

AI 프로그래밍

- 2024



11주차

- 저장된 model 사용
- Fully connect layer parameter 수
- 정규화
- wine quality 예측
 - 회귀 실습
 - 분류 실습
- Open CV library
 - 영상 처리 기초

- 총합이 1인 형태로 변환

그림 12-3에서와 같이 총합이 1인 형태로 바뀌서 계산해 주는 함수

3.0	1.5	0.3
-----	-----	-----



Soft max

0.7	0.2	0.1
-----	-----	-----

출력

```
Model.add(Dense(10,input_dim=4, activate='relu'))  
Model.add(Dense(3,activation='softmax'))
```

1.0	0.0	0.0
-----	-----	-----

(one hot label) 정답

케라스 신경망 실습 - 패션 아이템 분류

인하공전 컴퓨터 정보공학과

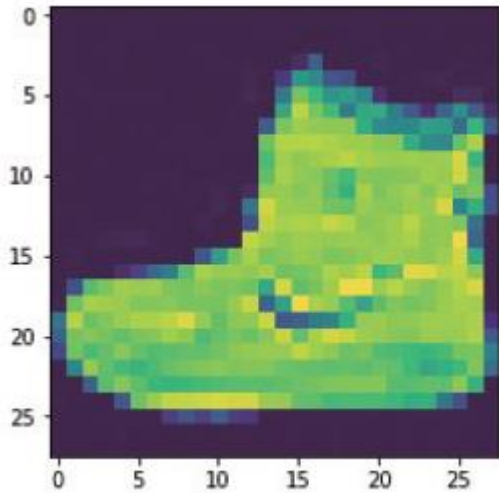
- 이미지는 28x28 크기이고
- 픽셀 값은 0과 255 사이
- 레이블(label)은 0에서 9까지의

레이블	범주
0	T-shirt/top
1	trouser
2	pullover
3	dress
4	coat
5	sandal
6	shirt
7	sneaker
8	bag
9	Ankle boot

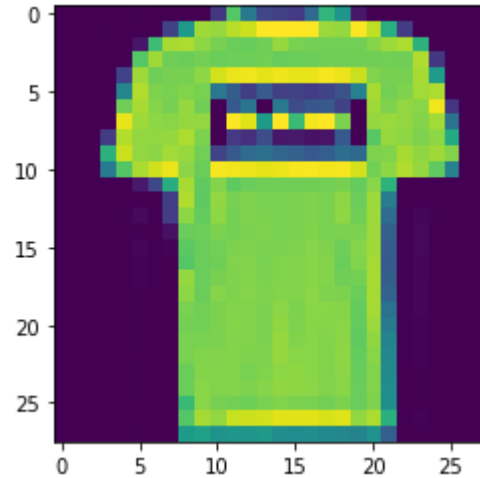
케라스 신경망 실습 - 패션 아이템 분류

인하공전 컴퓨터 정보공학과

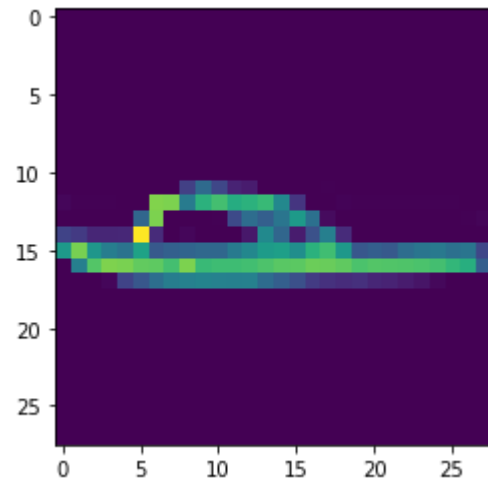
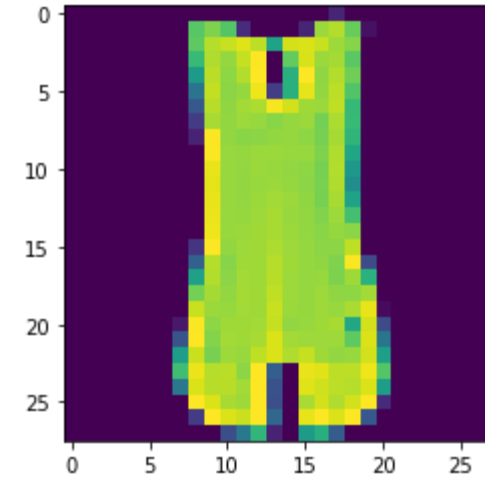
- `plt.imshow(train_images[0])`



두번째



네번째



28x28 gray image

케라스 신경망 실습 - 패션 아이템 분류

인하공전 컴퓨터 정보공학과

```
import tensorflow as tf
from tensorflow import keras
import numpy as np
import matplotlib.pyplot as plt
from tensorflow.keras import datasets, layers, models

fashion_mnist = keras.datasets.fashion_mnist
(train_images, train_labels), (test_images, test_labels) = fashion_mnist.load_data()

plt.imshow(train_images[0])

train_images = train_images / 255.0
test_images = test_images / 255.0
```

```
model = models.Sequential()
model.add(layers.Flatten(input_shape=(28, 28)))
model.add(layers.Dense(128, activation='relu'))
model.add(layers.Dense(10, activation='softmax'))

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

model.fit(train_images, train_labels, epochs=5)

test_loss, test_acc = model.evaluate(test_images, test_labels)
print('정확도:', test_acc)
```

케라스 신경망 실습 - 패션 아이템 분류

인하공전 컴퓨터 정보공학과

```
model = models.Sequential()
model.add(layers.Flatten(input_shape=(28, 28)))
model.add(layers.Dense(128, activation='relu'))
model.add(layers.Dense(10, activation='softmax'))

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

model.fit(train_images, train_labels, epochs=5)

test_loss, test_acc = model.evaluate(test_images, test_labels)
print('정확도:', test_acc)
```

```
10000/10000 [=====] - 0s 32us/sample - loss: 0.3560 -
acc: 0.8701
정확도: 0.8701
```



```
Import pandas as pd
```

```
df=read_csv('a.csv')
```

- df.drop : 열 삭제
- df.dropna : 결측치 포함 행 삭제
- df.head : 데이터 앞부분 확인
- df.info : 데이터 요약
- df.loc , df.iloc : 원소 선택

케라스 신경망 실습 - 타이타닉 생존자 예측

인하공전 컴퓨터 정보공학과

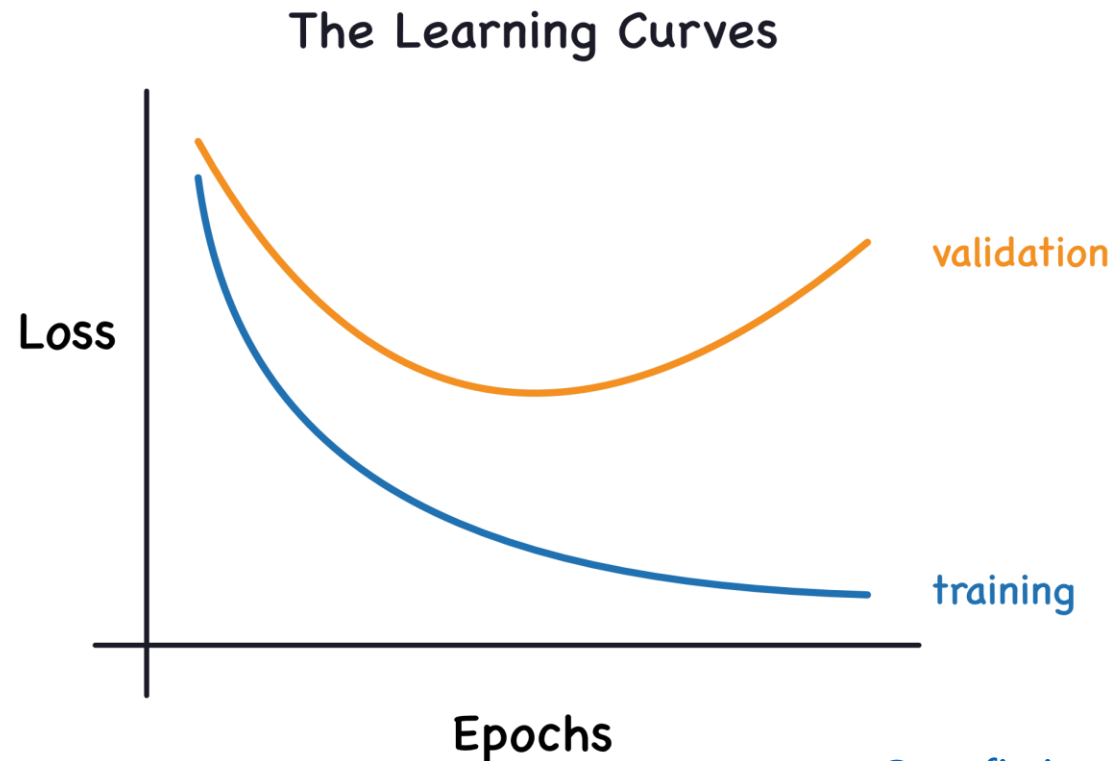
```
idx=test.shape[0]
pred=model.predict(test)
test_np=test.to_numpy()
for i in range(idx):
    print(test_np[i,:],pred[i])
```

Evaluate(testx,testy)

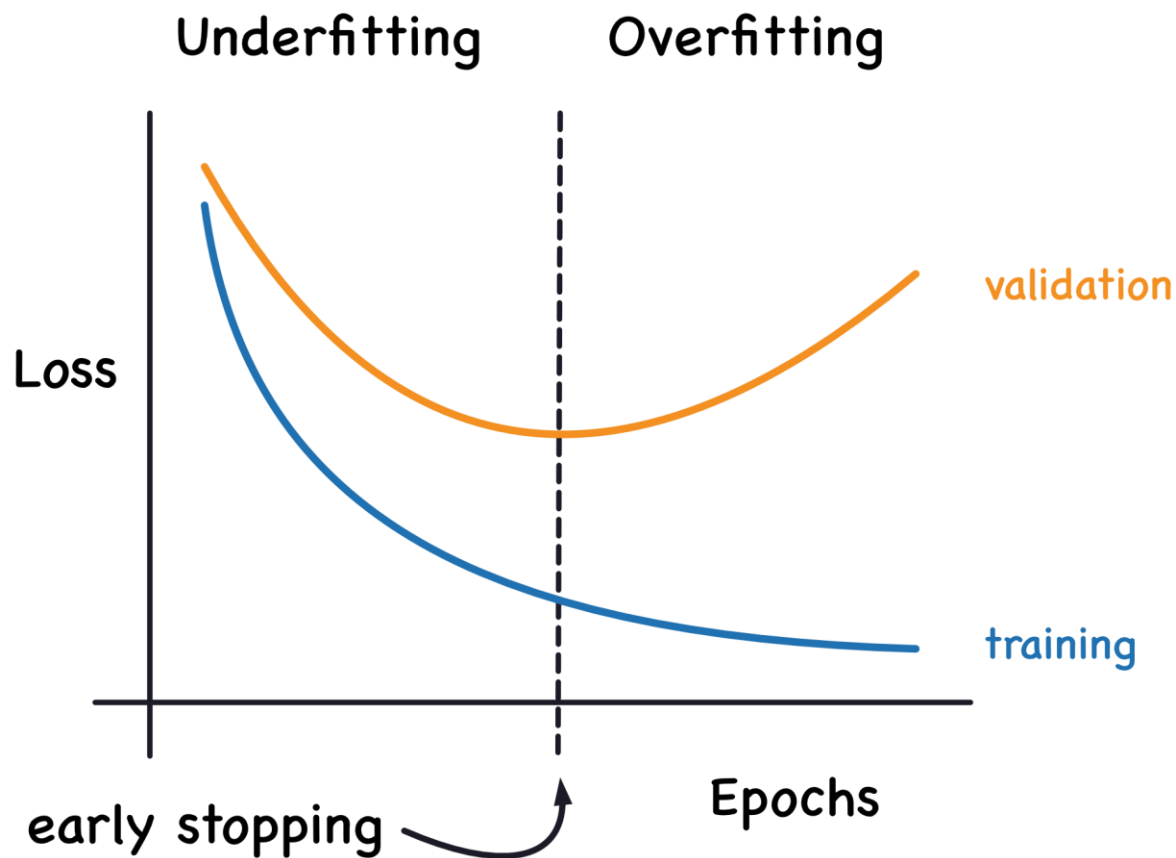
객실 등급, 성별, 생존 확률

```
[3. 1.] [0.11963955]
[3. 0.] [0.42622244]
[2. 1.] [0.20062831]
[3. 1.] [0.11963955]
[3. 0.] [0.42622244]
[3. 1.] [0.11963955]
[3. 0.] [0.42622244]
[2. 1.] [0.20062831]
[3. 0.] [0.42622244]
[3. 1.] [0.11963955]
[3. 1.] [0.11963955]
[1. 1.] [0.35509673]
[1. 0.] [0.9667859]
[2. 1.] [0.20062831]
[1. 0.] [0.9667859]
[2. 0.] [0.84050065]
[2. 1.] [0.20062831]
[3. 1.] [0.11963955]
[3. 0.] [0.42622244]
[3. 0.] [0.42622244]
[1. 1.] [0.35509673]
[3. 1.] [0.11963955]
[1. 0.] [0.9667859]
[1. 1.] [0.35509673]
[1. 0.] [0.9667859]
[3. 1.] [0.11963955]
```

- 학습(training) 데이터 loss는 감소 될 때 검증(validation) 데이터의 loss가 줄어들지 않으면 과적합 (overfitting)



- 검증 손실(validation loss)이 더 이상 감소하지않을때 학습을 중단=< (Early Stopping)라고 한다.



- 학습 끝난 후 model 저장
 - `model.save('./mymodel.hdf5')`
- 학습 끝난 후 model 저장
 - `From tensorflow.keras.models import load_model`
 - `model=load_model('./mymodel.hdf5')`

케라스 신경망 실습 - 와인 종류 예측 하기

인하공전 컴퓨터 정보공학과

```
from tensorflow import keras
from keras.models import Sequential
from keras.layers import Dense
from sklearn.model_selection import train_test_split
from keras.callbacks import ModelCheckpoint,
EarlyStopping
import os
import pandas as pd

# 데이터를 입력합니다.
df = pd.read_csv('/content/wine.csv', header=None)

df
```

케라스 신경망 실습 - 와인 종류 예측 하기

인하공전 컴퓨터 정보공학과



	0	1	2	3	4	5	6	7	8	9	10	11	12
0	7.4	0.70	0.00	1.9	0.076	11.0	34.0	0.99780	3.51	0.56	9.4	5	1
1	7.8	0.88	0.00	2.6	0.098	25.0	67.0	0.99680	3.20	0.68	9.8	5	1
2	7.8	0.76	0.04	2.3	0.092	15.0	54.0	0.99700	3.26	0.65	9.8	5	1
3	11.2	0.28	0.56	1.9	0.075	17.0	60.0	0.99800	3.16	0.58	9.8	6	1
4	7.4	0.70	0.00	1.9	0.076	11.0	34.0	0.99780	3.51	0.56	9.4	5	1
...
6492	6.2	0.21	0.29	1.6	0.039	24.0	92.0	0.99114	3.27	0.50	11.2	6	0
6493	6.6	0.32	0.36	8.0	0.047	57.0	168.0	0.99490	3.15	0.46	9.6	5	0
6494	6.5	0.24	0.19	1.2	0.041	30.0	111.0	0.99254	2.99	0.46	9.4	6	0
6495	5.5	0.29	0.30	1.1	0.022	20.0	110.0	0.98869	3.34	0.38	12.8	7	0
6496	6.0	0.21	0.38	0.8	0.020	22.0	98.0	0.98941	3.26	0.32	11.8	6	0

6497 rows x 13 columns

- 와인 데이터 셋

- 샘플 6497개
- 속성(특징) 12개
- 13번째 열 class (1:레드 0: 화이트)
- 샘플이 전체 6,497개 있음

케라스 신경망 실습 - 와인 종류 예측 하기

인하공전 컴퓨터 정보공학과

```
# 와인의 속성을 X로 와인의 분류를 y로 저장합니다.
```

```
X = df.iloc[:,0:12]
```

```
y = df.iloc[:,12]
```

```
X = X.astype(float)
```

```
y = y.astype(float)
```

```
#학습셋과 테스트셋으로 나눕니다.
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, shuffle=True)
```

케라스 신경망 실습 - 와인 종류 예측 하기

인하공전 컴퓨터 정보공학과

모델 구조를 설정합니다.

```
model = Sequential()  
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'))  
model.summary()
```

#모델을 컴파일합니다.

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

케라스 신경망 실습 - 와인 종류 예측 하기

인하공전 컴퓨터 정보공학과

학습의 자동 중단 및 최적화 모델 저장

학습이 언제 자동 중단 될지를 설정합니다.

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

#최적화 모델이 저장될 폴더와 모델의 이름을 정합니다.

modelpath="./data/model/bestmodel.hdf5"

최적화 모델을 업데이트하고 저장합니다.

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

#모델을 실행합니다.

history=model.fit(X_train, y_train, epochs=2000, batch_size=500, **validation_split=0.25,** verbose=1,
callbacks=[early_stopping_callback,checkpointer])

케라스 신경망 실습 - 와인 종류 예측 하기

인하공전 컴퓨터 정보공학과
인하공전 컴퓨터 정보공학과

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

- 검증 셋 (validation set) : 학습 과정에서 최적의 parameter를 찾기 위해서 사용
과적합 방지에 도움이 됨.
fit 함수 안에 validation_split 을 이용하여 만들어 짐

Early stopping 학습 중단 시점 선택

```
### 학습의 자동 중단 및 최적화 모델 저장  
# 학습이 언제 자동 중단 될지를 설정합니다.  
early_stopping_callback = EarlyStopping(monitor='val_loss', patience=30)
```

monitor='val_loss', patience=20이라고 지정하면 검증셋의 오차가 30번 이상 낮아지지 않을 경우 학습을 종료하라는 의미

모델 저장 시점 선택

#최적화 모델이 저장될 폴더와 모델의 이름을 정합니다.

```
modelpath="./data/model/bestmodel.hdf5"
```

최적화 모델을 업데이트하고 저장합니다.

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

=> 검증 셋 오차(validation loss) 기준으로 제일 성능이 좋을 때 모델 저장

#모델을 실행합니다.

```
history=model.fit(X_train, y_train, epochs=2000, batch_size=500, validation_split=0.25, verbose=1,  
callbacks=[early_stopping_callback,checkpointer])
```

과잉 적합 방지 -영화 리뷰 분류 drop out

인하공전 컴퓨터 정보공학과

- 영화 리뷰를 분류하는 문제를 생각해보자. IMDB 사이트는 영화에 대한 리뷰가 올라가 있는 사이트이다.
- 영화 리뷰를 입력하면 리뷰가 긍정적인지 부정적인지를 파악하는 문제

과잉 적합 방지 -영화 리뷰 분류 drop out

인하공전 컴퓨터 정보공학과

```
import numpy as numpy
import tensorflow as tf
import matplotlib.pyplot as plt

# 데이터 다운로드
(train_data, train_labels), (test_data, test_labels) = \
    tf.keras.datasets.imdb.load_data( num_words=1000)

# 원-핫 인코딩으로 변환하는 함수
def one_hot_sequences(sequences, dimension=1000):
    results = numpy.zeros((len(sequences), dimension))
    for i, word_index in enumerate(sequences):
        results[i, word_index] = 1.
    return results

train_data = one_hot_sequences(train_data)
test_data = one_hot_sequences(test_data)
```

과잉 적합 방지 -영화 리뷰 분류 drop out

인하공전 컴퓨터 정보공학과

```
# 신경망 모델 구축
```

```
model = tf.keras.Sequential()
```

```
model.add(tf.keras.layers.Dense(16, activation='relu', input_shape=(1000,)))
```

```
model.add(tf.keras.layers.Dense(16, activation='relu'))
```

```
model.add(tf.keras.layers.Dense(1, activation='sigmoid'))
```

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

```
# 신경망 훈련, 검증 데이터 전달
```

```
history = model.fit(train_data,  
                    train_labels,  
                    epochs=20,  
                    batch_size=512,  
                    validation_data=(test_data, test_labels),  
                    verbose=2)
```

과잉 적합 방지 -영화 리뷰 분류 drop out

인하공전 컴퓨터 정보공학과

```
# 훈련 데이터의 손실값과 검증 데이터의 손실값을 그래프에 출력
history_dict = history.history
loss_values = history_dict['loss']           # 훈련 데이터 손실값
val_loss_values = history_dict['val_loss']    # 검증 데이터 손실값
acc = history_dict['accuracy']               # 정확도
epochs = range(1, len(acc) + 1)             # 에포크 수

plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('Loss Plot')
plt.ylabel('loss')
plt.xlabel('epochs')
plt.legend(['train error', 'val error'], loc='upper left')
plt.show()
```

- 드롭아웃은 학습 과정에서 node들을 랜덤하게 비 활성화 하는것이다.
- 특정 노드에 너무 의존 하지 않도록
- 과적합 방지

과제 4)

Imdb.ipynb code에서 아래처럼 model 생성 부분에 drop out 을 추가 한 후 학습 시킨 후 생성된. loss plot (train err, val err)를 upload하시오

```
# 신경망 모델 구축
model = tf.keras.Sequential()
model.add(tf.keras.layers.Dense(16, activation='relu', input_shape=(10000,)))
model.add(tf.keras.layers.Dropout(0.5))
model.add(tf.keras.layers.Dense(16, activation='relu'))
model.add(tf.keras.layers.Dropout(0.5))
model.add(tf.keras.layers.Dense(1, activation='sigmoid'))
```

- 주택 가격 예측

- 회귀 문제
- 80개의 속성 (환경, 차고 등등)을 이용하여 집값 예측 을 맞히거나 여러 개의 보기 중 하나를 예측하는 분류 문제

케라스 신경망 실습 - 주택 가격 예측

인하공전 컴퓨터 정보공학과
인하공전 컴퓨터 정보공학과

df.head()

	Id	MSSubClass	LotFrontage	LotArea	OverallQual	OverallCond	YearBuilt	YearRemodAdd	MasVnrArea	Bs
0	1	60	65.0	8450	7	5	2003	2003	196.0	
1	2	20	80.0	9600	6	8	1976	1976	0.0	
2	3	60	68.0	11250	7	5	2001	2002	162.0	
3	4	70	60.0	9550	7	5	1915	1970	0.0	
4	5	60	84.0	14260	8	5	2000	2000	350.0	

5 rows x 289 columns

- 데이터 파악하기

- 총 80개의 속성으로 이루어져 있고 마지막 열이 우리의 타깃인 집 값(SalePrice)
- 모두 1,460개의 샘플이 들어 있음

케라스 신경망 실습 - 주택 가격 예측

```
from tensorflow import keras
from keras.models import Sequential
from keras.layers import Dense
from keras.callbacks import EarlyStopping, ModelCheckpoint
from sklearn.model_selection import train_test_split
```

```
import matplotlib.pyplot as plt
import seaborn as sns
```

```
import pandas as pd
import numpy as np
```

```
#데이터를 불러 옵니다.
```

```
df = pd.read_csv("/content/house_data.csv")
```

```
#카테고리형 변수를 0과 1로 이루어진 변수로 바꾸어 줍니다
```

```
df = pd.get_dummies(df)
```

```
#결측치를 전체 칼럼의 평균으로 대체하여 채워줍니다.
```

```
df = df.fillna(df.mean())
```

케라스 신경망 실습 - 주택 가격 예측

```
#업데이트된 데이터프레임을 출력해 봅니다.  
#df  
#집 값을 제외한 나머지 열을 저장합니다.  
cols_train=['OverallQual','GrLivArea','GarageCars','GarageArea','TotalBsmtSF']  
X_train_pre = df[cols_train]  
  
#집 값을 저장합니다.  
y = df['SalePrice'].values  
X_train_pre=X_train_pre.astype(float)  
y=y.astype(float)  
#전체의 80%를 학습셋으로, 20%를 테스트셋으로 지정합니다.  
X_train, X_test, y_train, y_test = train_test_split(X_train_pre, y, test_size=0.2)
```

케라스 신경망 실습 - 주택 가격 예측

#모델의 구조를 설정합니다.

```
model = Sequential()  
model.add(Dense(10, input_dim=X_train.shape[1], activation='relu'))  
model.add(Dense(30, activation='relu')) # 30  
model.add(Dense(40, activation='relu'))  
model.add(Dense(1))  
model.summary()
```

#모델을 실행합니다.

```
model.compile(optimizer='adam', loss='mean_squared_error')
```

20회 이상 결과가 향상되지 않으면 자동으로 중단되게끔 합니다.

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

케라스 신경망 실습 - 주택 가격 예측

모델의 이름을 정합니다.

```
modelpath="/content/house.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])
```

예측 값과 실제 값, 실행 번호가 들어갈 빈 리스트를 만듭니다.

```
real_prices = []
```

```
pred_prices = []
```

```
X_num = []
```

케라스 신경망 실습 - 주택 가격 예측

인하공전 컴퓨터 정보공학과

```
print('테스트 loss', model.evaluate(X_test, y_test) / 10000000)
```

- 주택 가격 예측 모델

- 실행에서 달라진 점은 손실 함수
- 선형 회귀이므로 평균 제곱 오차(mean_squared_error)를 적음



```
model.compile(optimizer='adam', loss='mean_squared_error')
```

케라스 신경망 실습 - 주택 가격 예측

- 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

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)
```

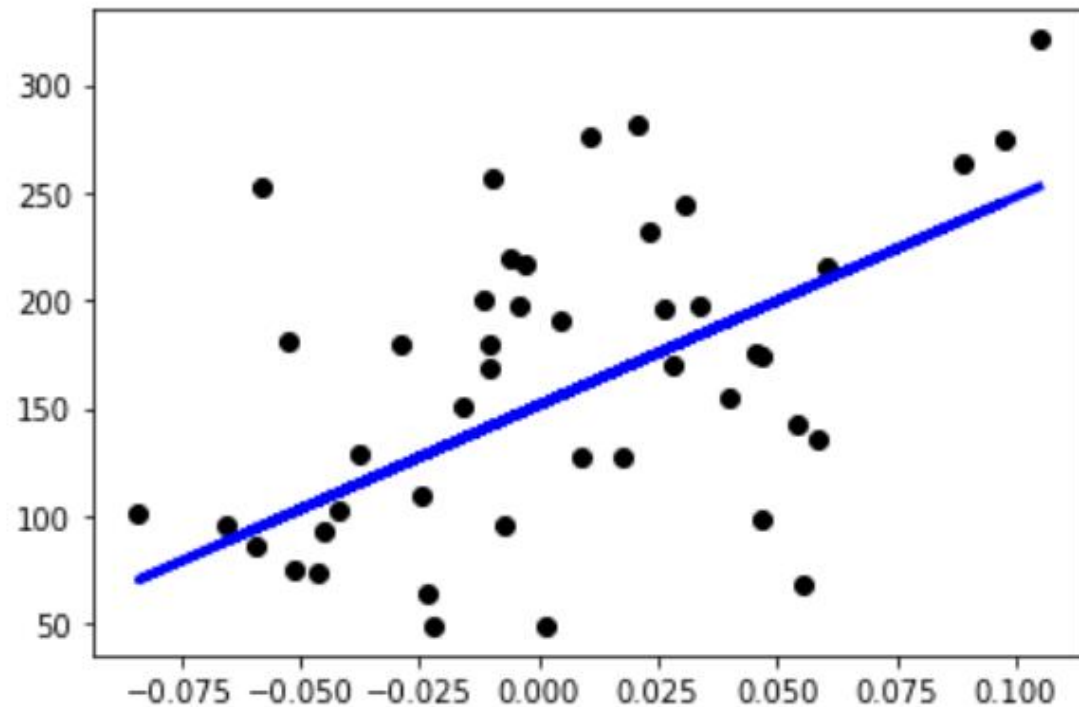
```
model=load_model ('/content/house2.hdf5')  
print('테스트 loss2', model.evaluate(X_test,y_test)/10000000)
```

house1.hdf5

house2.hdf5

test loss 값 과제 설명에 입력

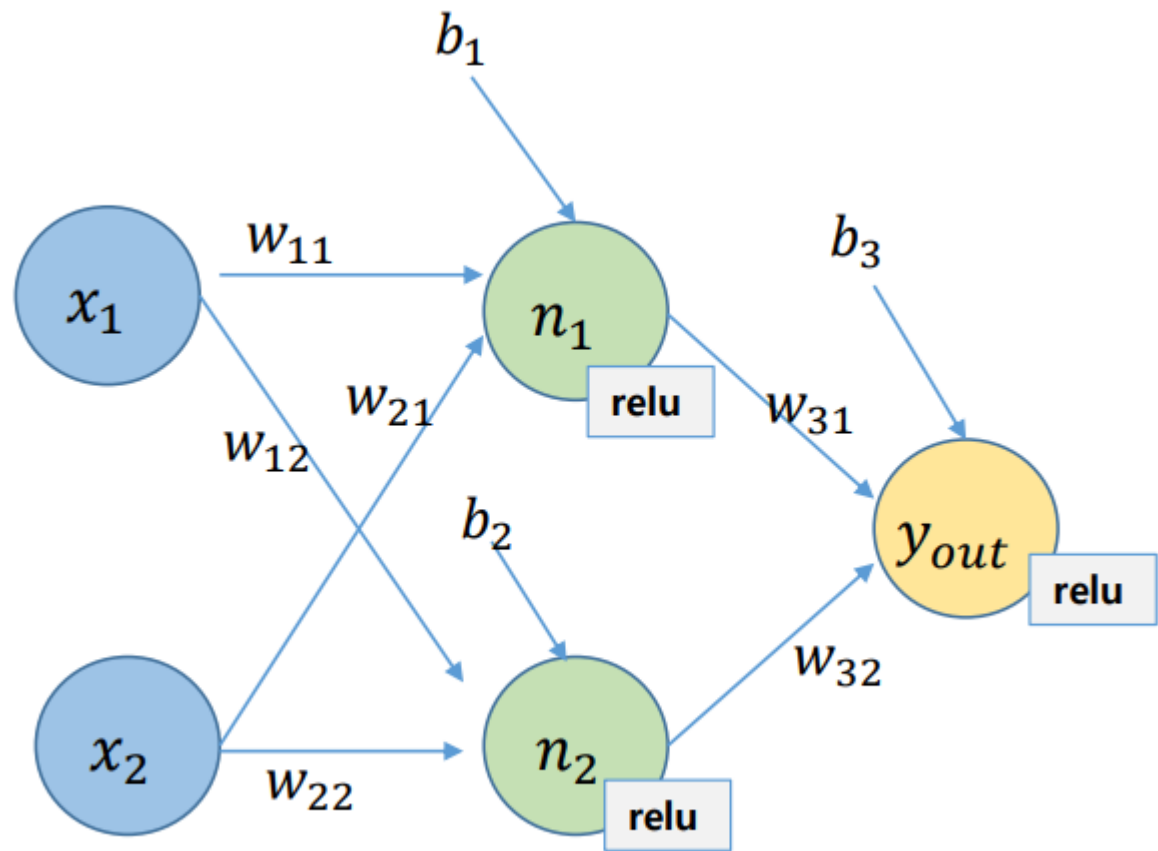
선형 회귀 parameter



```
iteration 0: loss 17.17 w 0.10 b 0.08
iteration 50: loss 0.61 w 1.73 b 1.71
iteration 100: loss 0.33 w 1.73 b 2.05
iteration 150: loss 0.24 w 1.63 b 2.23
iteration 200: loss 0.19 w 1.53 b 2.36
iteration 250: loss 0.16 w 1.47 b 2.45
iteration 300: loss 0.15 w 1.41 b 2.52
iteration 350: loss 0.14 w 1.37 b 2.58
iteration 400: loss 0.13 w 1.34 b 2.62
iteration 450: loss 0.13 w 1.32 b 2.65
##### final w,b 1.3033228991130752
2.6760184293088694
```

MLP (다층 퍼셉트론)

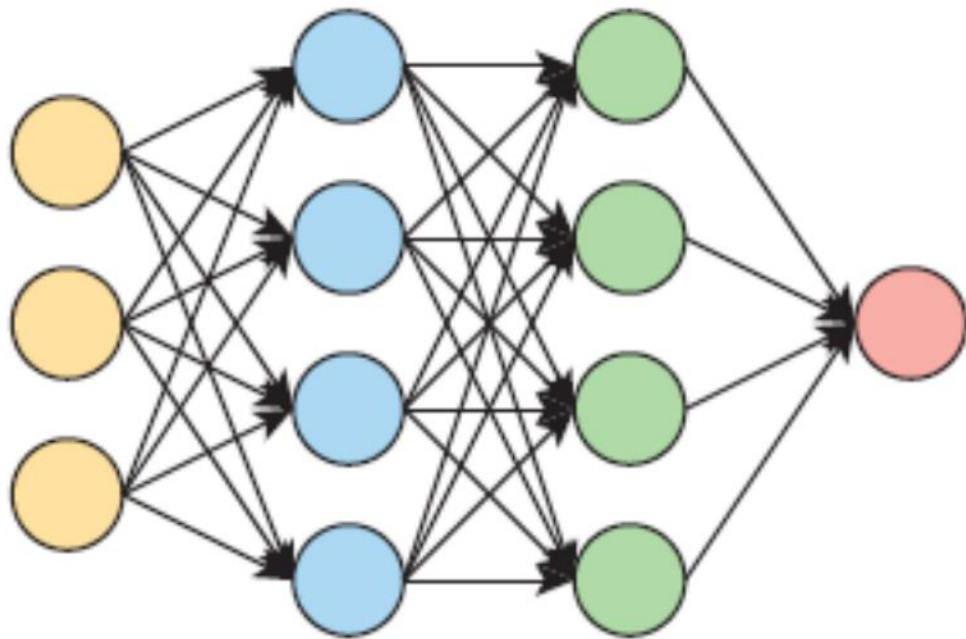
인하공전 컴퓨터 정보공학과



입력 층 : 2
은닉층 1 : 2
출력 층 : 1

MLP (다층 퍼셉트론): 파라미터 수

인하공전 컴퓨터 정보공학과



입력 층 : 3
은닉층 1 : 4 =>
은닉층 2 : 4 =>
출력 층 : 1 =>

현재 레이어 파라미터 수 : (앞 레이어 UNIT 개수) * (현재 레이어 유닛 갯수) + 현재 레이어 유닛 갯수

```
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)	
dense_9 (Dense)	(None, 1)	

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

실행 결과

	Id	MSSubClass	MSZoning	LotFrontage	LotArea	Street	Alley	LotShape	LandContour	Utilities	..
0	1	60	RL	65.0	8450	Pave	NaN	Reg	Lvl	AllPub	...
1	2	20	RL	80.0	9600	Pave	NaN	Reg	Lvl	AllPub	...
2	3	60	RL	68.0	11250	Pave	NaN	IR1	Lvl	AllPub	...
3	4	70	RL	60.0	9550	Pave	NaN	IR1	Lvl	AllPub	...
4	5	60	RL	84.0	14260	Pave	NaN	IR1	Lvl	AllPub	...
...
1455	1456	60	RL	62.0	7917	Pave	NaN	Reg	Lvl	AllPub	...
1456	1457	20	RL	85.0	13175	Pave	NaN	Reg	Lvl	AllPub	...
1457	1458	70	RL	66.0	9042	Pave	NaN	Reg	Lvl	AllPub	...
1458	1459	20	RL	68.0	9717	Pave	NaN	Reg	Lvl	AllPub	...
1459	1460	20	RL	75.0	9937	Pave	NaN	Reg	Lvl	AllPub	...

- 변수들의 범위를 $-1 \sim 1$ 의 범위로 맞춰주는것이 좋다

```
from sklearn.preprocessing import MinMaxScaler
data = [[-1, 2], [-0.5, 6], [0, 10], [1, 18]]

scaler = MinMaxScaler()
scaler.fit(data)          # 최대값과 최소값을 알아낸다.
print(scaler.transform(data))  # 데이터를 변환한다.
```

```
[[0.  0. ]
 [0.25 0.25]
 [0.5  0.5 ]
 [1.  1.  ]]
```

■ 과제 2

house_scale.ipynb 에서

X_data를 정규화 하고 데이터를 print 하는 부분 code를 작성하시오.

house_scale.ipynb, house_scale.py upload

와인 품질 예측 - 데이터

인하공전 컴퓨터 정보공학과

	A1											
		✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
		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	0.44	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

과제 3)

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))
```

과제 4)

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
```

wheat seeds 분류

인하공전 컴퓨터 정보공학과

과제 5:

데이터: wheat-seeds.csv

밀에서 추출한
종자의 측정값을 기준으로
밀의 종류 예측

7개의 입력특성

:A,P,C,LK,WK,A_coeff,LKG)

1개의 출력 변수

: target (0~3)

210개 sample

Code작성.

lpybn, *.py upload

A	P	C	LK	WK	A_Coef	LKG	target
15.26	14.84	0.871	5.763	3.312	2.221	5.22	1
14.88	14.57	0.8811	5.554	3.333	1.018	4.956	1
14.29	14.09	0.905	5.291	3.337	2.699	4.825	1
13.84	13.94	0.8955	5.324	3.379	2.259	4.805	1
16.14	14.99	0.9034	5.658	3.562	1.355	5.175	1
14.38	14.21	0.8951	5.386	3.312	2.462	4.956	1
14.69	14.49	0.8799	5.563	3.259	3.586	5.219	1
14.11	14.1	0.8911	5.42	3.302	2.7	5	1
16.63	15.46	0.8747	6.053	3.465	2.04	5.877	1
16.44	15.25	0.888	5.884	3.505	1.969	5.533	1
15.26	14.85	0.8696	5.714	3.242	4.543	5.314	1
14.03	14.16	0.8796	5.438	3.201	1.717	5.001	1
13.89	14.02	0.888	5.439	3.199	3.986	4.738	1

성능 기준

1. 0~3으로 분류 (4 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 0.95 보다 크게

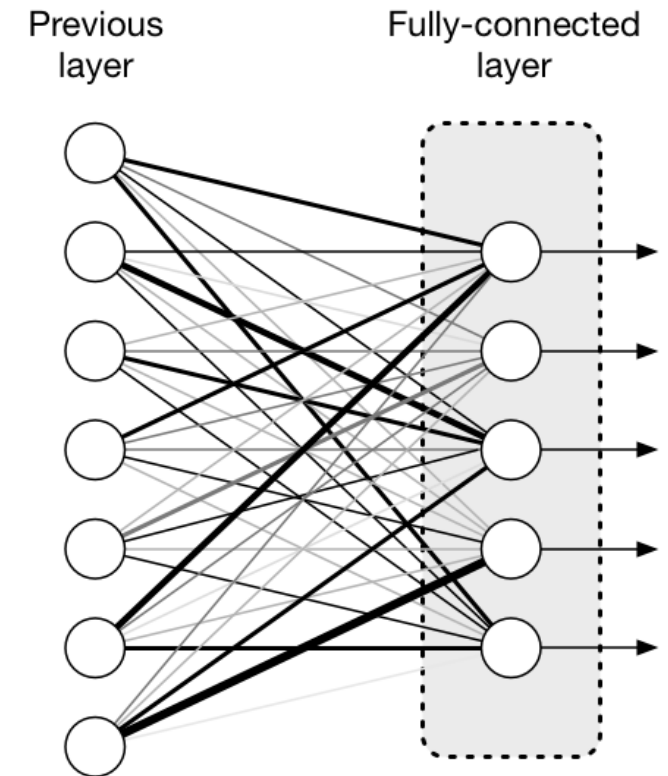
완전 연결 신경망 (Fully Connected Neural Network)

인공지능, 컴퓨터 공학과
인공지능, 컴퓨터 공학과

tf.keras.layers.**Dense**

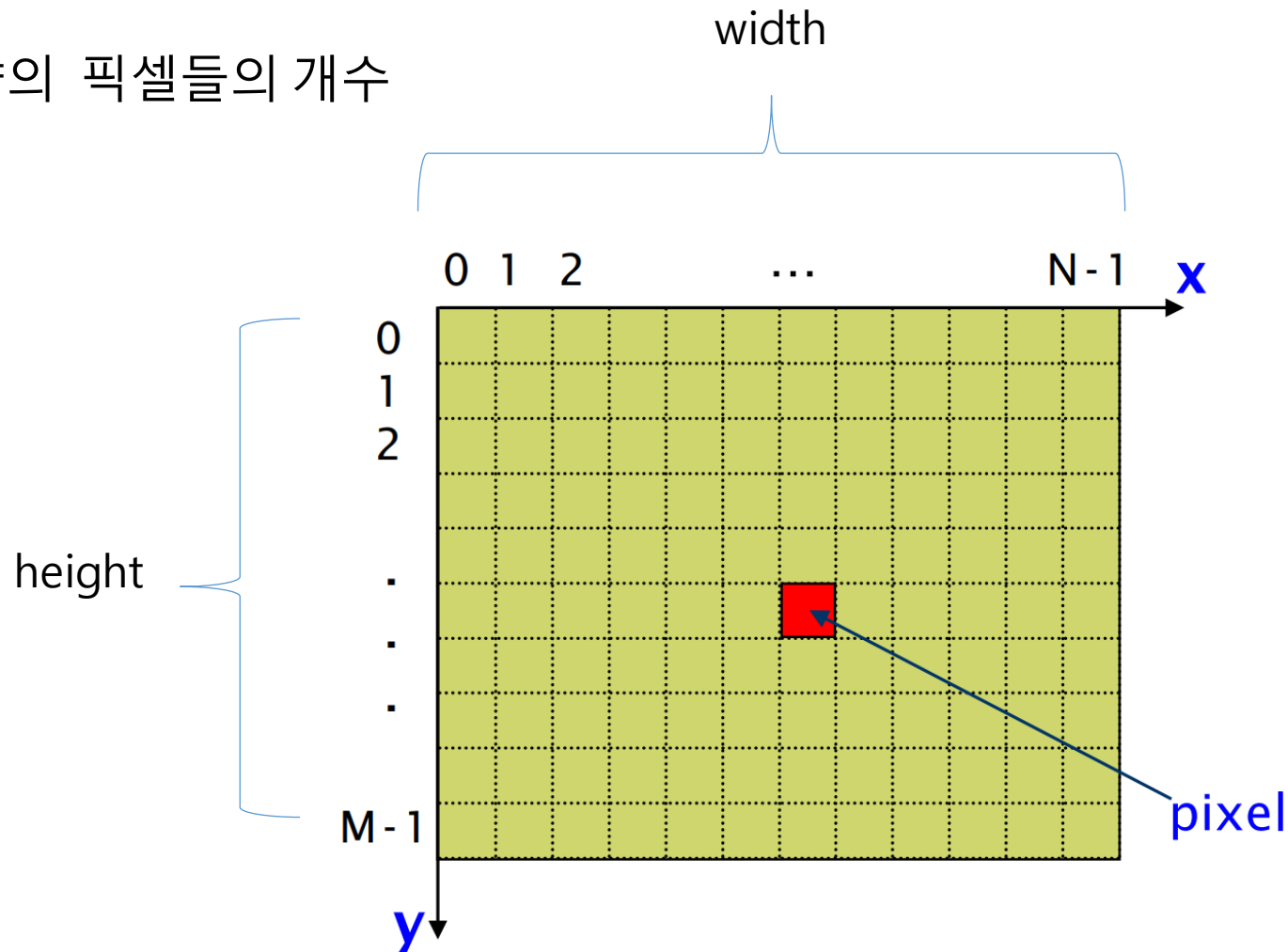
완전 연결 계층 : (Fully Connected Layer, Densely connected layer)

한 층 (layer)의 모든 뉴런이 그 다음 층의 모든 뉴런과 연결된 상태



https://keras.io/api/layers/core_layers/

- 영상은 픽셀로 이루어짐
- 해상도 (Resolution)
 - 2차원 공간 영역에서, x, y 축 방향의 픽셀들의 개수



영상 처리 기초

- 영상은 픽셀로 이루어짐.

- RGB color space

 - Red, Green, Blue

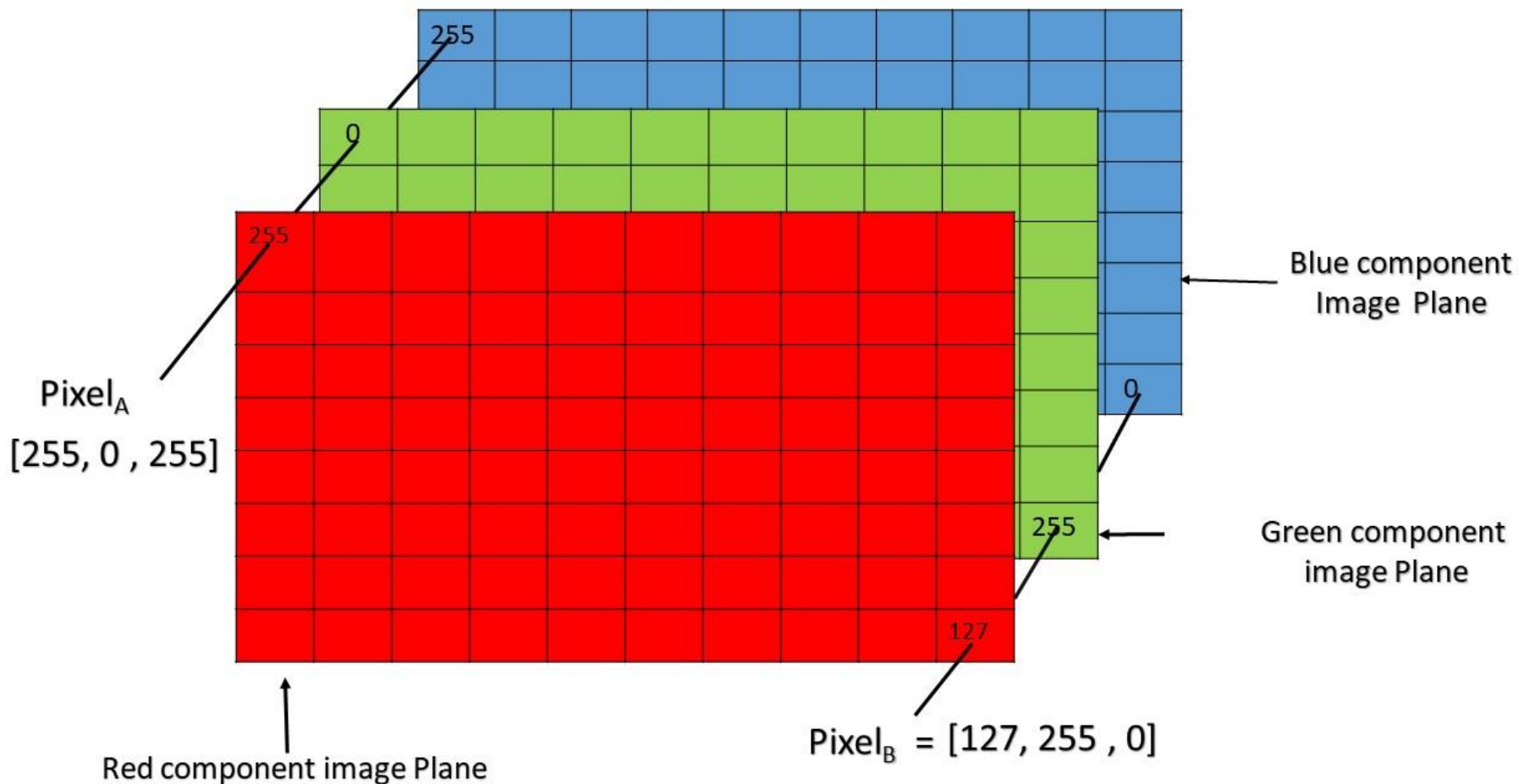
White = (255,255,255)

Black = (0,0,0)

Gray = (127,127,127)

Img.shape

=(height, width, rgb==3)



Pixel of an RGB image are formed from the corresponding pixel of the three component images



컬러 이미지

`Img.shape`

`=(height, width, rgb==3)`



그레이 이미지

`Img.shape`

`=(height, width)`

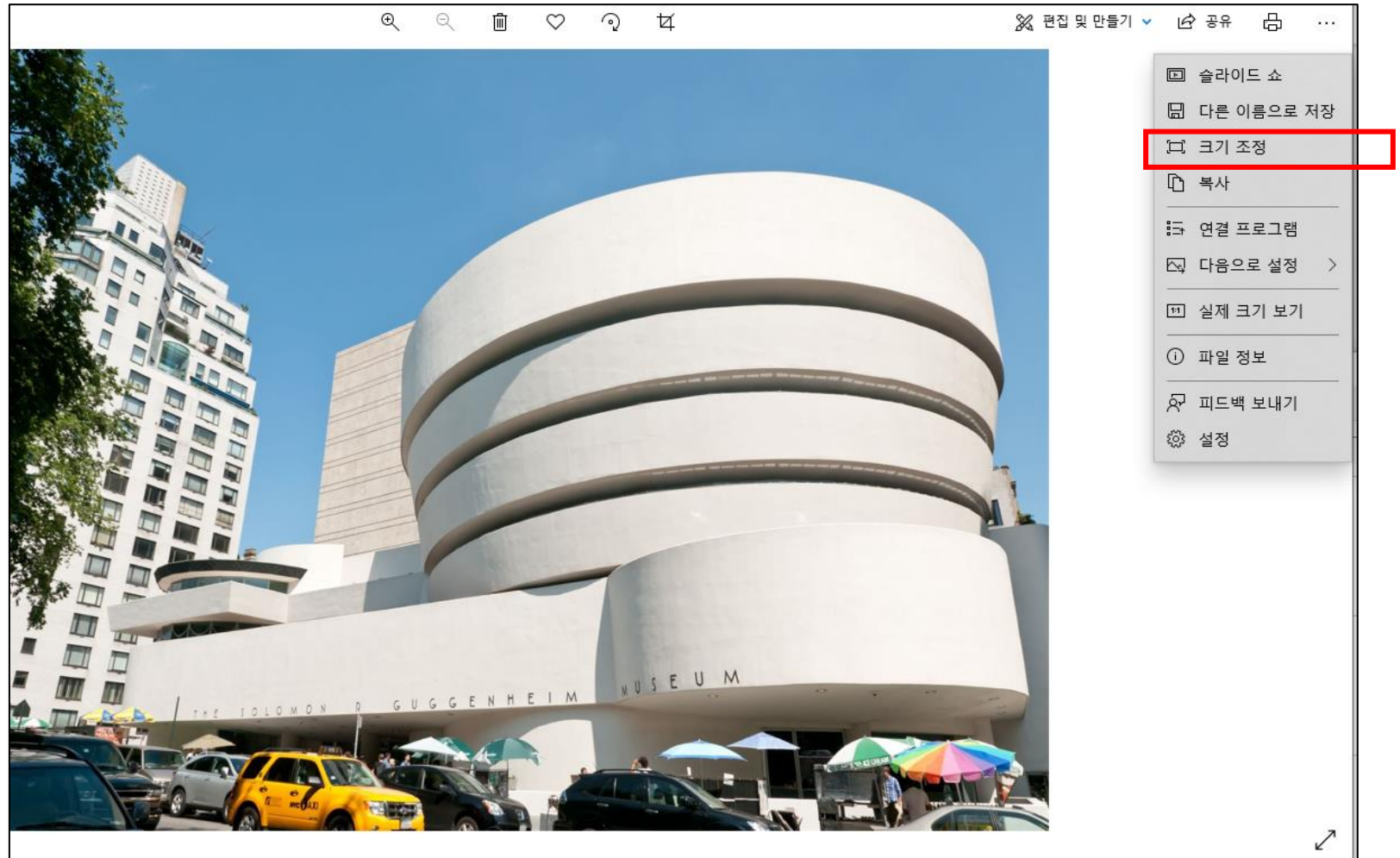
$$\text{Gray} = 0.299 \cdot R + 0.587 \cdot G + 0.114 \cdot B$$

영상 처리 기초 – Open CV 실습

인하공전 컴퓨터 정보공학과

본인이 찍어온 사진 or test1_1920.jpg 사용

본인이 찍어온 사진 크기 조정 => 1920x1440 과 비슷한 크기로



영상 처리 기초 – Open CV 실습

인하공전 컴퓨터 정보공학과

Colab에서 실습

```
import cv2
from google.colab.patches import cv2_imshow
img=cv2.imread('/content/test1_1920.jpg')
print('img.ndim =',img.ndim)
print('img.shape =',img.shape)
print('img.dtype = ',img.dtype)
#cv2_imshow(img)
img=cv2.resize(img, (0,0) ,fx=0.5,fy=0.5)
cv2.imwrite('test1_half.jpg',img)
print(img.shape)
cv2_imshow(img)
```

jupyter

cv2.imshow(img)

test1_half.jpg

test1_half_changed.jpg

test1_halfgray.jpg

test_halfgray_filt1.jpg

test_halfgray_filt2.jpg

test_halfgray_filt3.jpg

test_halfgray_filde.jpg

영상 처리 기초 – Open CV 실습

인하공전 컴퓨터 정보공학과

Colab에서 실습

#colab

```
cv2_imshow(img)
```

#jupyter

```
cv2.imshow(img)
```

Colab에서 실습

```
import cv2
from google.colab.patches import cv2_imshow
img=cv2.imread('/content/test1_1920.jpg')
print('img.ndim =',img.ndim)
print('img.shape =',img.shape)
print('img.dtype = ',img.dtype)
#cv2_imshow(img)
img=cv2.resize(img, (0,0) ,fx=0.5,fy=0.5)
cv2.imwrite('test1_half.jpg',img)
print(img.shape)
cv2_imshow(img)
```

```
img.ndim = 3
img.shape = (1440, 1920, 3)
img.dtype = uint8
```

```
(720, 960, 3)
```

영상 처리 기초 – Open CV 실습

인하공전 컴퓨터 정보공학과

```
img=cv2.imread('/content/test1_half.jpg')
img[100:300,100:300,:]=255
img[100:300,300:500,:]=0
img[100:300,500:700,:]=127

img[300:500,100:300,0]=255
img[300:500,300:500,1]=255
img[300:500,500:700,2]=255

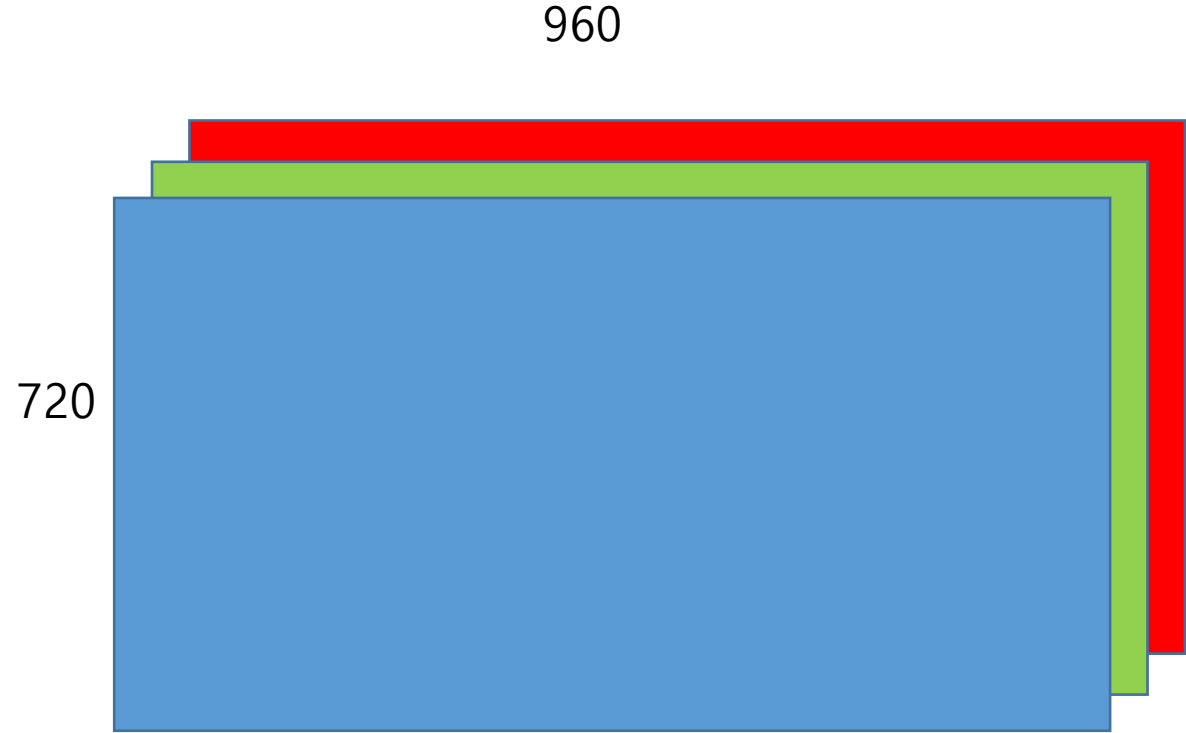
cv2.imwrite('test1_half_changed.jpg',img)

cv2_imshow(img)
```

White= (255,255,255)

Black = (0,0,0)

Gray =(127,127,127)



수고하셨습니다

jhmin@inhatec.ac.kr