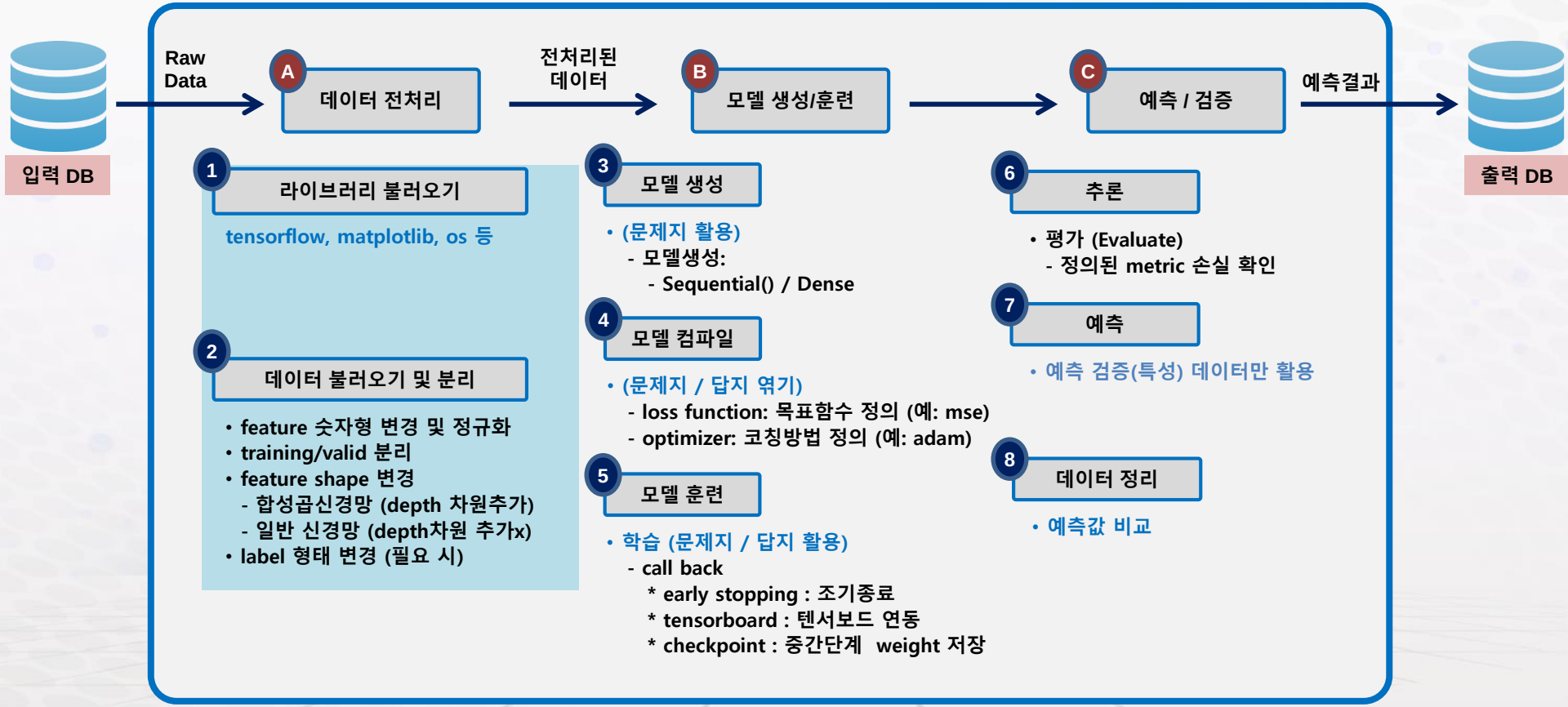
참조. 딥러닝 작동 원리


입력 DB
Raw Data
A
데이터 전처리
전처리된 데이터
B
모델 생성/훈련
C
예측 / 검증
예측결과
출력 DB
1
라이브러리 불러오기
tensorflow, matplotlib, os 등
2
데이터 불러오기 및 분리
• feature 숫자형 변경 및 정규화
• training/valid 분리
• feature shape 변경
- 합성곱신경망 (depth 차원추가)
- 일반 신경망 (depth차원 추가x)
• label 형태 변경 (필요 시)
3
모델 생성
• (문제지 활용)
- 모델생성:
- Sequential() / Dense
4
모델 컴파일
• (문제지 / 답지 엮기)
- loss function: 목표함수 정의 (예: mse)
- optimizer: 코칭방법 정의 (예: adam)
5
모델 훈련
• 학습 (문제지 / 답지 활용)
- call back
* early stopping : 조기종료
* tensorboard : 텐서보드 연동
* checkpoint : 중간단계 weight 저장
6
추론
• 평가 (Evaluate)
- 정의된 metric 손실 확인
7
예측
• 예측 검증(특성) 데이터만 활용
8
데이터 정리
• 예측값 비교

감정인식모델 생성 후 실시간캠과 연동하기

김 효 관

교육목표: 이미지를 학습한 후 실시간 영상에서 학습한 내용을 활용하여 예측

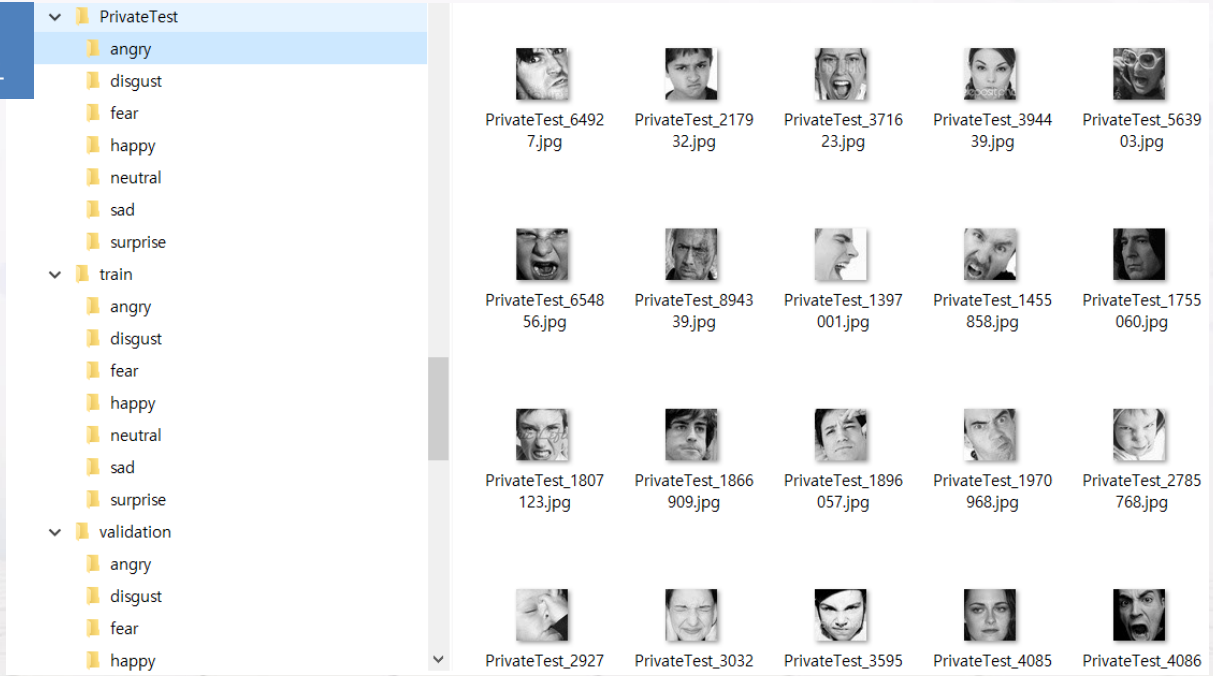
CONTENTS

1. 합성곱 신경망 (CNN) 모델 생성
2. 얼굴인식 모델 (haarface detection)
3. 얼굴인식 (haar) + 예측모델
4. 핵심정리 및 Q&A

1. 합성곱 신경망(CNN) 모델 생성

1. 이미지 준비

제공된 이미지를
images 폴더 내 저장하세요



1. 합성곱 신경망(CNN) 모델 생성

데이터 불러오기 전략

[훈련/테스트 세트] : 이미지/답지 별도 저장

폴더 리스트 반복

angry, disgust ...

이미지 리스트 반복

img_4.jpg, ..

데이터 저장

- 컬러: 그레이
- 사이즈: 48 * 48
- 이미지 저장
- 폴더 인덱스-> 답지

```
1 img
```

```
array([[  3,   0,   0,   3,   7,   3,   0,   3,   0,  11,   0,   0,   3,
          0,   0,   3,   8,   0,   0,   3,   0,   0,   0,   2,   0,   0,
          0,   0],
       [  0,   0,   0,   0,   0,   0,   0,   1,   5,   0,  12,   0,  16,
          0,   0,   4,   0,   2,   8,   3,   0,   4,   8,   0,   0,   0,
          0,   0],
```

1행

2행

```
EMOTIONS_LIST = ["Angry", "Disgust",
                 "Fear", "Happy",
                 "Neutral", "Sad",
                 "Surprise"]
```

1. 합성곱 신경망(CNN) 모델 생성

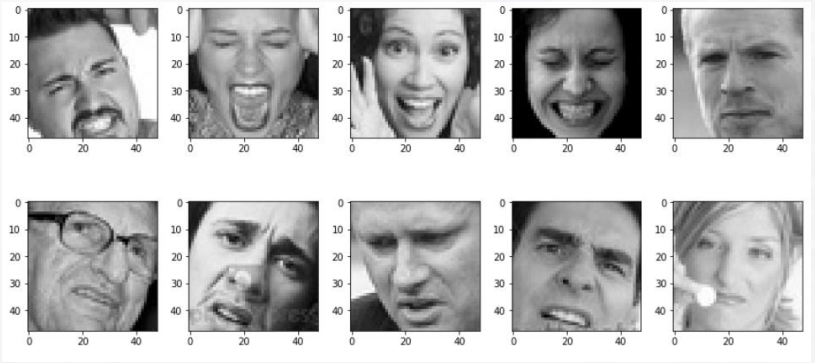
2. 라이브러리 정의

```
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import os
%matplotlib inline
```

1. 합성곱 신경망(CNN) 모델 생성

3. 이미지 확인

```
# size of the image: 48*48 pixels
pic_size = 48

# input path for the images
base_path = "../images/emotion_f/input/"

plt.figure(0, figsize=(12,20))
cpt = 0

for expression in os.listdir(base_path + "train/"):
    for i in range(1,6):
        cpt = cpt + 1
        plt.subplot(7,5,cpt)
        img = load_img(base_path + "train/" + expression + "/" +os.listdir(base_path + "train/" + expression)[i],
target_size=(pic_size, pic_size))
        plt.imshow(img, cmap="gray")

plt.tight_layout()
plt.show()
```



1. 합성곱 신경망(CNN) 모델 생성

4. 훈련 데이터 생성

```
TRAIN_DIR = '../images/realtime/input/Training/'
train_folder_list = array(os.listdir(TRAIN_DIR))
train_folder_list

IMG_SIZE = 48
train_images=[]
train_labels=[]
for index in range(0, len(train_folder_list)):
    path = os.path.join(TRAIN_DIR, train_folder_list[index])
    path = path + '/'
    img_list = os.listdir(path)
    for img in img_list:
        img_path = os.path.join(path, img)
        try:
            img = cv2.imread(img_path, cv2.IMREAD_GRAYSCALE)
            new_img = cv2.resize(img, (IMG_SIZE, IMG_SIZE))
            train_images.append(new_img)
            train_labels.append(index)
        except:
            pass
```

이미지가 저장되어있는 폴더를 클래스로 인식하여 이미지를 불러옴


```
EMOTIONS_LIST = ["Angry", "Disgust",
                 "Fear", "Happy",
                 "Neutral", "Sad",
                 "Surprise"]
```

1. 합성곱 신경망(CNN) 모델 생성

4. 훈련 데이터 생성

```
train_data = array(train_images)
train_labels = array(train_labels)

array(train_data).shape

w, h = 48, 48
x_train = train_data.reshape(train_data.shape[0], w, h,1 )

from keras.utils import to_categorical

# Change the labels from integer to categorical data
train_labels_one_hot = to_categorical(train_labels)

# Display the change for category label using one-hot encoding
print('Original label 0 : ', train_labels[100000])

y_train = train_labels_one_hot

modelDim = x_train[0].shape
modelDim
```

```
train_data = array(train_images)
train_labels = array(train_labels)
```

```
array(train_data).shape
```

```
(114826, 48, 48)
```

```
w, h = 48, 48
x_train = train_data.reshape(train_data.shape[0], w, h,1 )
```

```
from keras.utils import to_categorical
```

```
# Change the labels from integer to categorical data
train_labels_one_hot = to_categorical(train_labels)


# Display the change for category label using one-hot encoding
print('Original label 0 : ', train_labels[100000])
```

```
Original label 0 :  6
```

```
y_train = train_labels_one_hot
```

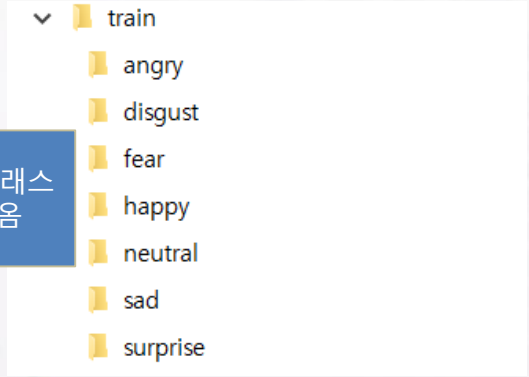
```
modelDim = x_train[0].shape
modelDim
```

1. 합성곱 신경망(CNN) 모델 생성

4. ImageDataGenertor 활용 훈련/테스트 데이터 생성

```
from keras.preprocessing.image import ImageDataGenerator

# number of images to feed into the NN for every batch
batch_size = 128

datagen_train = ImageDataGenerator()
datagen_validation = ImageDataGenerator()

train_generator = datagen_train.flow_from_directory(base_path + "train",
                                                    target_size=(pic_size,pic_size),
                                                    color_mode="grayscale",
                                                    batch_size=batch_size,
                                                    class_mode='categorical',
                                                    shuffle=True)
validation_generator = datagen_validation.flow_from_directory(base_path + "validation",
                                                    target_size=(pic_size,pic_size),
                                                    color_mode="grayscale",
                                                    batch_size=batch_size,
                                                    class_mode='categorical',
                                                    shuffle=False)
```

이미지가 저장되어있는 폴더를 클래스로 인식하여 이미지를 불러옴



1. 합성곱 신경망(CNN) 모델 생성

5. 케라스 모델 정의

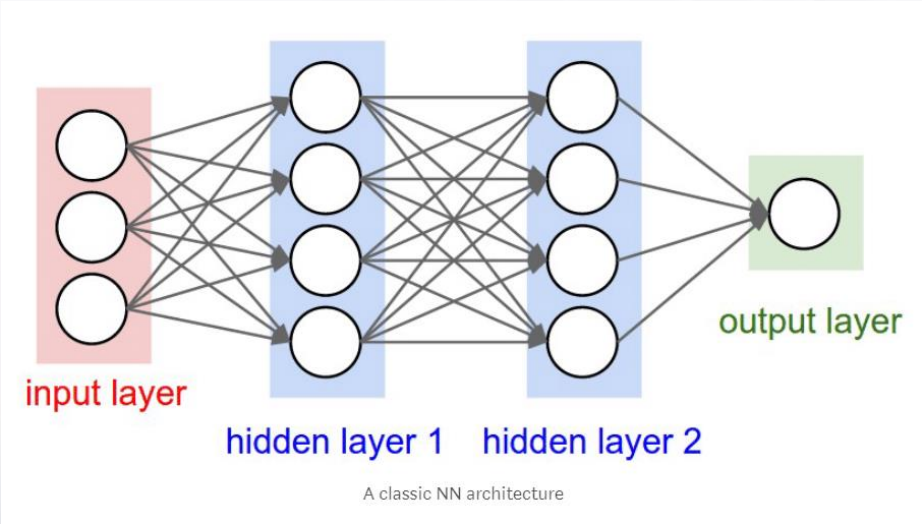
레이어	레이어	개수	특성
1	CNN	4개	이미지 특성 추축
2	Dense (Fully Connected Layer)	2개	7개 클래스를 위한 멀티클래스 추출 레이어

* Sequential 로 정의 : 층층이 계속 레이어를 쌓아감

1. 합성곱 신경망(CNN) 모델 생성

5. 케라스 모델 정의

```
from keras.layers import Dense, Input, Dropout,
GlobalAveragePooling2D, Flatten, Conv2D,
BatchNormalization, Activation, MaxPooling2D
from keras.models import Model, Sequential
from keras.optimizers import Adam

# label 값 (클래스 개수)
nb_classes = len(set(train_generator.classes))

# 기본 Sequential 방식으로 레이어를 한개씩 쌓음
model = Sequential()

# 1 - Convolution
model.add(Conv2D(64,(3,3), padding='same',
input_shape=(48, 48,1)))
model.add(BatchNormalization())
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))
```



1. 합성곱 신경망(CNN) 모델 생성

5. 케라스 모델 정의

```
# 2nd Convolution layer
model.add(Conv2D(128,(5,5), padding='same'))
model.add(BatchNormalization())
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))

# 3rd Convolution layer
model.add(Conv2D(512,(3,3), padding='same'))
model.add(BatchNormalization())
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))

# 4th Convolution layer
model.add(Conv2D(512,(3,3), padding='same'))
model.add(BatchNormalization())
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))
```

1. 합성곱 신경망(CNN) 모델 생성

5. 케라스 모델 정의

```
# Flattening
model.add(Flatten())

# Fully connected layer 1st layer
model.add(Dense(256))
model.add(BatchNormalization())
model.add(Activation('relu'))
model.add(Dropout(0.25))

# Fully connected layer 2nd layer
model.add(Dense(512))
model.add(BatchNormalization())
model.add(Activation('relu'))
model.add(Dropout(0.25))

model.add(Dense(nb_classes, activation='softmax'))

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

1. 합성곱 신경망(CNN) 모델 생성

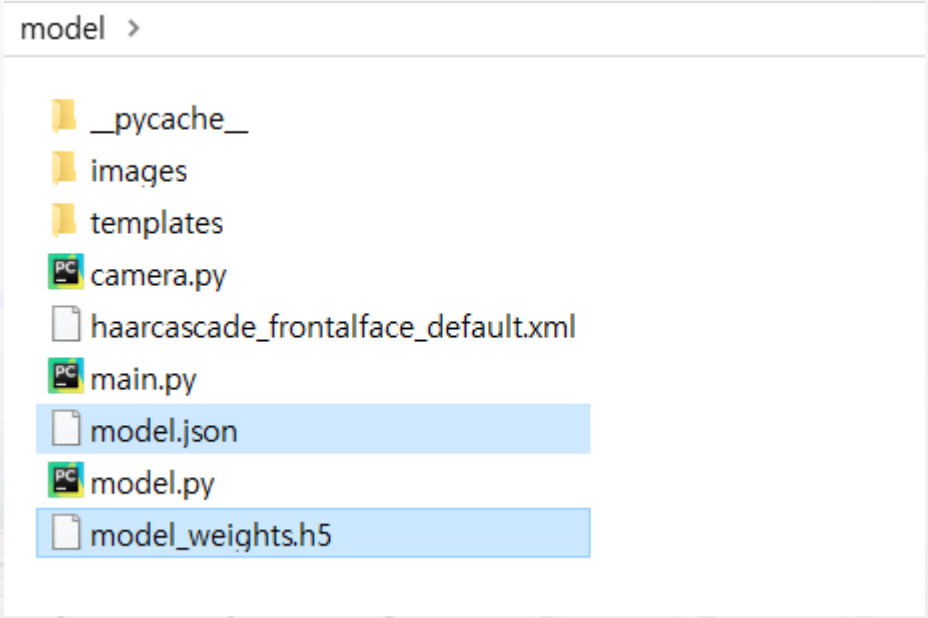
6. 모델 훈련

```
# number of epochs to train the NN
epochs = 3

from keras.callbacks import ModelCheckpoint

checkpoint = ModelCheckpoint("model_weights3.h5", monitor='acc', verbose=1, save_best_only=True,
mode='max')
callbacks_list = [checkpoint]

#checkpointer = ModelCheckpoint(filepath='model.weights.best.hdf5', verbose = 1, save_best_only=True)
loaded_model.fit(x_train,
        y_train,
        batch_size=64,
        epochs=epochs,
        callbacks=callbacks_list
        )
```

1. 합성곱 신경망(CNN) 모델 생성

7. 모델 구조 저장

```
# 모델 구조 Json 파일로 저장
model_json = model.to_json()
with open("model.json", "w") as json_file:
    json_file.write(model_json)
```



1. 합성곱 신경망(CNN) 모델 생성

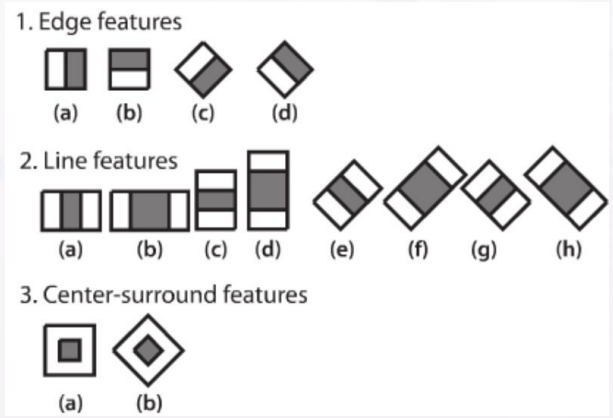
8. 테스트 파일 예측

```
import cv2

testimg = cv2.imread(₩
    "../images/facedetec2t_webcam.png", cv2.IMREAD_GRAYSCALE)

testimg.shape

testimg2 = testimg.reshape(1,48,48,1)

testprd = loaded_model.predict(testimg2)

EMOTIONS_LIST = ["Angry", "Disgust",
                 "Fear", "Happy",
                 "Neutral", "Sad",
                 "Surprise"]

import numpy as np

EMOTIONS_LIST[np.argmax(testprd)]
```

2. 얼굴인식 모델 (haar face detection)

haar face detection

1. 머신러닝 기반 오브젝트 검출 알고리즘 (얼굴인지 등)
2. 2001년 특징기반 비디오 내 오브젝트 검출



- haar feature selection
- creating integral images
- adaboost training
- cascading classifier

https://docs.opencv.org/4.1.0/dc/d88/tutorial_traincascade.html

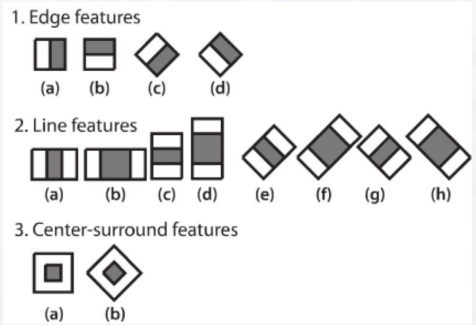
2. 얼굴인식 모델 (haar face detection)

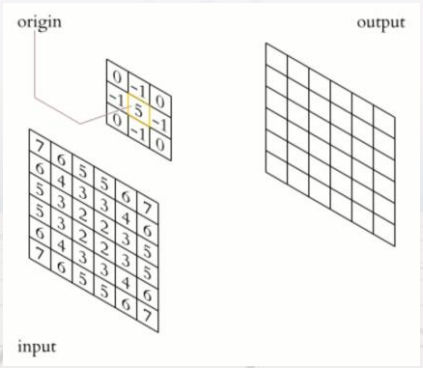
1. haar cascade

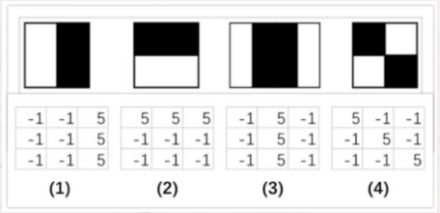
1. haar 특징 계산 (cnn가 유사)
2. 계산 방식: 검은색 영역에서 흰색영역 픽셀 뺀 값

haar feature selection
↓
creating integral images
↓
adaboost training
↓
cascading classifier

Haar-Feature는 CNN의 필터 커널과 유사
CNN에서는 커널 값이 훈련에 의해 결정
Haar-Feature는 커널 값 수동결정







2. 얼굴인식 모델 (haar face detection)

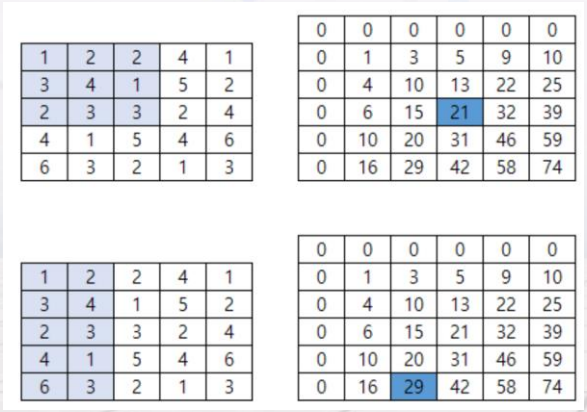
2. integral images

haar feature selection
↓
creating integral images
↓
adaboost training
↓
cascading classifier

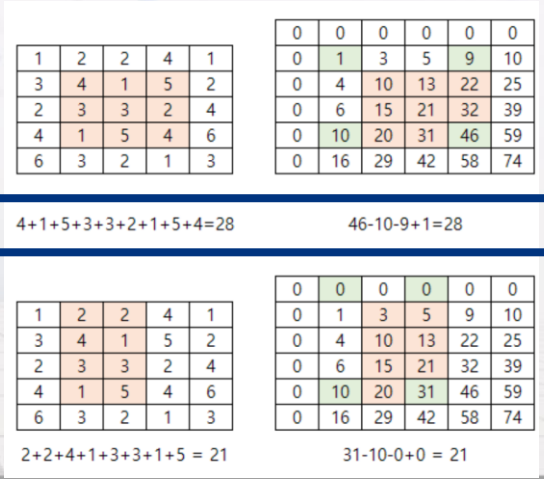
1 픽셀 합 계산(하얀색, 검정색) 시 빠르게 연산 가능

2 2001년 특징기반 비디오 내 오브젝트 검출

본 이미지에 적분 값 맵핑 후좌측 및 상단에 0채움



계산: 적분합 – 양끝단 + 초기점



2. 얼굴인식 모델 (haar face detection)

3. adaboost

haar feature selection
↓
creating integral images
↓
adaboost training
↓
cascading classifier

1. 영역별 최적 haar 특징 찾기
2. haar cascade param (160,000 + features), 24*24

- 모든 haar 특징에 동일 가중치 적용
- 잘못분류 하르 특징에 가중치 증가
- 성능 좋은 haar 특징은 낮은 에러률 (낮은에어률 haar 특징 선택)

line feature(세로)는
코 탐색 시 유의미 하지만
입술 탐색 시에는 무의미 하다.

line feauture(가로)는
눈 주위가 더 어둡다는 특징으로 탐색
(Adaboost는 가장 최상의 기능만 활용)

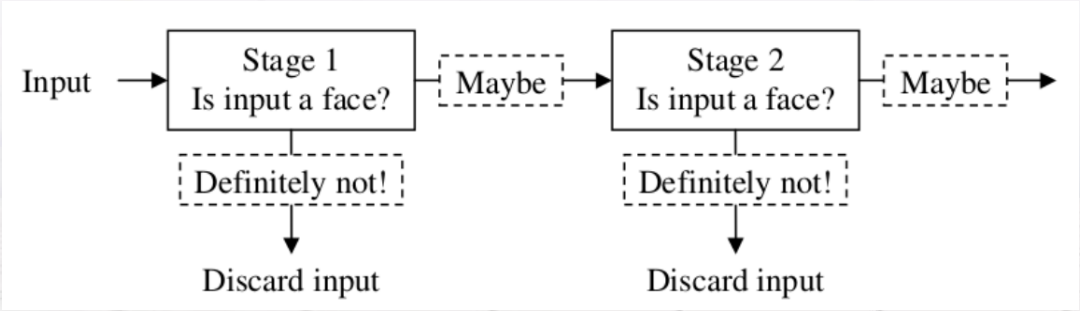
2. 얼굴인식 모델 (haar face detection)

4. cascading

haar feature selection
↓
creating integral images
↓
adaboost training
↓
cascading classifier

1 현재 영역에 대한 얼굴영역 유/무를 단계적으로 체크

2 낮은 단계에서는 빠른탐색, 상위 단계에서는 세밀하게 탐색



2. 얼굴인식 모델 (haar face detection)

라이브러리 선언 및 비디오 캡쳐

```
import cv2
import sys

# 정면 얼굴인식 모델 미리 학습시켜 놓은 XML 포맷으로 저장된 분류기를 로드합니다.
faceCascade = cv2.CascadeClassifier('haarcascade_frontalface_default.xml')

video_capture = cv2.VideoCapture(0)
```

2. 얼굴인식 모델 (haar face detection)

프레임별 화면 출력

```
while video_capture.isOpened:

    ret, frame = video_capture.read()
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    k = cv2.waitKey(1)
    faces = faceCascade.detectMultiScale(
        gray, scaleFactor=1.5, minNeighbors=5,
        minSize=(30, 30),
        flags=cv2.CASCADE_SCALE_IMAGE
    )

    # 인식된 얼굴 주위에 사각형 영역 표시
    for (x, y, w, h) in faces:
        cv2.rectangle(frame, (x, y), (x+w, y+h), (0, 255, 0), 2)

    # 화면 출력
    cv2.imshow('FaceDetection', frame)
    if k == 27: #ESC Pressed
        break
    elif k == 32:  # SPACE pressed
        img_name = "facedetect_webcam.png"
        cv2.imwrite(img_name, frame)
```

2. 얼굴인식 모델 (haar face detection)

종료

```
# 종료 후 캡쳐 릴리즈 및 윈도우 종료
video_capture.release()
cv2.destroyAllWindows()
```

3. 얼굴인식(haar) + 예측모델

1. 저장된 모델 구조 및 weight 불러오기

```
import warnings
warnings.filterwarnings("ignore")

from keras.models import model_from_json
import numpy as np


EMOTIONS_LIST = ["Angry", "Disgust", "Fear", "Happy","Neutral", "Sad", "Surprise"]

model_json_file ="./model.json"
model_weights_file = "./model_weights.h5"
with open(model_json_file, "r") as json_file:
    loaded_model_json = json_file.read()
    loaded_model = model_from_json(loaded_model_json)

# load weights into the new model
loaded_model.load_weights(model_weights_file)
#loaded_model._make_predict_function()
```

3. 얼굴인식(haar) + 예측모델

2. 얼굴인식 모델 불러오기 및 캡쳐 생성

```
import cv2
import sys

font = cv2.FONT_HERSHEY_SIMPLEX
faceCascade = cv2.CascadeClassifier('haarcascade_frontalface_default.xml')

video_capture = cv2.VideoCapture(0)
```

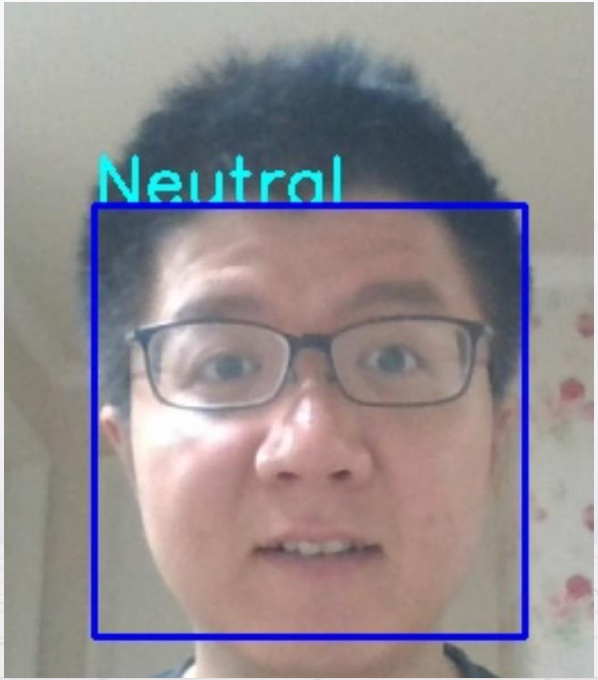
3. 얼굴인식(haar) + 예측모델

3. 예측

```
while video_capture.isOpened:

    ret, frame = video_capture.read()
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    k = cv2.waitKey(1)
    faces = faceCascade.detectMultiScale(
        gray, scaleFactor=1.5, minNeighbors=5,
        flags=cv2.CASCADE_SCALE_IMAGE
    )
     gray_fr = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)

    # 인식된 얼굴 주위에 사각형 영역 표시
    for (x, y, w, h) in faces:
          fc = gray_fr[y:y+h, x:x+w]
          roi = cv2.resize(fc, (48, 48))
          predict = loaded_model.predict(roi.reshape(1,48,48,1))
          pred = EMOTIONS_LIST[np.argmax(predict)]
          cv2.putText(frame, pred, (x, y), font, 1, (255, 255, 0), 2)
        cv2.rectangle(frame, (x, y), (x+w, y+h), (0, 255, 0), 2)

# 화면 출력
    cv2.imshow('FaceDetection', frame)

    if k == 27: #ESC Pressed
        break
```

3. 얼굴인식(haar) + 예측모델

4. 실행결과 확인



3. 얼굴인식(haar) + 예측모델

나이, 성별 등의 추가 예측모델을 생성한 후
실제 웹캠과 연동하는 모델을 생성하세요.
(프로젝트 시간에 실습)

5. 핵심정리 및 Q&A

기억합시다

1. CNN (합성곱 신경망), Keras, OpenCV를 라이브러리를 이해한다.
2. 실습을 통해 모델 훈련 및 훈련결과를 활용하여 예측하는 방법을 이해한다.
3. OpenCV를 활용하여 실시간 웹캠 연동 및 훈련결과를 연동하는 방법을 익힌다.

감사합니다.