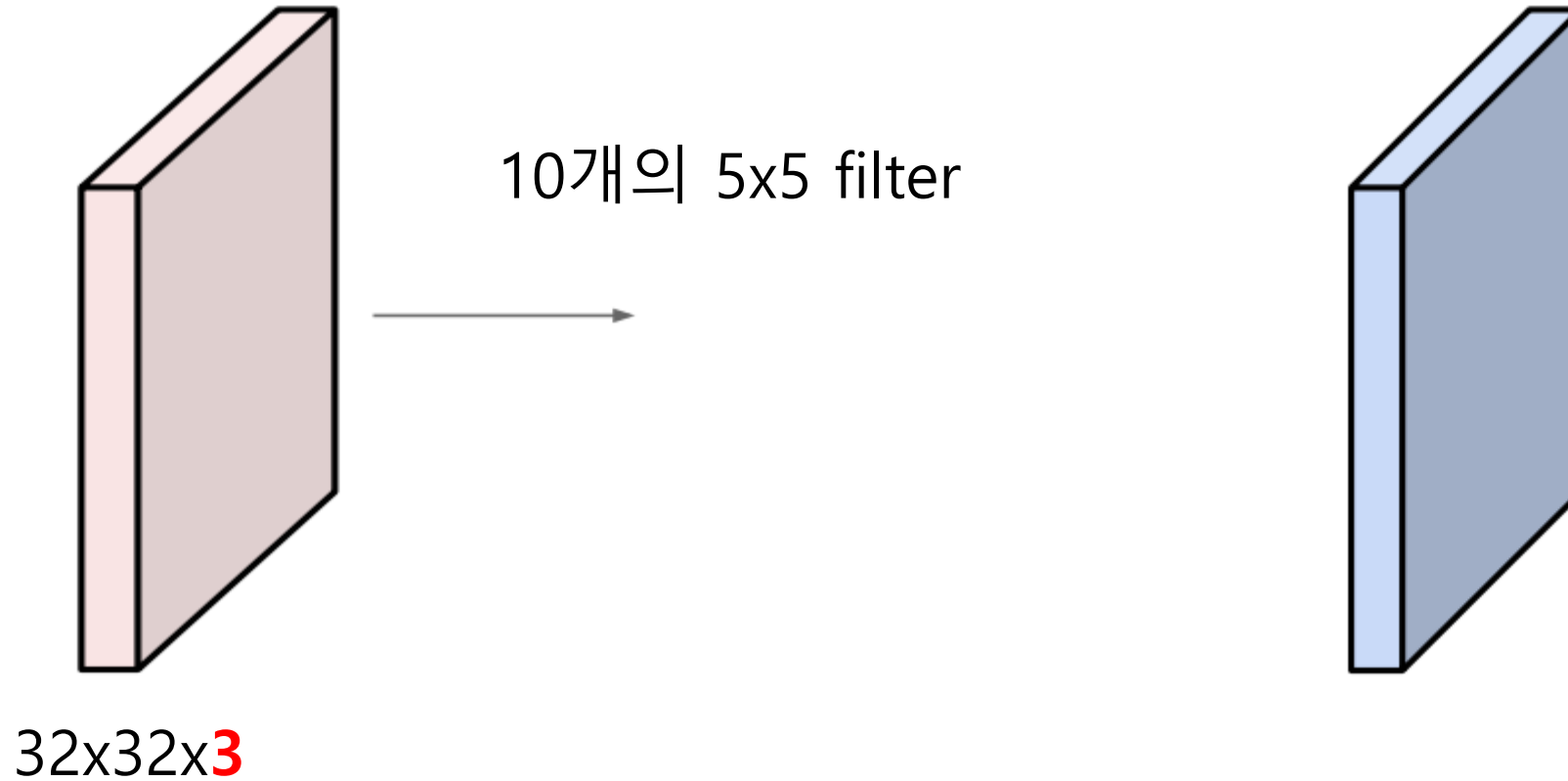
model.add(layers.Input(shape=input_shape))
model.add(layers.Conv2D(32, (3, 3),
activation='relu',padding='same'))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Conv2D(64, (5, 5), activation='relu'))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Conv2D(128, (3, 3), activation='relu'))

model.add(layers.Flatten())
model.add(layers.Dense(64, activation='relu'))
model.add(layers.Dense(10, activation='softmax'))
```

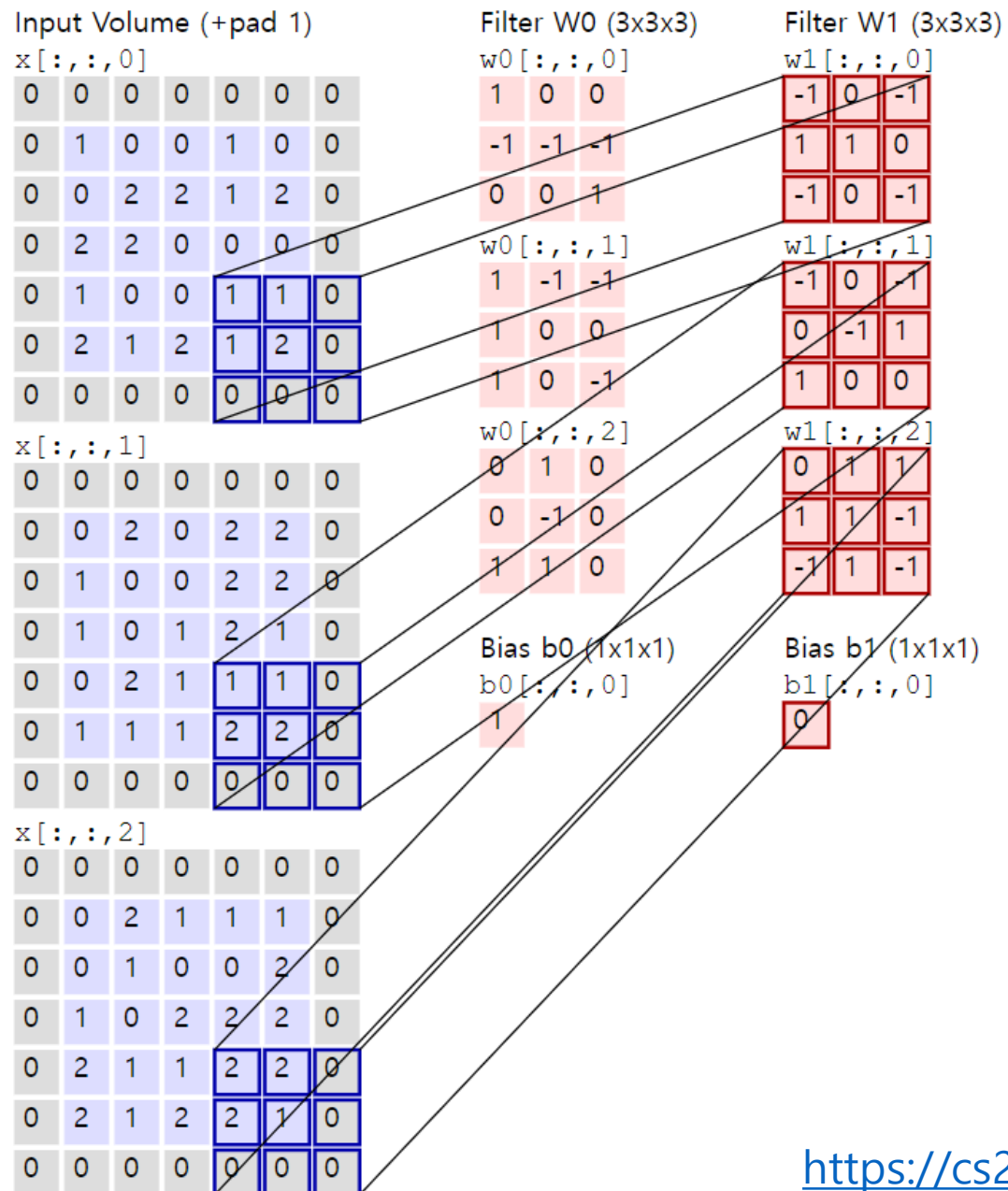
Layer (type)	Output Shape	Param #
=====		
conv2d (Conv2D)	(None, 28, 28, 32)	320
max_pooling2d (MaxPooling2D)	(None, 14, 14, 32)	0
conv2d_1 (Conv2D)	(None, 10, 10, 64)	51264
max_pooling2d_1 (MaxPooling 2D)	(None, 5, 5, 64)	0
conv2d_2 (Conv2D)	(None, 3, 3, 128)	73856
flatten (Flatten)	(None, 1152)	0
dense (Dense)	(None, 64)	73792
dense_1 (Dense)	(None, 10)	650

Convolution Layer Parameter

인하공전 컴퓨터 정보공학과

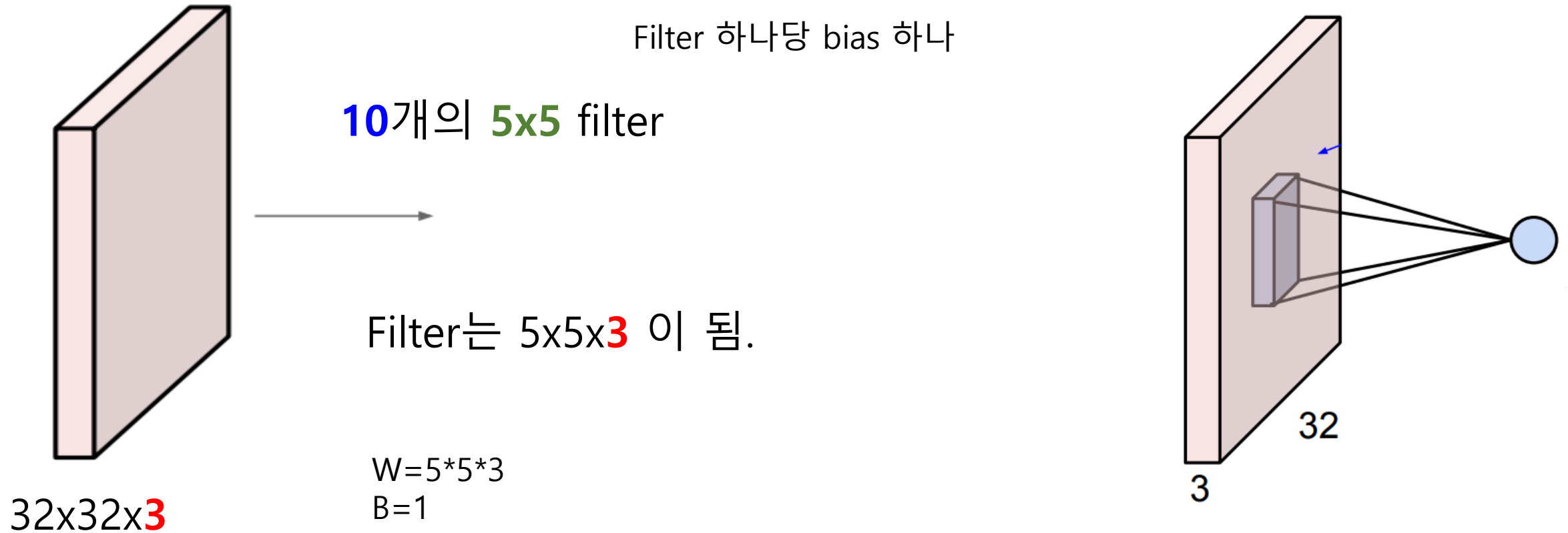


Filter는 5x5x3



Convolution Layer Parameter

인하공전 컴퓨터 정보공학과



Filter 하나당 parameter 5x5x3+1

Filter 가 10개

$$10 * (5 * 5 * 3 + 1) = 760$$

채널이 10개

이 layer의 parameter 수는 760

MNIST 손 글씨 이미지 분류- CNN 케라스 구현

인하공전 컴퓨터 정보공학과

```
from tensorflow import keras  
from tensorflow.keras import layers
```

```
input_shape = (28, 28, 1)
```

```
model = keras.Sequential(  
    [  
        keras.Input(shape=input_shape),  
        layers.Conv2D(32, kernel_size=(3, 3), activation="relu"),  
        layers.MaxPooling2D(pool_size=(2, 2)),  
        layers.Conv2D(64, kernel_size=(3, 3), activation="relu"),  
        layers.MaxPooling2D(pool_size=(2, 2)),  
        layers.Flatten(),  
        layers.Dropout(0.5),  
        layers.Dense(num_classes, activation="softmax"),  
    ]  
)
```

MNIST 손글씨 이미지 분류- CNN 케라스 구현

인하공전 컴퓨터 정보공학과

```
from tensorflow import keras
from tensorflow.keras import layers
```

```
input_shape = (28, 28, 1)
```

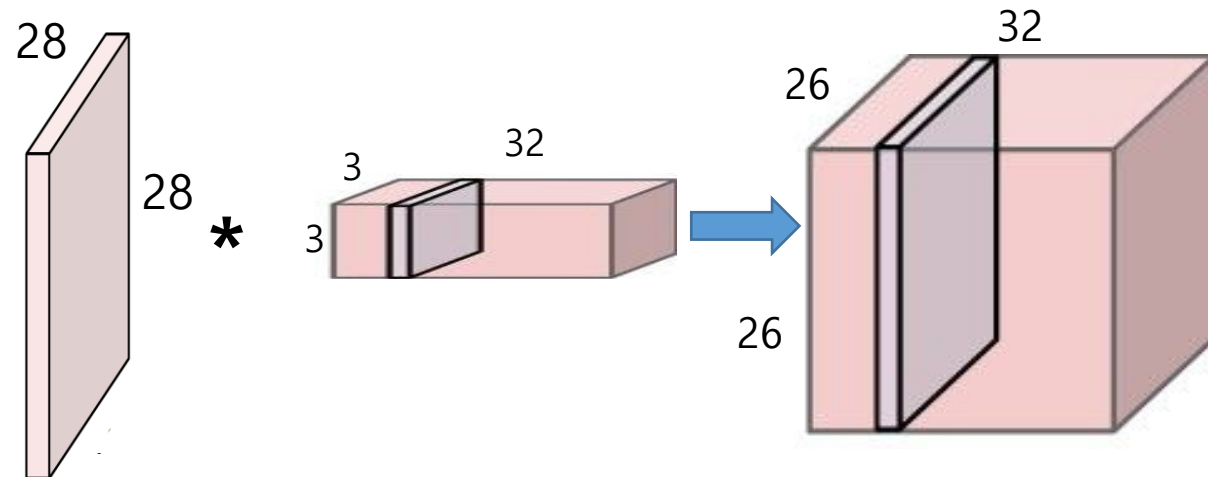
```
model = keras.Sequential(
    [
        keras.Input(shape=input_shape),
        layers.Conv2D(32, kernel_size=(3, 3), activation="relu"),
        layers.MaxPooling2D(pool_size=(2, 2)),
        layers.Conv2D(64, kernel_size=(3, 3), activation="relu"),
        layers.MaxPooling2D(pool_size=(2, 2)),
        layers.Flatten(),
        layers.Dropout(0.5),
        layers.Dense(num_classes, activation="softmax"),
    ]
)
```

Number of parameters0

=채널*(input depth*kernel size*kernel size+1

=>32*(1*3*3+1)=320

→ Input=28x28x1, kernel=3x3, channel=32
=> output size=26x26x32



MNIST 손글씨 이미지 분류- CNN 케라스 구현

인하공전 컴퓨터 정보공학과

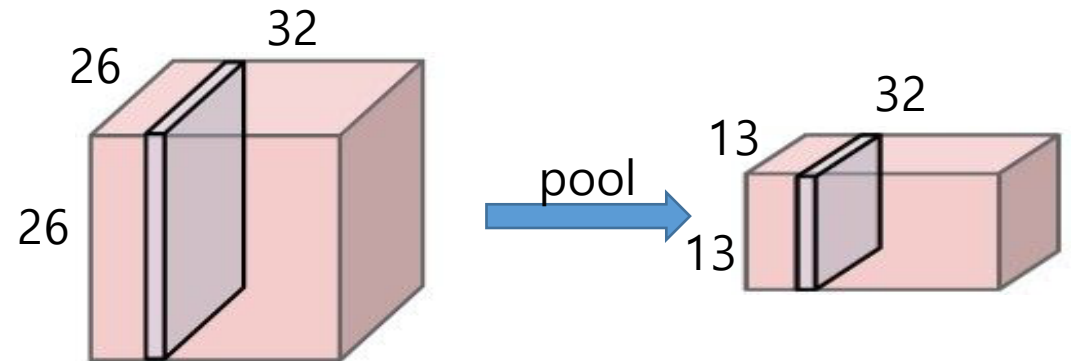
```
from tensorflow import keras  
from tensorflow.keras import layers
```

```
input_shape = (28, 28, 1)
```

```
model = keras.Sequential(  
    [  
        keras.Input(shape=input_shape),  
        layers.Conv2D(32, kernel_size=(3, 3), activation="relu"),  
        layers.MaxPooling2D(pool_size=(2, 2)),  
        layers.Conv2D(64, kernel_size=(3, 3), activation="relu"),  
        layers.MaxPooling2D(pool_size=(2, 2)),  
        layers.Flatten(),  
        layers.Dropout(0.5),  
        layers.Dense(num_classes, activation="softmax"),  
    ]  
)
```



Input=26x26x32, MaxPool =(2,2)
=> output size=13x13x32



MNIST 손 글씨 이미지 분류- CNN 케라스 구현

인하공전 컴퓨터 정보공학과

```
from tensorflow import keras  
from tensorflow.keras import layers
```

```
input_shape = (28, 28, 1)
```

```
model = keras.Sequential(  
    [  
        keras.Input(shape=input_shape),  
        layers.Conv2D(32, kernel_size=(3, 3), activation="relu"),  
        layers.MaxPooling2D(pool_size=(2, 2)),  
        layers.Conv2D(64, kernel_size=(3, 3), activation="relu"),  
        layers.MaxPooling2D(pool_size=(2, 2)),  
        layers.Flatten(),  
        layers.Dropout(0.5),  
        layers.Dense(num_classes, activation="softmax"),  
    ]  
)
```

Number of parameters

=채널*(input depth*kernel size*kernel size+1)

=> $64 \times (32 \times 3 \times 3 + 1) = 18496$



Input=13x13x32, kernel=3x3, channel=64
=> output size=11x11x64

MNIST 손 글씨 이미지 분류- CNN 케라스 구현

인하공전 컴퓨터 정보공학과

```
from tensorflow import keras
from tensorflow.keras import layers
```

```
input_shape = (28, 28, 1)
```

```
model = keras.Sequential(
    [
        keras.Input(shape=input_shape),
        layers.Conv2D(32, kernel_size=(3, 3), activation="relu"),
        layers.MaxPooling2D(pool_size=(2, 2)),
        layers.Conv2D(64, kernel_size=(3, 3), activation="relu"),
        layers.MaxPooling2D(pool_size=(2, 2)),
        layers.Flatten(),
        layers.Dropout(0.5),
        layers.Dense(num_classes, activation="softmax"),
    ]
)
```

→ Input=11x11x64, MaxPool =(2,2)
=> output size=5x5x64

MNIST 손 글씨 이미지 분류- CNN 케라스 구현

인하공전 컴퓨터 정보공학과

```
from tensorflow import keras  
from tensorflow.keras import layers
```

```
input_shape = (28, 28, 1)
```

```
model = keras.Sequential(  
    [  
        keras.Input(shape=input_shape),  
        layers.Conv2D(32, kernel_size=(3, 3), activation="relu"),  
        layers.MaxPooling2D(pool_size=(2, 2)),  
        layers.Conv2D(64, kernel_size=(3, 3), activation="relu"),  
        layers.MaxPooling2D(pool_size=(2, 2)),  
        layers.Flatten(),  
        layers.Dropout(0.5),  
        layers.Dense(num_classes, activation="softmax"),  
    ]  
)
```

➡ Input=5x5x64, => output size=1600

MNIST 손 글씨 이미지 분류- CNN 케라스 구현

인하공전 컴퓨터 정보공학과

```
from tensorflow import keras  
from tensorflow.keras import layers
```

```
input_shape = (28, 28, 1)
```

```
model = keras.Sequential(  
    [  
        keras.Input(shape=input_shape),  
        layers.Conv2D(32, kernel_size=(3, 3), activation="relu"),  
        layers.MaxPooling2D(pool_size=(2, 2)),  
        layers.Conv2D(64, kernel_size=(3, 3), activation="relu"),  
        layers.MaxPooling2D(pool_size=(2, 2)),  
        layers.Flatten(),  
        layers.Dropout(0.5),  
        layers.Dense(num_classes, activation="softmax"),  
    ]  
)
```

➡ Input=1600, => output size=1600

MNIST 손 글씨 이미지 분류- CNN 케라스 구현

인하공전 컴퓨터 정보공학과

```
from tensorflow import keras
from tensorflow.keras import layers
```

```
input_shape = (28, 28, 1)
```

```
model = keras.Sequential(
    [
        keras.Input(shape=input_shape),
        layers.Conv2D(32, kernel_size=(3, 3), activation="relu"),
        layers.MaxPooling2D(pool_size=(2, 2)),
        layers.Conv2D(64, kernel_size=(3, 3), activation="relu"),
        layers.MaxPooling2D(pool_size=(2, 2)),
        layers.Flatten(),
        layers.Dropout(0.5),
        layers.Dense(num_classes, activation="softmax"),
    ]
)
```

Number of parameters

= 앞 layer node 수 * 현재 layer node 수 + 현재 layer node 수

=> $1600 * 10 + 10 = 16010$

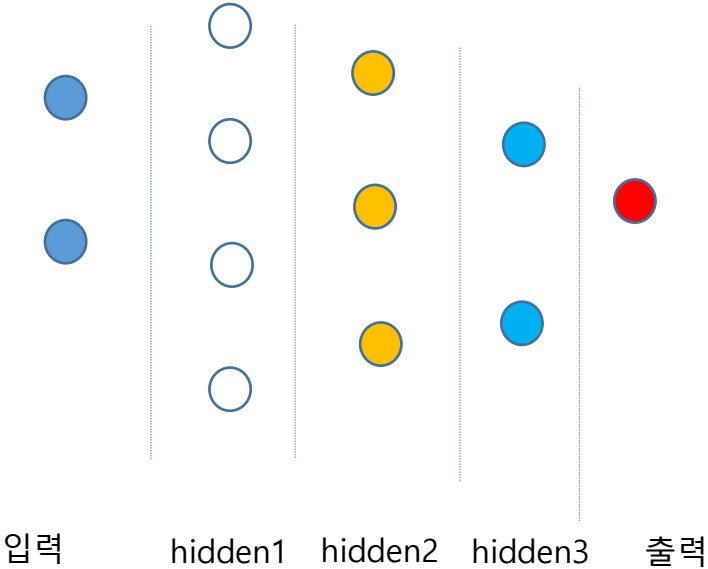
➡ Input=1600, => output size=10

```
model = tf.keras.models.Sequential()  
model.add(tf.keras.layers.Dense(4, activation='relu', input_shape=(2,)))  
model.add(tf.keras.layers.Dense(3, activation='relu'))  
model.add(tf.keras.layers.Dense(2, activation='relu'))  
model.add(tf.keras.layers.Dense(1, activation='sigmoid'))
```

입력 층 : 2
은닉층 1 : 4 (relu)
은닉층 2 : 3 (relu)
은닉층 3 : 2 (relu)
출력 층 : 1

Layer (type)	Output Shape	Param #
dense_4 (Dense)	(None, 4)	12
dense_5 (Dense)	(None, 3)	15
dense_6 (Dense)	(None, 2)	8
dense_7 (Dense)	(None, 1)	3

total params: 38
Trainable params: 38
Non-trainable params: 0



Parameter 수 = 앞 layer node수* 현재 layer node수 + 현재 layer node 수

MNIST 손 글씨 이미지 분류- CNN 케라스 구현

인하공전 컴퓨터 정보공학과

```
from tensorflow import keras
from tensorflow.keras import layers
```

```
input_shape = (28, 28, 1)
```

```
model = keras.Sequential(
    [
        keras.Input(shape=input_shape),
        layers.Conv2D(32, kernel_size=(3, 3), activation="relu"),
        layers.MaxPooling2D(pool_size=(2, 2)),
        layers.Conv2D(64, kernel_size=(3, 3), activation="relu"),
        layers.MaxPooling2D(pool_size=(2, 2)),
        layers.Flatten(),
        layers.Dropout(0.5),
        layers.Dense(num_classes, activation="softmax"),
    ]
)
```

Model: "sequential"

Layer (type)	Output Shape	Param #
=====Model: "sequential"		

Layer (type)	Output Shape	Param #
=====		
conv2d (Conv2D)	(None, 26, 26, 32)	320

max_pooling2d (MaxPooling2D)	(None, 13, 13, 32)	0

conv2d_1 (Conv2D)	(None, 11, 11, 64)	18496

max_pooling2d_1 (MaxPooling2D)	(None, 5, 5, 64)	0

flatten (Flatten)	(None, 1600)	0

dropout (Dropout)	(None, 1600)	0

dense (Dense)	(None, 10)	16010

=====

Total params: 34,826

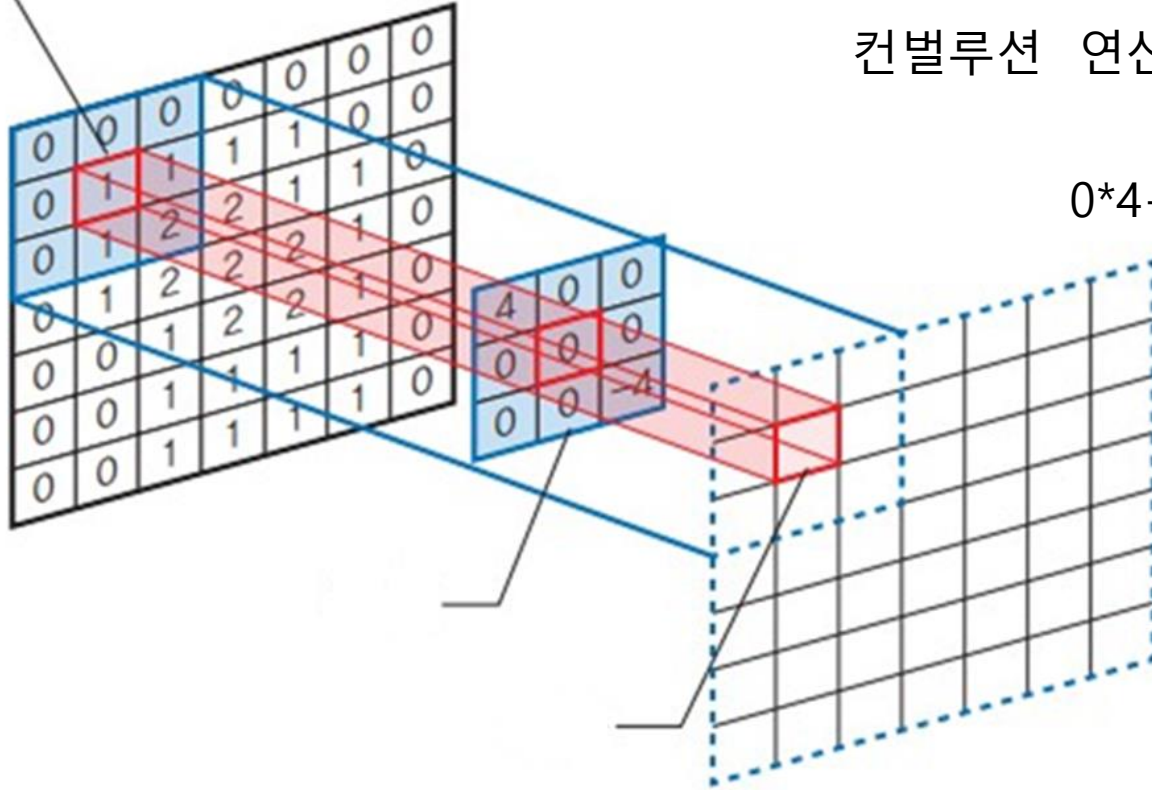
Trainable params: 34,826

Non-trainable params: 0

컨볼루션 (Convolution) 계산

인하공전 컴퓨터 정보공학과

입력층



컨볼루션 연산 수행 결과

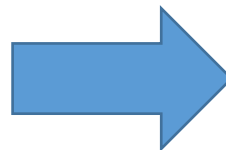
$$0*4+0*0+0*0+1*0+1*0+0*0+1*0+2*-4= -8$$

컨벌루션 (Convolution) 계산

다음과 같은 입력에서 (3,3) 커널과 valid 패딩으로 컨벌루션을 수행합니다. 컨벌루션의 결과를 계산해 보시오

3	0	9	1	2
5	1	2	0	7
8	2	4	1	3
2	1	5	3	6
4	1	6	2	7

2	0	1
2	0	1
2	0	1



47	8	42
41	12	38
43	14	46

$$3*2+5*2+8*2+9+2+4=6+10+16+9+2+4=32+15=47$$

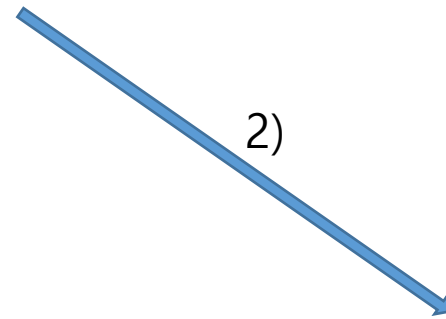
풀링 (Pooling) 계산

- 1) 다음의 feature 맵의 (2,2) 최대 풀링(Max Pooling) 결과를 구하시오
- 2) 다음의 feature 맵의 (2,2) 평균 풀링(Average Pooling) 결과를 구하시오

9	10	6	2
4	1	4	8
8	0	7	1
8	8	3	1



10	8
8	7



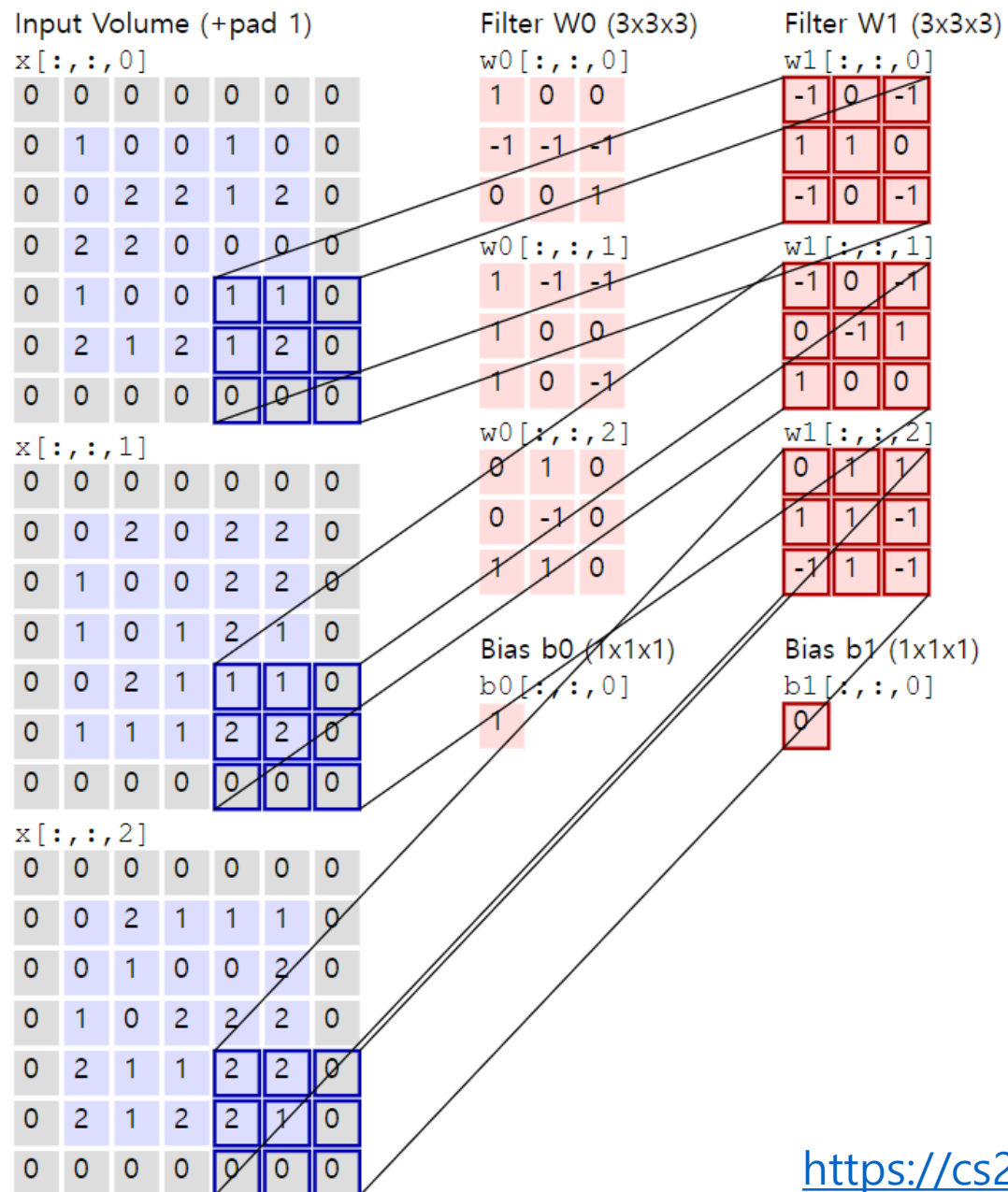
6	5
8	3

컨볼루션 (Convolution) 계산











인하공전 컴퓨터 정보공학과

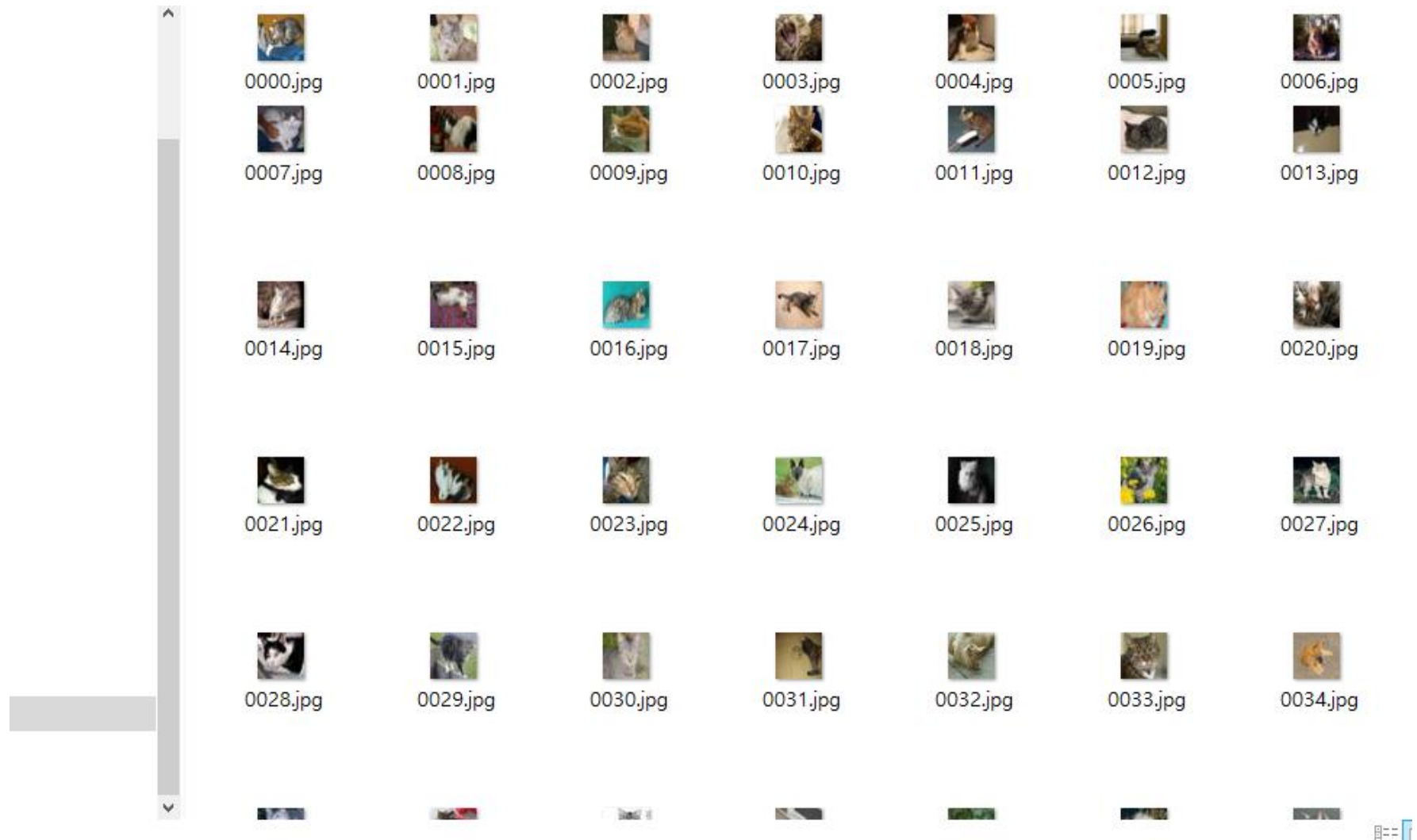
8x8 컬러 영상에 5개의 필터와 'same' padding으로 컨볼루션 연산을 수행 하고 (2,2) 풀링을 통과한 특성 맵의 크기는?

$(8,8,3) \Rightarrow (8,8,5) \Rightarrow (4,4,5)$



■ JPG IMAGE 이용

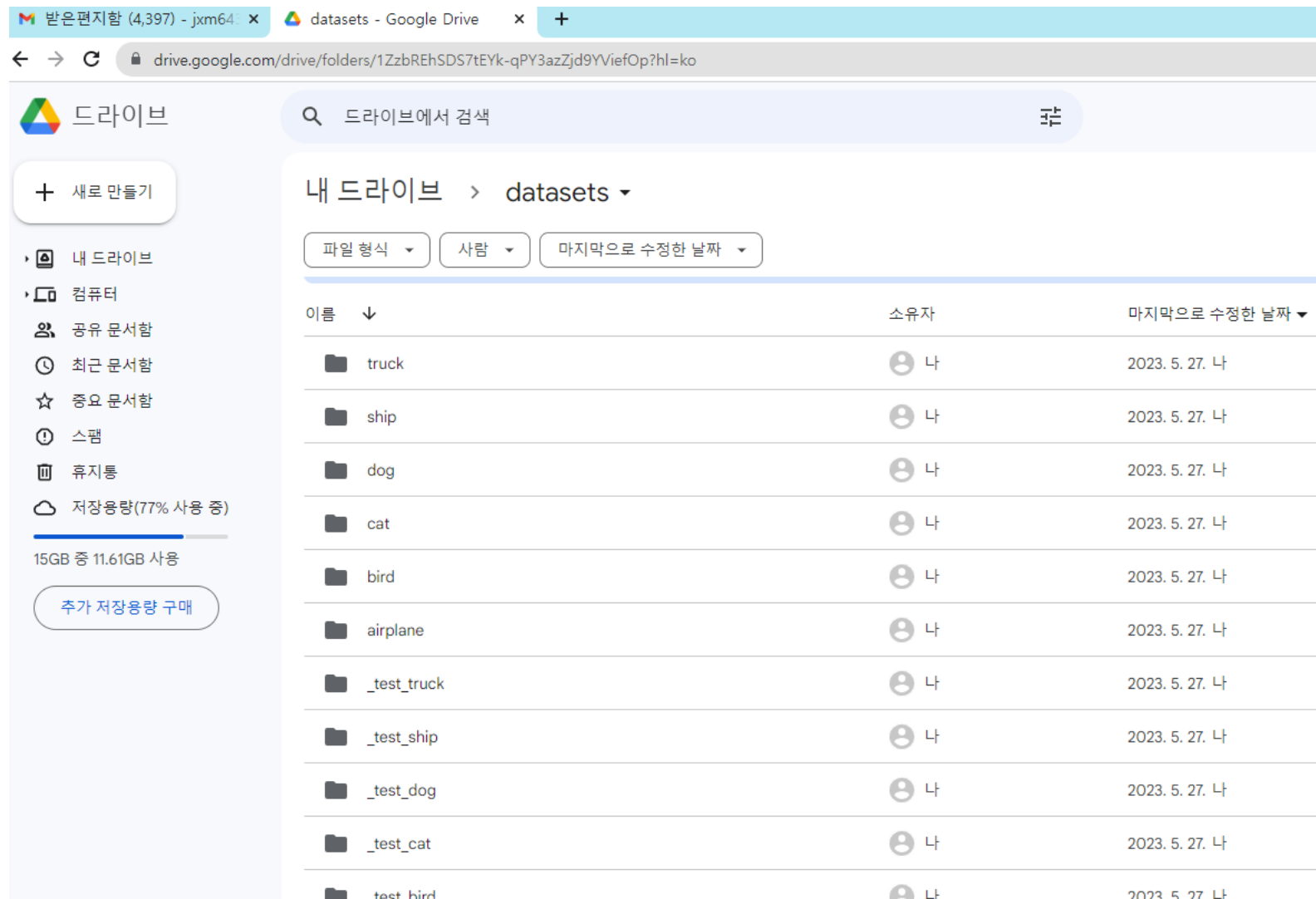
	airplane	수정한 날짜: 2023-05-27 오후 8:45
	bird	수정한 날짜: 2023-05-27 오후 8:45
	car	수정한 날짜: 2023-05-27 오후 8:46
	cat	수정한 날짜: 2023-05-27 오후 8:53
	deer	수정한 날짜: 2023-05-27 오후 8:54
	dog	수정한 날짜: 2023-05-27 오후 8:54
	frog	수정한 날짜: 2023-05-27 오후 8:55
	horse	수정한 날짜: 2023-05-27 오후 8:55
	ship	수정한 날짜: 2023-05-27 오후 8:56
	truck	수정한 날짜: 2023-05-27 오후 8:56



영상 분류 예제

인하공전 컴퓨터 정보공학과


영상 구글 드라이브 upload



The screenshot shows a Google Drive interface with a folder named 'datasets'. The left sidebar shows the '내 드라이브' (My Drive) section with a storage usage bar indicating 15GB total and 11.61GB used. The main area displays a list of folders within 'datasets'.

이름	소유자	마지막으로 수정한 날짜
truck	나	2023. 5. 27. 나
ship	나	2023. 5. 27. 나
dog	나	2023. 5. 27. 나
cat	나	2023. 5. 27. 나
bird	나	2023. 5. 27. 나
airplane	나	2023. 5. 27. 나
_test_truck	나	2023. 5. 27. 나
_test_ship	나	2023. 5. 27. 나
_test_dog	나	2023. 5. 27. 나
_test_cat	나	2023. 5. 27. 나
_test_bird	나	2023. 5. 27. 나






■ Google drive mount


 3classification.ipynb ☆

파일 수정 보기 삽입 런타임 도구 도움말 모든 변경사항이 저장됨

댓글

파일


{x}  ..

drive

MyDrive

sample_data

+ 코드 + 텍스트

 `from google.colab import drive
drive.mount('/content/drive')`

24 초

Mounted at /content/drive

[] `from keras.models import Sequential
from keras.layers import Dense
from keras.layers import Dropout
from keras.layers import Flatten
from keras.layers import Activation, Dropout`

영상 분류 예제

인하공전 컴퓨터 정보공학과

```
dir_path = '/content/drive/MyDrive/datasets'
```

```
categories = ['airplane','truck','bird','cat']
```

```
data = []
```

```
for i in categories:
```

```
    paths = os.path.join(dir_path,i)
```

```
    print('paths',paths)
```

```
    ii=0
```

```
    for j in os.listdir(paths):
```

```
        img_path = os.path.join(paths,j)
```

```
        labels = categories.index(i)
```

```
        print('image_pathlabels',img_path,labels)
```

```
        if (os.path.splitext(img_path)[1]==' .jpg'):
```

```
            img = cv2.imread(img_path)
```

```
            img = cv2.resize(img,(32,32))
```

```
            data.append([img,labels])
```

```
random.shuffle(data)
```

```
print(len(data))
```

paths /content/drive/MyDrive/datasets/airplane










/content/drive/MyDrive/datasets/truck/0016.jpg 1

영상 분류 예제

인하공전 컴퓨터 정보공학과

```
categories = ['airplane','truck','bird','cat']
```

0 1 2 3

data	img									
	labels	1	0	2	1	3	2	1	0	2

Train image : 카테고리 별로 20장

Test image : 카테고리 별로 5장

영상 분류 예제

인하공전 컴퓨터 정보공학과

```
x = []
y = []

for features,label in data:
    x.append(features)
    y.append(label)

#Converting lists into numpy arrays
x = np.array(x)
y = np.array(y)
x = x/255.0
x = np.array(x).reshape(-1, 32, 32, 3)
print("Shape of train images is:", x.shape)
print("Shape of labels is:", y.shape)
print(y)
print(x.shape[1:])
```

Shape of train images is: (80, 32, 32, 3)

Shape of labels is: (80,)

```
[3 2 3 0 0 1 3 1 3 2 3 2 2 1 1 2 2 2 1 1 3 0 1 0 0 2 0 3
 3 3 0 0 1 2 0 2 2
 0 1 2 0 0 2 3 2 3 1 0 1 2 1 2 1 3 1 0 3 0 2 2 0 2 1 1 3
 1 1 0 2 0 3 1 3 3
 1 3 0 0 3 3]
(32, 32, 3)
```

```
model = Sequential()  
model.add(Conv2D(64, (3, 3), activation='relu', input_shape =  
    (x.shape[1:])))  
model.add(MaxPooling2D(pool_size=(2, 2)))  
  
model.add(Conv2D(64, (3, 3), activation='relu'))  
model.add(MaxPooling2D(pool_size=(2, 2)))  
  
model.add(Conv2D(128, (3, 3), activation='relu'))  
model.add(MaxPooling2D(pool_size=(2, 2)))  
model.add(Dropout(0.5))  
model.add(Flatten())  
  
model.add(Dense(128, activation='relu'))  
model.add(Dropout(0.5))  
model.add(Dense(4, activation='softmax'))  
model.summary()
```

Layer (type)	Output Shape	Param #
=====		
conv2d_3 (Conv2D)	(None, 30, 30, 64)	1792
max_pooling2d_3 (MaxPooling2D)	(None, 15, 15, 64)	0
conv2d_4 (Conv2D)	(None, 13, 13, 64)	36928
max_pooling2d_4 (MaxPooling2D)	(None, 6, 6, 64)	0
conv2d_5 (Conv2D)	(None, 4, 4, 128)	73856
max_pooling2d_5 (MaxPooling2D)	(None, 2, 2, 128)	0
dropout_2 (Dropout)	(None, 2, 2, 128)	0
flatten_1 (Flatten)	(None, 512)	0
dense_2 (Dense)	(None, 128)	65664
dropout_3 (Dropout)	(None, 128)	0
dense_3 (Dense)	(None, 4)	516

```
model.compile(optimizer='adam',  
              loss='sparse_categorical_crossentropy',  
              metrics=['accuracy'])  
model.fit(x, y, epochs=60, batch_size=50, validation_split=0.2)
```

```
Epoch 1/60  
5/5 [=====] - 2s 182ms/step - loss: 1.1215 -  
accuracy: 0.3167 - val_loss: 1.0945 - val_accuracy: 0.3000  
Epoch 2/60  
5/5 [=====] - 1s 116ms/step - loss: 1.1089 -  
accuracy: 0.3583 - val_loss: 1.0947 - val_accuracy: 0.3500  
Epoch 3/60  
5/5 [=====] - 1s 110ms/step - loss: 1.0879 -  
accuracy: 0.3958 - val_loss: 1.0857 - val_accuracy: 0.4167  
Epoch 4/60  
5/5 [=====] - 1s 168ms/step - loss: 1.0916 -  
accuracy: 0.3583 - val_loss: 1.0818 - val_accuracy: 0.5333  
Epoch 5/60  
5/5 [=====] - 1s 183ms/step - loss: 1.0810 -  
accuracy: 0.4083 - val_loss: 1.0788 - val_accuracy: 0.3833  
Epoch 6/60  
5/5 [=====] - 1s 177ms/step - loss: 1.0724 -  
accuracy: 0.3958 - val_loss: 1.0579 - val_accuracy: 0.4333  
Epoch 7/60
```

```
import numpy as np
CATEGORIES = ['airplane', 'truck','bird','cat']

def image(path):
    print('path',path)
    img = cv2.imread(path)

    new_arr = cv2.resize(img, (32, 32))
    new_arr = np.array(new_arr)
    new_arr = new_arr/255.0
    new_arr = new_arr.reshape(-1, 32, 32, 3)

    return new_arr
```

```
prediction = model.predict(image('/content/drive/MyDrive/datasets/_test_bird/0046.jpg'))  
#print(prediction.argmax())  
print(np.round(prediction,3))  
#airplane: 1 0 0 0, truck : 0 1 0 0, bird : 0 0 1 0,cat: 0 0 0 1
```

```
path /content/drive/MyDrive/datasets/_test_bird/0046.jpg  
1/1 [=====] - 0s 87ms/step  
[[0.02  0.001 0.84  0.139]]
```

과제3 . airplane, truck ,cat을 분류하는 code를 작성하고 실행하시오.

airplane, truck, cat data 사용

*.ipynb *.py upload

4classification_2024.ipynb code 사용

미니 프로젝트







인하공전 컴퓨터 정보공학과

dataset_large image 를 이용하여 airplane, truck ,cat을 분류하는 code를 만드시오,

성능 기준 : 마지막 혹은 best val accurac가 0.8 이상
30개 test 데이터중 25개 이상 예측 과 정답이 같아야 함 (correct_cnt.

MODEL 구조, EPOCH, BATCH SIZE 등을 변경 하여 성능을 높여보시오.

*.ipynb, *,.py upload

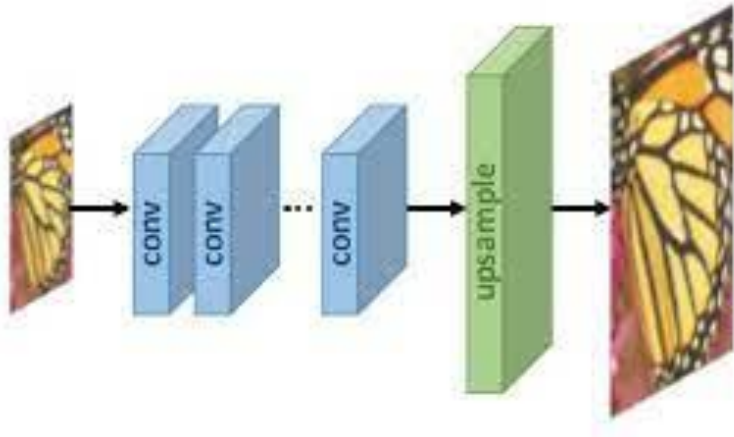
폴름 (D:) > 교과목 > 2024-1 > AI 프로그래밍 > 13주차 > dataset_large		
이름	수정한 날짜	유형
 airplane	2024-05-25 오후 ...	파일 폴더
 cat	2024-05-25 오후 ...	파일 폴더
 test_airplane	2024-05-25 오후 ...	파일 폴더
 test_cat	2024-05-25 오후 ...	파일 폴더
 test_truck	2024-05-25 오후 ...	파일 폴더
 truck	2024-05-25 오후 ...	파일 폴더

<https://transcranial.github.io/keras-js/#/mnist-cnn>

CNN 실습 – super resolution

인하공저 컴퓨터 정보과
인하공전 컴퓨터 정보공학과

<https://transcranial.github.io/keras-js/#/mnist-cnn>



CNN 실습 – style transfer

과제 1

style_transfer.ipynb

content1.jpg

... content8.jpg

style1.jpg

style2.jpg

style5.jpg

Content image, style image 중에 하나는 인터넷에서 찾아서 할 것

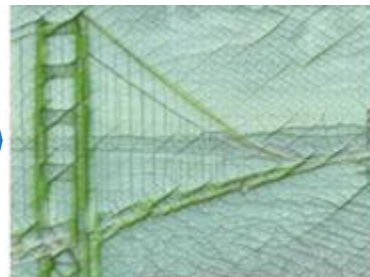
Style transfer content image, style image, 결과 이미지 upload



원본 영상



style영상



결과 영상



원본 영상



style영상



결과 영상

CNN 실습 – style transfer

style_transfer.ipynb

content1.jpg

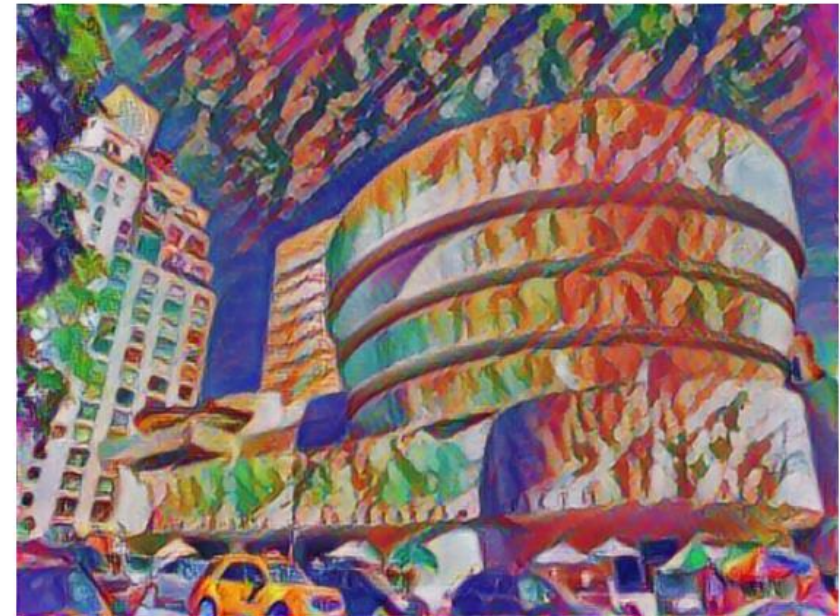
content2.jpg

content3.jpg

style1.jpg

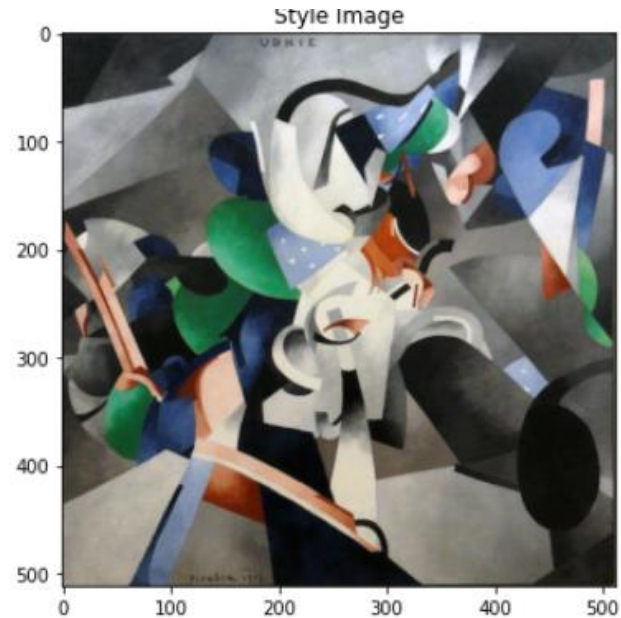
style2.jpg

style5.jpg



CNN 실습 – style transfer

style_transfer.ipynb



CNN 실습 – style transfer

style_transfer.ipynb

Content Image

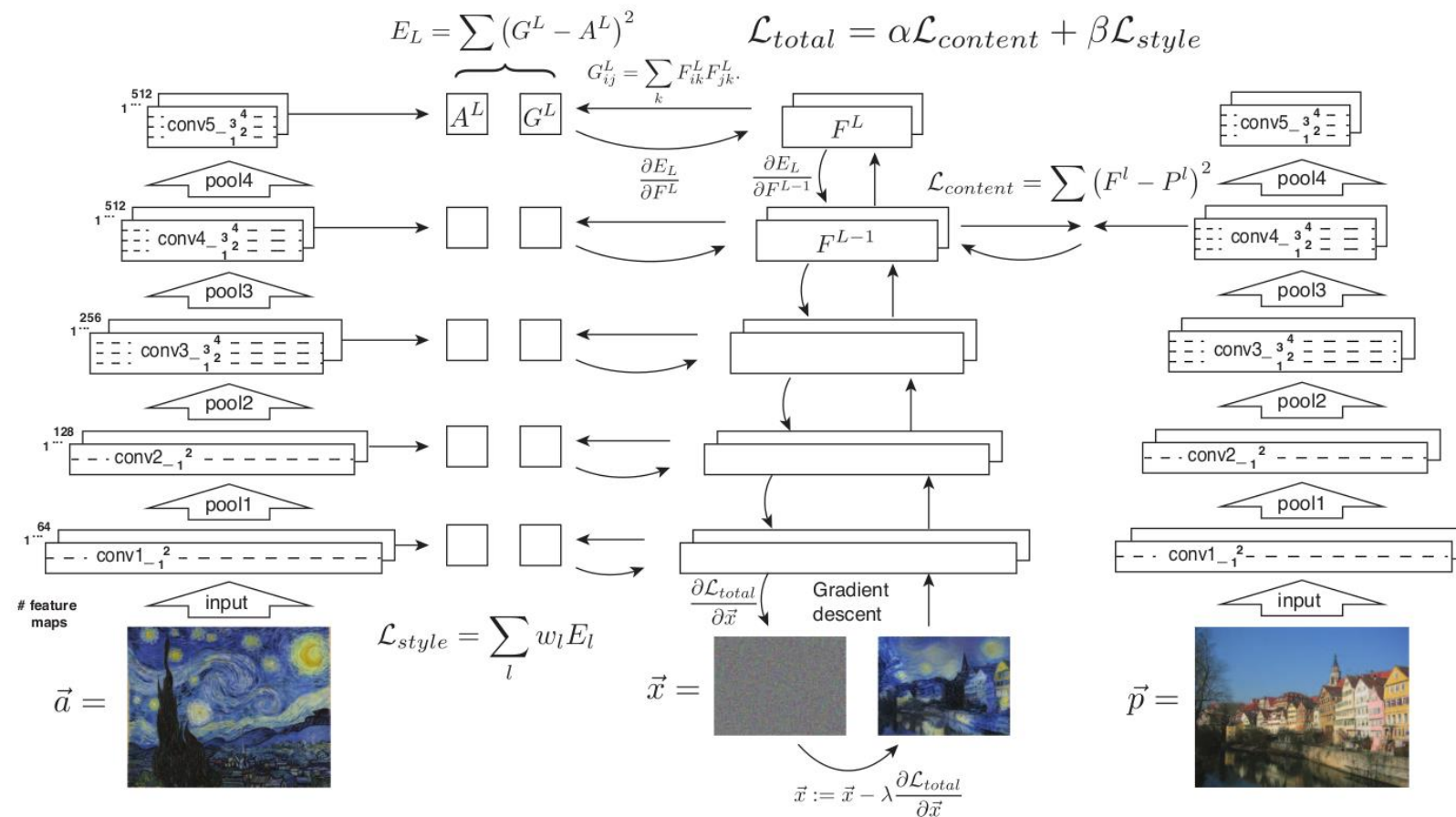


Style Image



CNN 실습 – style transfer

style_transfer.ipynb



와인 데이터- 시계열 데이터가 아님

인하공전 컴퓨터 정보공학과

fixed acidity												
	A	B	C	D	E	F	G	H	I	J	K	L
1	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur	total sulfur	density	pH	sulphates	alcohol	quality
2	7	0.27	0.36	20.7	0.045	45	170	1.001	3	0.45	8.8	6
3	6.3	0.3	0.34	1.6	0.049	14	132	0.994	3.3	0.49	9.5	6
4	8.1	0.28	0.4	6.9	0.05	30	97	0.9951	3.26	0.44	10.1	6
5	7.2	0.23	0.32	8.5	0.058	47	186	0.9956	3.19	0.4	9.9	6
6	7.2	0.23	0.32	8.5	0.058	47	186	0.9956	3.19	0.4	9.9	6
7	8.1	0.28	0.4	6.9	0.05	30	97	0.9951	3.26	0.44	10.1	6
8	6.2	0.32	0.16	7	0.045	30	136	0.9949	3.18	0.47	9.6	6
9	7	0.27	0.36	20.7	0.045	45	170	1.001	3	0.45	8.8	6
10	6.3	0.3	0.34	1.6	0.049	14	132	0.994	3.3	0.49	9.5	6
11	8.1	0.22	0.43	1.5	0.044	28	129	0.9938	3.22	0.45	11	6
12	8.1	0.27	0.41	1.45	0.033	11	63	0.9908	2.99	0.56	12	5
13	8.6	0.23	0.4	4.2	0.035	17	109	0.9947	3.14	0.53	9.7	5
14	7.9	0.18	0.37	1.2	0.04	16	75	0.992	3.18	0.63	10.8	5
15	6.6	0.16	0.4	1.5	0.044	48	143	0.9912	3.54	0.52	12.4	7
16	8.3	0.42	0.62	19.25	0.04	41	172	1.0002	2.98	0.67	9.7	5
17	6.6	0.17	0.38	1.5	0.032	28	112	0.9914	3.25	0.55	11.4	7
18	6.3	0.48	0.04	1.1	0.046	30	99	0.9928	3.24	0.36	9.6	6
19	6.2	0.66	0.48	1.2	0.029	29	75	0.9892	3.33	0.39	12.8	8
20	7.4	0.34	0.42	1.1	0.033	17	171	0.9917	3.12	0.53	11.3	6
21	6.5	0.31	0.14	7.5	0.044	34	133	0.9955	3.22	0.5	9.5	5
22	6.2	0.66	0.48	1.2	0.029	29	75	0.9892	3.33	0.39	12.8	8
23	6.4	0.31	0.38	2.9	0.038	19	102	0.9912	3.17	0.35	11	7
24	6.8	0.26	0.42	1.7	0.049	41	122	0.993	3.47	0.48	10.5	8
25	7.6	0.67	0.11	1.5	0.074	25	168	0.9927	2.95	0.51	9.2	5

- 1 - 고정 산도
- 2 - 휘발성 산도
- 3 - 구연산
- 4 - 잔류 설탕
- 5 - 염화물
- 6 - 유리 이산화황
- 7 - 총 이산화황
- 8 - 밀도
- 9 - pH
- 10 - 황산염
- 11 - 알코올
- 12 - 품질(0에서 10 사이의 점수)

👉 winequality.csv

: 샘플 4,898개

: 입력 변수 fixed acidity ~ alcohol - 11개

: 출력 변수 quality - 1개

인하공전 컴퓨터 정보공학과

- 순차데이터란, 순서가 있는 데이터이다. 시간적인 순서도 가능하고 공간적인 순서도 가능하다.

- 매일의 빵 수요량

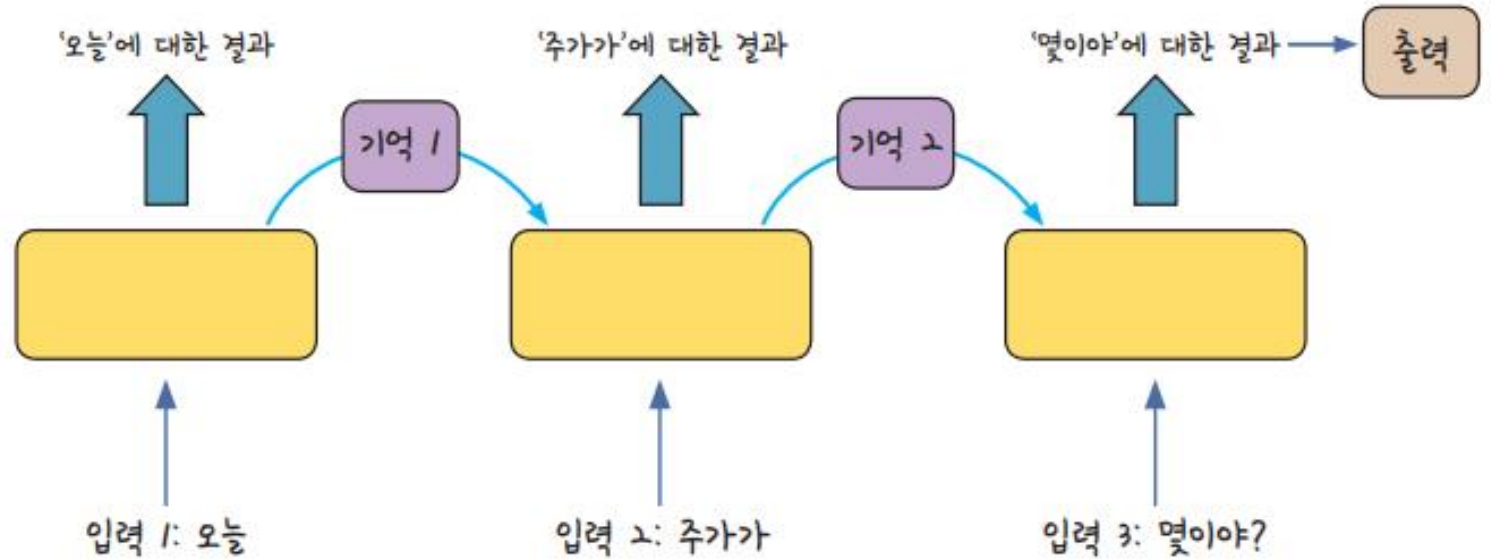
- 순차 데이터를 처리하여 정확한 예측을 하려면, 과거의 데이터를 어느 정도 기억하고 있어야 한다.

- 우리가 앞에서 학습한 표준 신경망을 사용하여도 어느 정도 예측이 가능하지만, 표준 신경망은 멀리 떨어진 과거의 데이터를 잘 기억하지는 못한다.
- 과거의 데이터로 다음을 예측
- 표준 신경망의 구조를 순차 데이터를 잘 처리하게끔 변경하는 것이 필요하다. 이것이 바로 순환 신경망(**RNN**: recurrent neural network)이다.

- 우리가 앞에서 학습한 표준 신경망을 사용하여도 어느 정도 예측이 가능하지만, 표준 신경망은 멀리 떨어진 과거의 데이터를 잘 기억하지는 못한다.
- 과거의 데이터로 다음을 예측
- 표준 신경망의 구조를 순차 데이터를 잘 처리하게끔 변경하는 것이 필요하다. 이것이 바로 순환 신경망(**RNN**: recurrent neural network)이다.

순환 신경망 (Recurrent Neural Network)

인하공전 컴퓨터 정보공학과



- 전체 데이터를 다음과 같이, 크기가 3인 샘플과 정답으로 분리하게 된다

샘플 번호	x(입력)	y(정답)
1	[16, 8, 32]	[9]
2	[8, 32, 9]	[23]
3	[32, 9, 23]	[11]
...
1092	[22, 15, 7]	[18]

1	2	3	4	5	6	7	...	1092	1093	1094	1095
16	8	32	9	23	11	19	...	22	15	7	18

샘플 #1 

샘플 #2 

샘플 #3 

참고 : 딥러닝 익스프레스

주가 예측

인하공전 컴퓨터 정보공학과

2016

2022.5.26

1374

200

30

```
print(samsung.shape)
print(train_data.shape)
print(test_data.shape)
```

(1574, 6)
(1374, 1)
(200, 1)

30일

최근 30일 데이터

내일 가격?

시계열 데이터 (Time-Series Data)

인하공전 컴퓨터 정보공학과

분야	형태	최종 결과물
음성 인식		What are recurrent neural network?
감정 분석	"It is my favorite time travel sci-fi"	★★★★★
자동 번역	"순환 신경망이란 무엇인가?"	"What is Recurrent Neural Network?"

최근 3학기 성적에서 다음 학기 성적 예측

최근 20일 빵 수요량에서 내일 빵 수요량 예측

참고 : 딥러닝 익스프레스

- 우리가 앞에서 학습한 표준 신경망을 사용하여도 어느 정도 예측이 가능하지만, 표준 신경망은 멀리 떨어진 과거의 데이터를 잘 기억하지는 못한다.
- 과거의 데이터로 다음을 예측
- 표준 신경망의 구조를 순차 데이터를 잘 처리하게끔 변경하는 것이 필요하다. 이것이 바로 순환 신경망(**RNN**: recurrent neural network)이다.

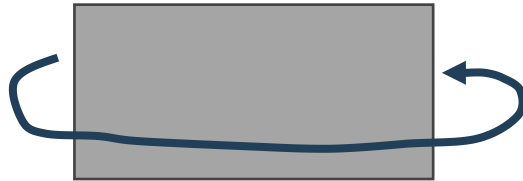
출력



입력

일반 신경망
(feedforward neural
network)

출력

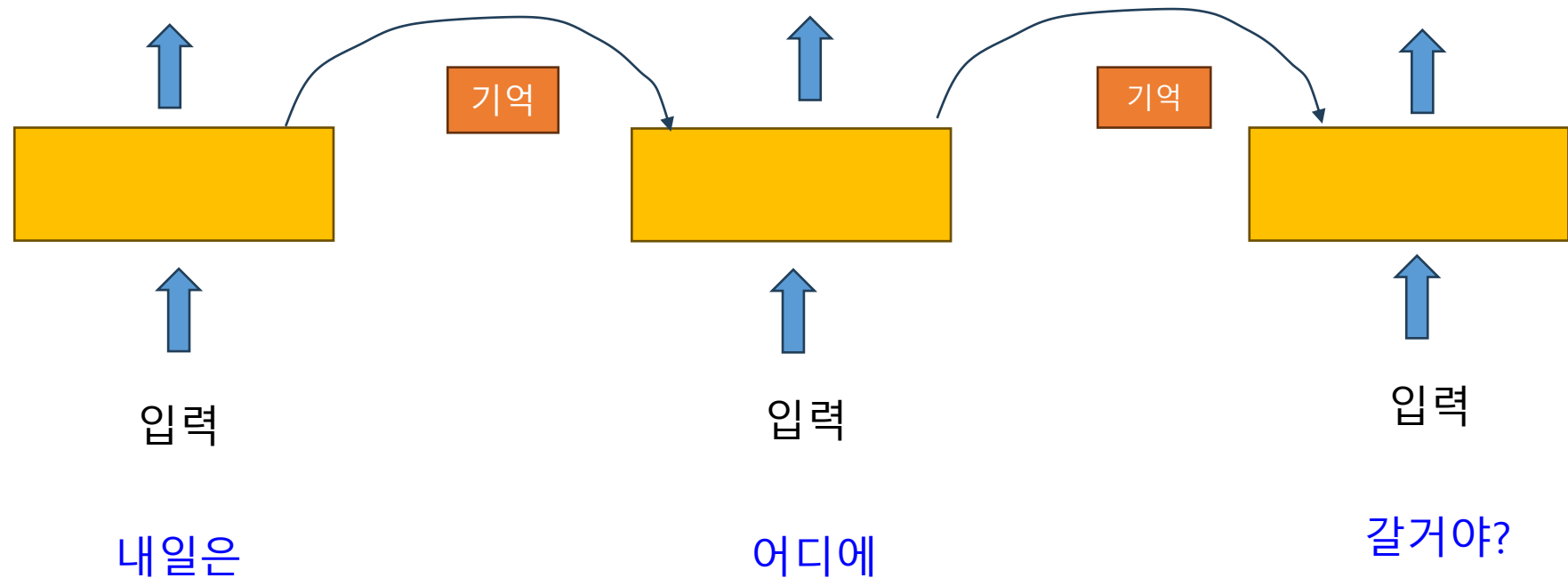


입력

순환 신경망
(recurrent neural
network)

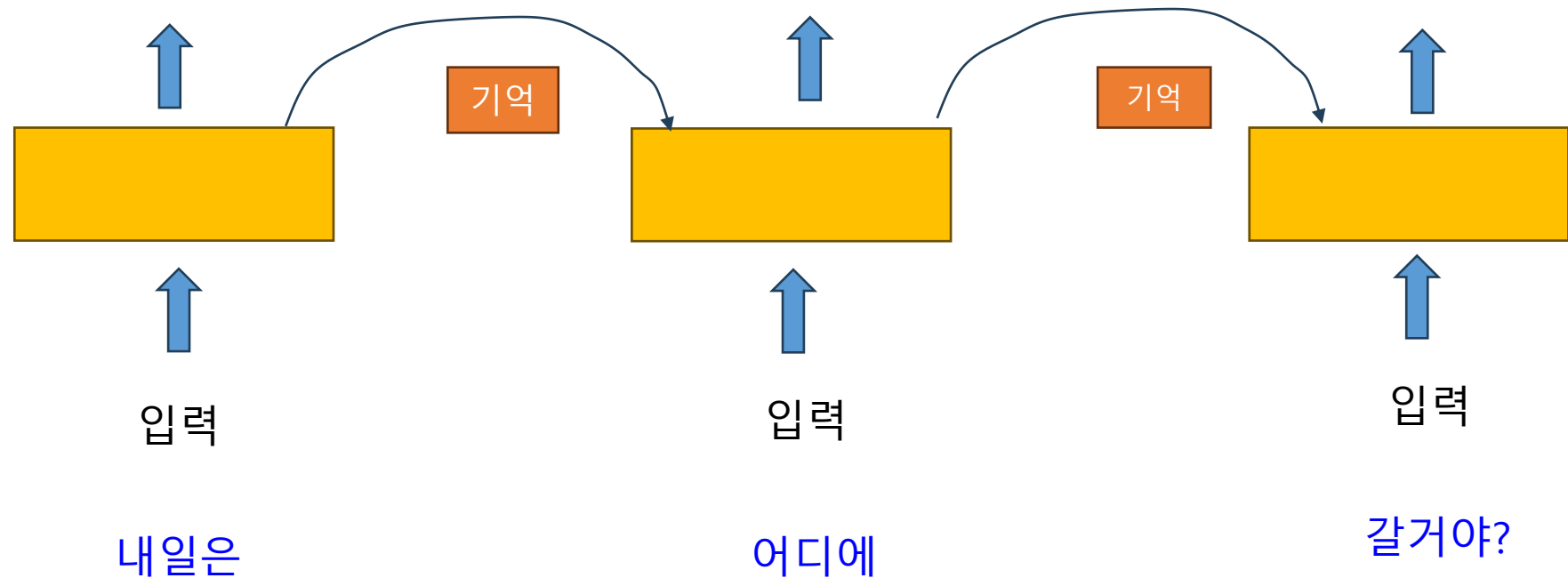
순환 신경망(Recurrent Neural Network)

인하공전 컴퓨터 정보공학과



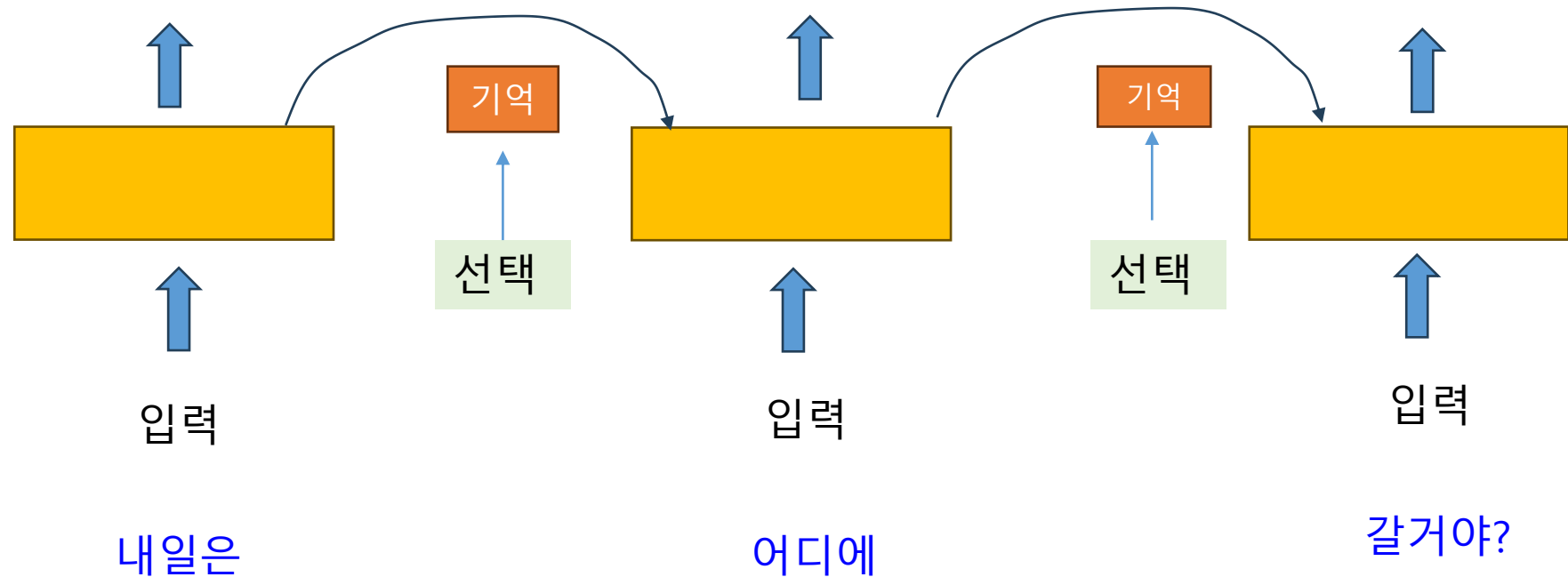
순환 신경망(Recurrent Neural Network)

인하공전 컴퓨터 정보공학과



LSTM (long short term memory)

인하공전 컴퓨터 정보공학과



RNN이 발전된 VERSION

다음 층으로 기억된 값을 넘길지 여부를 관리하는 단계가 추가됨

순환 신경망 (Recurrent Neural Network)

인하공전 컴퓨터 정보공학과

```
# 라이브러리 포함
import FinanceDataReader as fdr
import numpy as np
import matplotlib.pyplot as plt

# 삼성전자 코드='005930', 2020년 데이터부터 다운로드
samsung = fdr.DataReader('005930', '2020')

# 시작가만 취한다.
seq_data = (samsung[['Open']]).to_numpy()
# 선형 그래프로 그린다.
plt.plot(seq_data, color='blue')
plt.title("Samsung Electronics Stock Price")
plt.xlabel("days")
plt.xlabel("")
plt.show()
```

순환 신경망 (Recurrent Neural Network)

인하공전 컴퓨터 정보공학과



참고 : 딥러닝 익스프레스

순환 신경망 (Recurrent Neural Network)

인하공전 컴퓨터 정보공학과

```
# 라이브러리 포함
import FinanceDataReader as fdr
import numpy as np
import matplotlib.pyplot as plt

# 삼성전자 코드='005930', 2020년 데이터부터 다운로드
samsung = fdr.DataReader('005930', '2020')

# 시작가만 취한다.
seq_data = (samsung[['Open']]).to_numpy()
# 선형 그래프로 그린다.
plt.plot(seq_data, color='blue')
plt.title("Samsung Electronics Stock Price")
plt.xlabel("days")
plt.xlabel("")
plt.show()
```

참고 : 딥러닝 익스프레스

■ 데이터를 잘라서 샘플을 작성해보자

```
seq_data = (samsung[['Open']]).to_numpy()

def make_sample(data, window):
    train = []
    target = []
    for i in range(len(data)-window):
        train.append(data[i:i+window])
        target.append(data[i+window])
    return np.array(train), np.array(target)

X, y = make_sample(seq_data, 7)
print(X.shape, y.shape)
print(X[0], y[0])
```

공백 리스트 생성

데이터의 길이만큼 반복
i부터 (i+window-1) 까지를 저장
(i+window) 번째 요소는 정답
훈련 샘플과 정답 레이블을 반환

윈도우 크기=7
넘파이 배열의 형상 출력
첫 번째 샘플 출력

순환 신경망 (Recurrent Neural Network)

인하공전 컴퓨터 정보공학과

```
(284, 7, 1) (284, 1)
[[55500]
[56000]
[54900]
[55700]
[56200]
[58400]
[58800]] [59600]
```

참고 : 딥러닝 익스프레스

순환 신경망 (Recurrent Neural Network)

인하공전 컴퓨터 정보공학과

Open	High	Low	Close	Volume	Change		
Date							
2016-01-04	25200	25200	24100	24100	306939	-0.043651	
2016-01-05	24040	24360	23720	24160	216002	0.002490	
2016-01-06	24160	24160	23360	23500	366752	-0.027318	
2016-01-07	23320	23660	23020	23260	282388	-0.010213	
2016-01-08	23260	23720	23260	23420	257763	0.006879	
...	
2022-05-23	68800	68800	67600	67900	13684088	-0.001471	
2022-05-24	67500	67700	66500	66500	15482576	-0.020619	
2022-05-25	66700	67100	65900	66400	15150490	-0.001504	
2022-05-26	66300	67200	65500	65900	15970890	-0.007530	
2022-05-27	66700	66900	66200	66500	11346700	0.009105	

```
import FinanceDataReader as fdr
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd

samsung = fdr.DataReader('005930', '2016')
print(samsung)

openValues = samsung[['Open']]

from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler(feature_range = (0, 1))
scaled = scaler.fit_transform(openValues)

TEST_SIZE = 200
train_data = scaled[:-TEST_SIZE]
test_data = scaled[-TEST_SIZE:]
```

삼성전자 주가 2016부터 데이터

테스트 데이터를 후반부의 200개로

주가 예측

인하공전 컴퓨터 정보공학과

2016

2022.5.26

1374

200

30

```
print(samsung.shape)
print(train_data.shape)
print(test_data.shape)
```

(1574, 6)
(1374, 1)
(200, 1)

30일

최근 30일 데이터

내일 가격?

```
def make_sample(data, window):  
    train = []  
    target = []  
    for i in range(len(data)-window):  
        train.append(data[i:i+window])  
        target.append(data[i+window])  
    return np.array(train), np.array(target)  
  
X_train, y_train = make_sample(train_data, 30)
```

```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
from tensorflow.keras.layers import LSTM

model = Sequential()
model.add(LSTM(16,
               input_shape=(X_train.shape[1], 1),
               activation='tanh',
               return_sequences=False)
)
model.add(Dense(1))

model.compile(optimizer = 'adam', loss = 'mean_squared_error')
model.fit(X_train, y_train, epochs = 100, batch_size = 16)
```

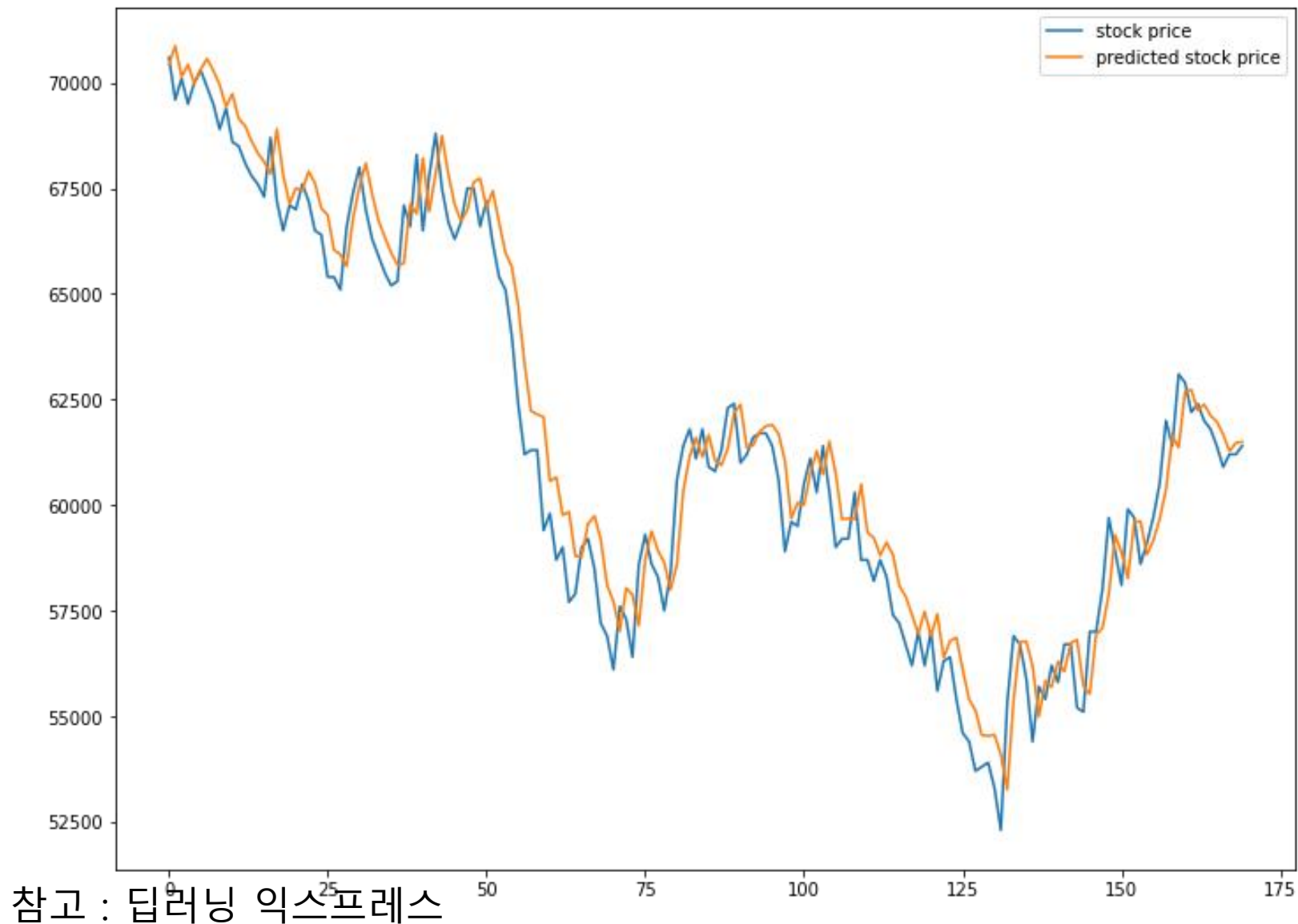
메모리 셀의 개수는 기억용량 정도와 출력 형태를 결정짓습니다. Dense 레이어에서의 출력 뉴런 수와 비슷한 의미/


```
X_test, y_test = make_sample(test_data, 30)
pred = model.predict(X_test)

import matplotlib.pyplot as plt
plt.figure(figsize=(12, 9))
plt.plot(y_test*data_max_, label='stock price')
plt.plot(pred*data_max_, label='predicted stock price')
plt.legend()
plt.show()
pred_n=pred*scaler.data_max_
y_test_n=y_test*scaler.data_max_
print('# RMSE ',np.mean(losses.mean_squared_error(pred_n,y_test_n)**0.5))
```

케라스를 이용한 주가 예측

인하공전 컴퓨터 정보공학과



■ 성능 비교

RNN1

```
model = Sequential()  
model.add(SimpleRNN(16, activation='tanh', input_shape=(X_train.shape[1], 1)))  
model.add(Dense(1))
```

RMSE : 757.733

RNN2

```
model = Sequential()  
model.add(SimpleRNN(16, activation='tanh', return_sequences=True, input_shape=(X_train.shape[1], 1)))  
model.add(SimpleRNN(20))  
model.add(Dense(1))
```

RMSE : 843.077

LSTM

```
model = Sequential()  
model.add(LSTM(16,  
              input_shape=(X_train.shape[1], 1),  
              activation='tanh',  
              return_sequences=False)  
          )  
model.add(Dense(1))
```

RMSE : RMSE : 855.9182

케라스를 이용한 주가 예측-과제

인하공전 컴퓨터 정보공학과

■ 성능 비교

RNN1

```
model = Sequential()  
model.add(SimpleRNN(16, activation='tanh', input_shape=(X_train.shape[1], 1)))  
model.add(Dense(1))
```

RMSE :

RNN2

```
model = Sequential()  
model.add(SimpleRNN(16, activation='tanh', return_sequences=True, input_shape=(X_train.shape[1], 1)))  
model.add(SimpleRNN(20))  
model.add(Dense(1))
```

LSTM

```
model = Sequential()  
model.add(LSTM(16,  
              input_shape=(X_train.shape[1], 1),  
              activation='tanh',  
              return_sequences=False)  
          )  
model.add(Dense(1))
```

RMSE :

RMSE :

케라스를 이용한 주가 예측-과제

인하공전 컴퓨터 정보공학과

■ 성능 비교

big14_stock_assign_nofill.ipynb

과제 2)

특정 주식 선택 . 종목 코드 검색
. 주가: 2018년부터 데이터 이용

EPOCH 는 50번

데스트 데이터 : 후반부 100개 사용

결과 과제 제출시 입력. Ppt 제출 할 필요 없음.

	RNN1	RNN2	LSTM
RMSE			

수고하셨습니다

jhmin@inhatec.ac.kr