

# Apache Kafka, OpenCV를 이용한 영상처리

Open GIST Science Lab 2025

❖ 본 실험 강의는 아래와 같은 환경에서 진행되었습니다. ❖

## ❖ 실험 환경

- ✓ NUC 11TNHi5
- ✓ Raspberry Pi 4
- ✓ Raspberry Pi Camera 2

## ❖ 실험 OS

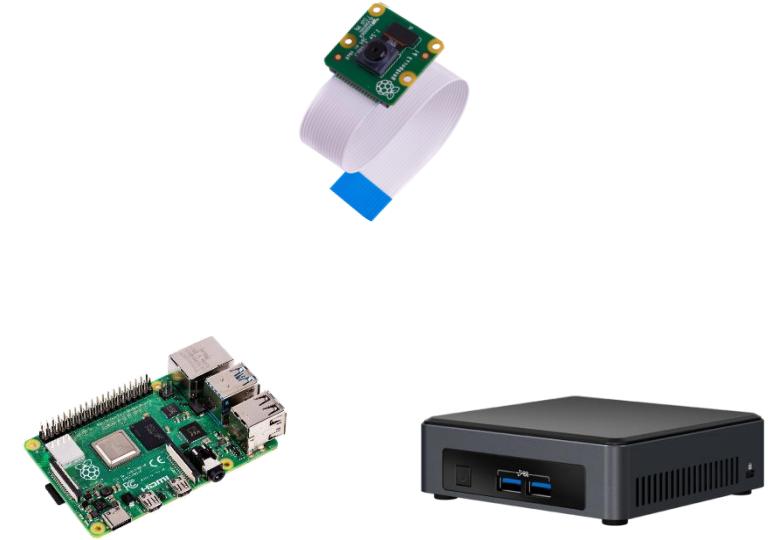
- ✓ Ubuntu 22.04.4 LTS
- ✓ Raspberry Pi OS Bookworm

## ▣ 프로그램 버전

- ✓ Python 3.10.12 ✓ Kafka 2.8.0
- ✓ kafka-python 2.0.2 ✓ opencv-python 4.4.0.46
- ✓ imutils 0.5.4 ✓ yt-dlp 2024.7.9 ✓ dlib 19.22.0
- ✓ opencv-python 4.10.0.84 ✓ face-recognition 1.3.0

# 장비 소개 카메라 세팅

Raspberry Pi & NUC  
실험 기본 세팅  
라즈베리 파이 카메라 연결



# Raspberry Pi & NUC



## 라즈베리파이 (Raspberry Pi)

- 영국의 라즈베리 파이 재단에서 교육용으로 만든 초소형, 초저가의 컴퓨터
- 싱글 보드 컴퓨터(SBC)로, 한 PCB(인쇄 회로 기판)위에 메모리, 입출력 기능을 위한 최소 컴퓨터 요구 사항을 포함



## 라즈베리 파이 카메라 2 (Raspberry Pi Camera 2)

- 라즈베리파이 전용 카메라



## NUC

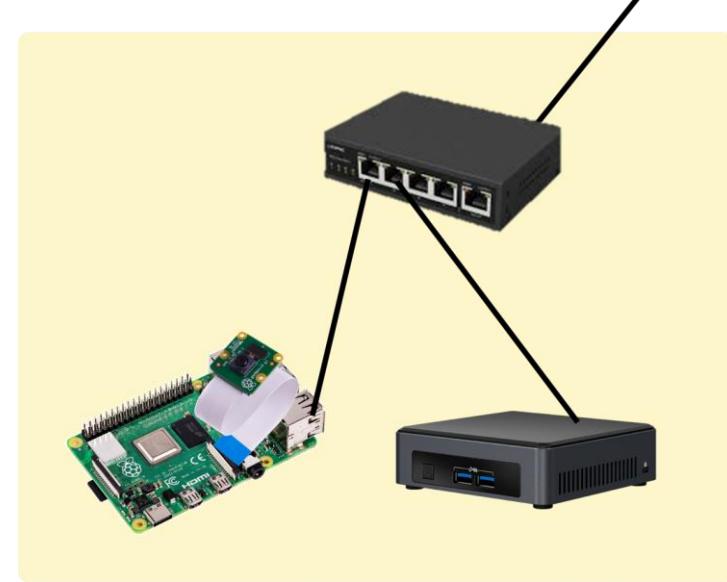
- 인텔에서 출시한 미니 PC 플랫폼
- 작은 크기로, 라즈베리 파이보다 더 좋은 성능을 지녔으며, 일반 PC로 사용

# 실험 환경

## ■ 스위치를 통한 Pi와 NUC 연결

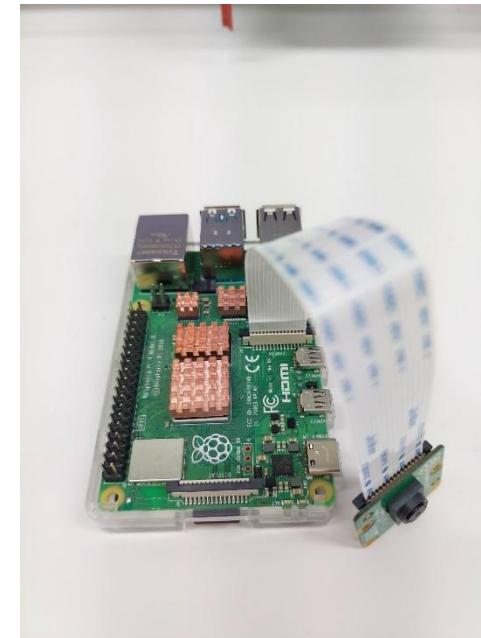
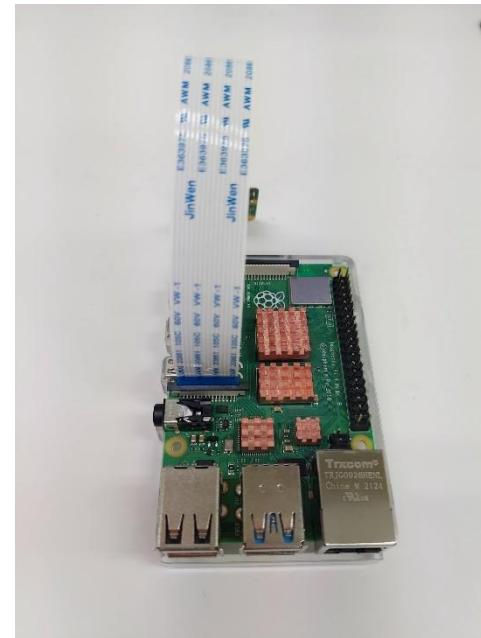
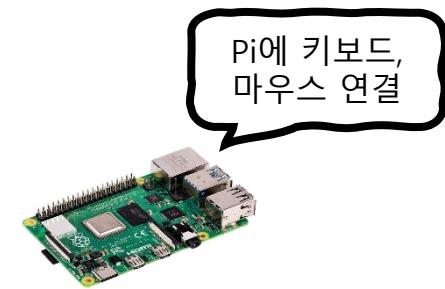
- 스위치는 자신에게 연결된 디바이스들의 MAC Address와 포트가 기록된 표를 갖고 있고, 이 표를 사용해 데이터의 목적지를 파악해 해당하는 장치에 데이터를 전송
- 스위치를 사용하여 Raspberry Pi와 NUC을 네트워크에 연결

## □ Pi, NUC을 제어할 모니터, 키보드, 마우스



# Raspberry Pi에 카메라 연결(1)

- ❖ 전원이 꺼진 상태에서 Raspberry Pi에 카메라 모듈을 연결 후 부팅





# Raspberry Pi에 카메라 연결(2)

## 🎥 Python으로 Raspberry Pi Camera의 영상 녹화하기 📹

- 필요한 패키지를 설치 : VIM 편집기, Python 패키지 관리자
- APT (Advanced Packaging Tool) : Linux 시스템에서 필요한 패키지를 설치하고 제거하는데 사용

```
pi@raspberrypi:~$ sudo apt update
```

apt update는 설치되어 있는 패키지들의  
새로운 버전이 있는지 확인할 때 사용

```
pi@raspberrypi:~$ sudo apt install vim
```

- VIM 편집기 실행

```
pi@raspberrypi:~$ sudo vi record_video.py
```

# Raspberry Pi에 카메라 연결(3)



## 🎥 Python으로 Raspberry Pi Camera의 영상 녹화하기 🎥

- Vi에서 i를 눌러서 편집 모드 진입, 그 후 아래와 같이 작성(한글 설명은 작성하지 않음)

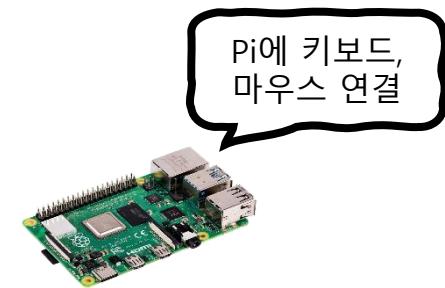
```
from picamera2.encoders import H264Encoder      PiCamera2 library 호출 하여 Raspberry Pi에 연결
from picamera2 import Picamera2
import time

picam2 = Picamera2()                                카메라를 다룰 수 있는 camera 요소를 만듬
video_config = picam2.create_video_configuration()
picam2.configure(video_config)                      카메라의 영상 입력을 확인하고, h264 코덱으로 비디오 녹화를 시작

encoder = H264Encoder(bitrate=10000000)
output = "my_video.h264"                            5초 동안 동작을 멈춤(sleep). 이 부분을 수정하여 비디오 녹화시간 지정 가능

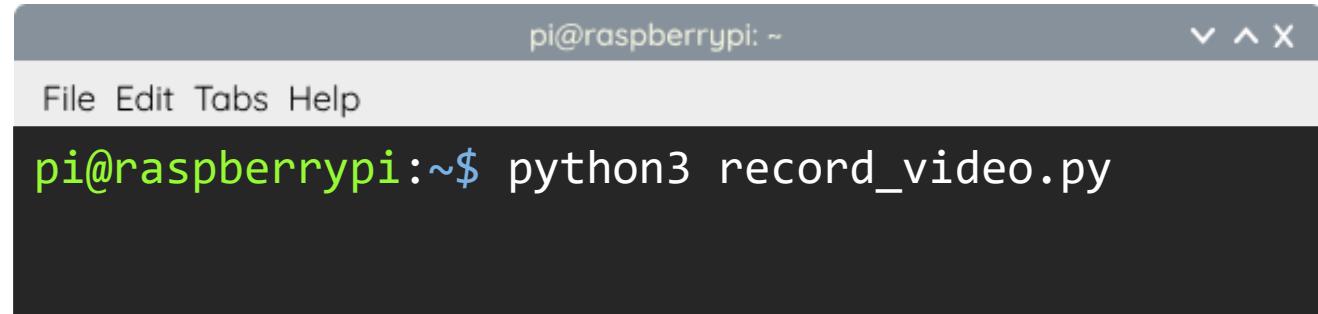
picam2.start_recording(encoder, output)
time.sleep(5)                                       sleep에서 정의된 시간이 흐른 뒤, 카메라 영상 녹화와 미리보기를 멈춤
picam2.stop_recording()
```

# Raspberry Pi에 카메라 연결(4)



🎥 Python으로 Raspberry Pi Camera의 영상 녹화하기 🎥

- python code를 실행한 뒤, 녹화된 영상을 확인



```
pi@raspberrypi: ~
File Edit Tabs Help
pi@raspberrypi:~$ python3 record_video.py
```

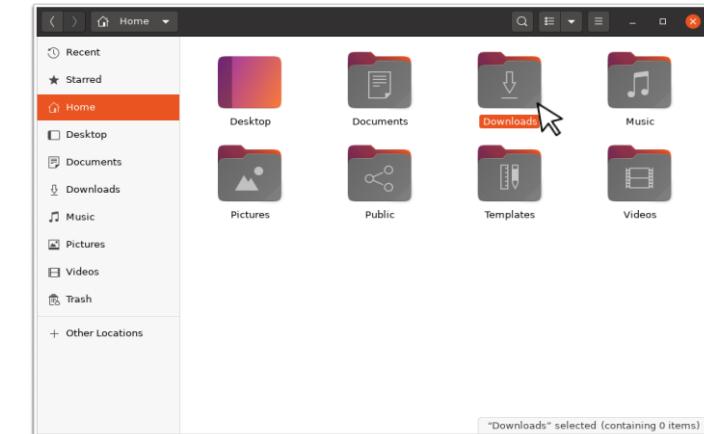
# Vim 소개



- ❖ Vim(Vi improved) : UNIX CLI(Command Line Interface) 환경에서 사용하는 편집기(Editor)
  - ❖ 입력 모드 : 원하는 글자를 입력
    - **a** : 현재 위치의 다음부터 입력 시작
    - **i** : 현재 위치의 앞에서부터 입력 시작
  - ❖ 명령 모드 : 문서 편집을 할 수 있으며, 입력 모드 상태에서 ESC키를 누르면 명령 모드로 전환됨
    - **x** : 커서가 있는 문자 삭제
    - **dd** : 현재 줄 전체 삭제
  - ❖ 라인 모드 : ESC 키를 누른 후 colon(:) prompt에서 명령을 입력하며 저장, 편집, 검색 기능 제공
    - **:q** : 그대로 종료하기
    - **:q!** : 변경된 내용을 저장하지 않고 종료하기
    - **:wq** : 변경된 내용을 저장하고 종료하기

```
ubuntu@ubuntu-machine: ~
ubuntu@ubuntu-machine: ~ $ ls
Desktop Document Downloads Music Pictures Public Templates Videos
ubuntu@ubuntu-machine: ~ $ cd Downloads/
```

CLI(command Line Interface)



GUI(Graphic User Interface)

# 기기 상호 연결

SSH 연결



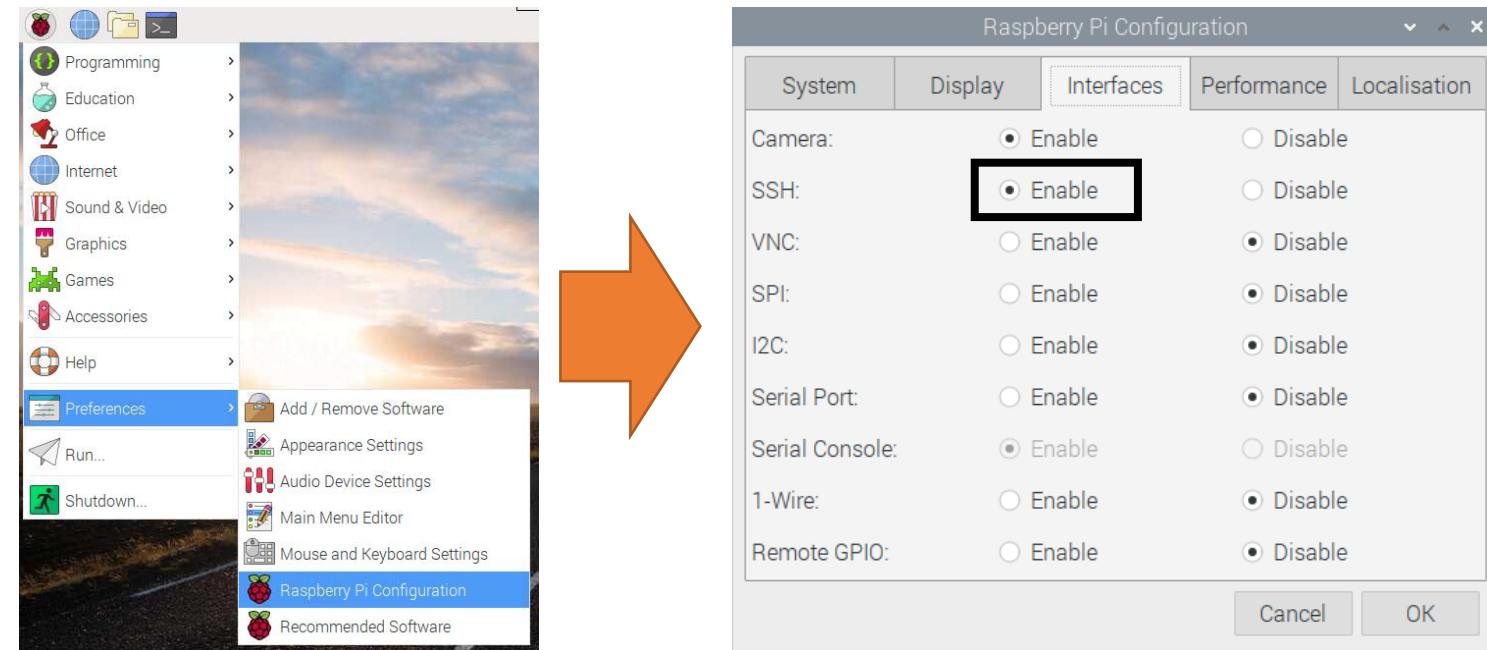
# SSH(Secure Shell Protocol)이란?



Pi에 키보드,  
마우스 연결

- ❖ 서로 다른 기기와의 연결을 위해 사용
- ❖ 컴퓨터 장비들이 인터넷을 통해 데이터를 전송하고, 원격으로 다른 컴퓨터를 제어하기 위해 만들어짐

## ❖ Raspberry Pi SSH 설정



# NUC - Pi SSH 설정



NUC의 터미널에서 실행

- ❖ openssh-server package 설치

```
ubuntu@ubuntu-machine:~$ sudo apt install openssh-server
```

- ❖ NUC에서 SSH를 이용하여 Pi에 접속

```
ubuntu@ubuntu-machine:~$ ssh {raspberry PI username}@{raspberry PI IP address}
```

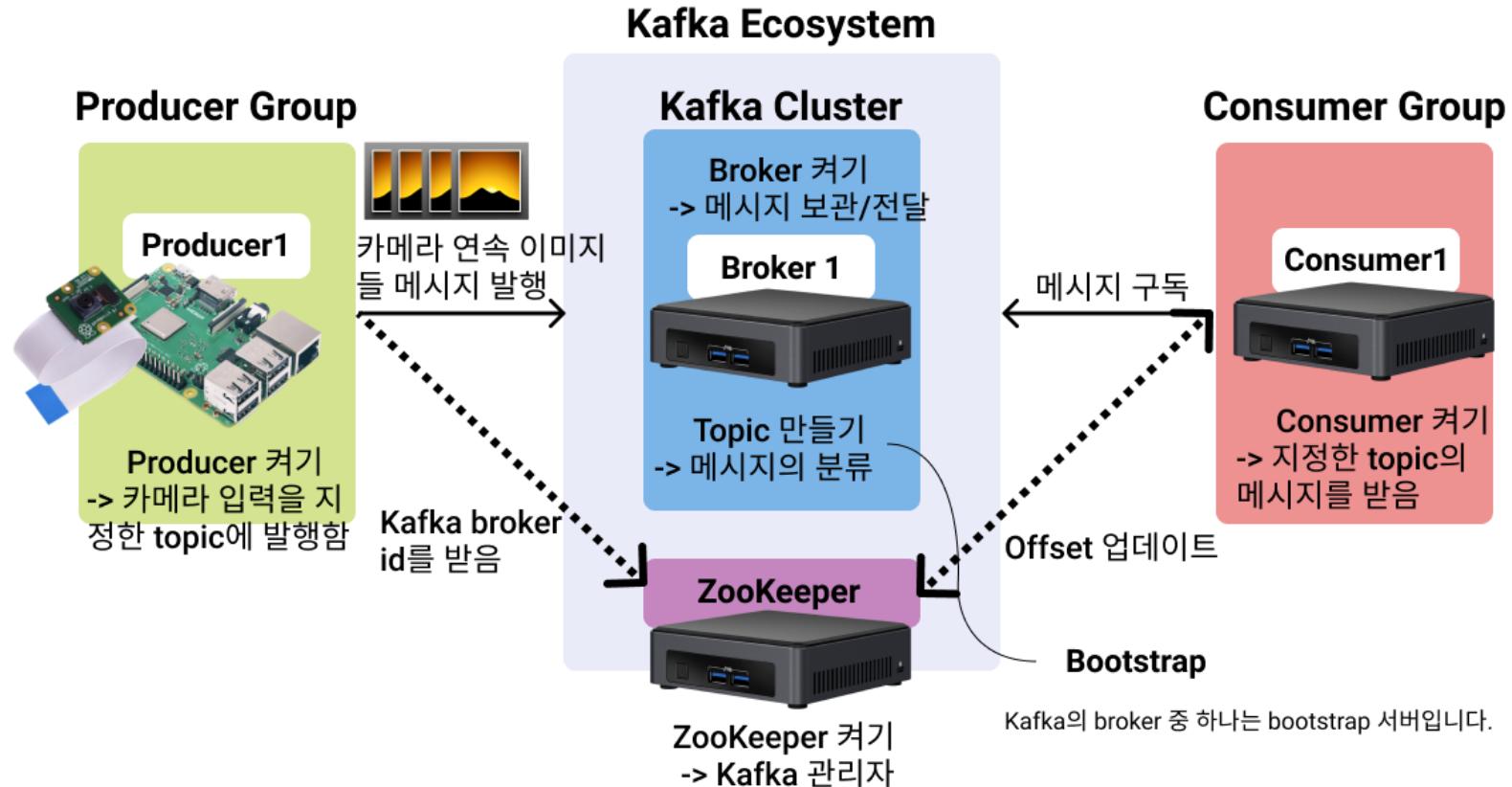
➤ Example

```
ubuntu@ubuntu-machine:~$ ssh pi@172.29.1.13
```

➤ pi로 login 되었으면 성공!

# Kafka 실행

Kafka를 설치하고 실행, 콘솔창의 메시지를 전송



# Experiments requirements

- ❖ Git을 이용하여 실험에 필요한 파일 다운로드(Pi)

```
pi@raspberrypi:~$ git clone https://github.com/gistbdkim/open_gist.git
```



SSH로 Pi에  
서 실행

- ❖ Git을 이용하여 실험에 필요한 파일 다운로드(NUC)

```
ubuntu@ubuntu-machine:~$ sudo apt install git
ubuntu@ubuntu-machine:~$ git clone https://github.com/gistbdkim/open_gist.git
```



NUC의 터미  
널에서 실행

# Package 및 Kafka 설치(NUC)



NUC의 터미널에서 실행

```
ubuntu@ubuntu-machine:~$ sudo apt install -y vim
ubuntu@ubuntu-machine:~$ sudo apt install -y python3-pip
                                         Python3 라이브러리 설치 도구를 설치
ubuntu@ubuntu-machine:~$ sudo apt install -y default-jdk
                                         Kafka 실행에 필요한 JDK를 설치
ubuntu@ubuntu-machine:~$ sudo pip3 install kafka-python
                                         Kafka를 Python에서 실행하게 해주는 패키지를 설치
ubuntu@ubuntu-machine:~$ sudo su
root@ubuntu-machine:/home/ubuntu# wget http://archive.apache.org/dist/kafka/
2.8.0/kafka_2.12-2.8.0.tgz
root@ubuntu-machine:/home/ubuntu# tar -xzf kafka_2.12-2.8.0.tgz
root@ubuntu-machine:/home/ubuntu# exit
```

# Package 및 Kafka 설치(Pi)



SSH로 Pi에서  
실행

```
pi@raspberrypi:~$ sudo apt install -y default-jdk          Kafka 실행에 필요한 JDK를 설치
pi@raspberrypi:~$ sudo rm /usr/lib/python3.11/EXTERNALLY-MANAGED
pi@raspberrypi:~$ sudo pip3 install kafka-python           Kafka를 Python에서 실행하게 해주는 패키지를 설치
pi@raspberrypi:~$ sudo su
root@raspberrypi:/home/pi# wget http://archive.apache.org/dist/kafka/
2.8.0/kafka_2.12-2.8.0.tgz
root@raspberrypi:/home/pi# tar -xzf kafka_2.12-2.8.0.tgz
root@raspberrypi:/home/pi# exit
```

# Kafka 실행 순서

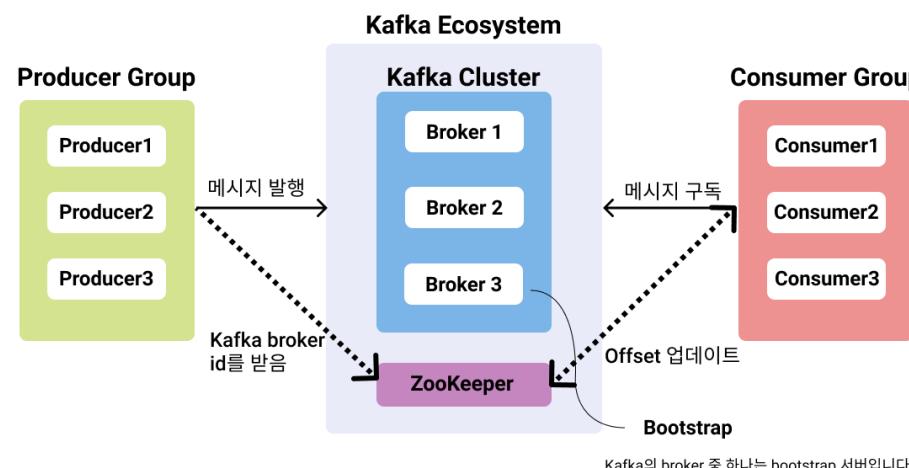
- ❖ Zookeeper 실행(Kafka Server)
- ❖ Server(broker)를 실행 후 Kafka topic을 만듦(Kafka Server)
  - ❖ topic은 이벤트들의 분류이며, 같은 종류=topic의 이벤트들을 함께 묶어서 관리하기 때문에, 이벤트/메시지를 생성하기 전에 topic을 먼저 만들어야 함.
  - ❖ topic은 다시 partition이라는 일정 크기의 조각으로 나뉘어 여러 broker에 복제되어 저장
- ❖ Producer와 Consumer를 실행(Kafka의 Client)
  - ❖ Producer는 이벤트를 생성, Consumer는 이벤트를 받아오는 역할

# Kafka Zookeeper 실행



- ❖ 터미널에서 새로운 탭 open

```
ubuntu@ubuntu-machine: ~  
ubuntu@ubuntu-machine:~$ cd kafka_2.12-2.8.0  
Kafka가 설치된 폴더로 이동  
ubuntu@ubuntu-machine:~/kafka_2.12-2.8.0$ sudo bin/zookeeper-server-start.sh config/zookeeper.properties  
Kafka Zookeeper를 실행
```



- Kafka의 ZooKeeper를 실행해 분산된 메시지 큐의 정보를 관리
- Kafka를 사용하려면 반드시 ZooKeeper를 실행해야 함.

# Kafka Server 실행(1)



- ❖ 터미널에서 새로운 탭 open

```
ubuntu@ubuntu-machine:~$ cd kafka_2.12-2.8.0
```

Kafka가 설치된 폴더로 이동

```
ubuntu@ubuntu-machine:~/kafka_2.12-2.8.0$ sudo vim config/server.properties
```

Kafka server 설정을 수정

- ❖ 아래와 같은 부분을 찾아서 `# advertised.listeners=PLAINTEXT://` 부분을 아래와 같이 `#`을 제거하고 NUC의 IP로 수정

```
# Hostname and port the broker will advertise to producers and consumers. If not
set,
# it uses the value for "listeners" if configured, Otherwise, it will use the
value
# returned from java.net.InetAddress.getCanonicalHostName().
advertiesd.listeners=PLAINTEXT://<NUC IP>:9092
```

# Kafka Server 실행 및 Topic 생성



NUC의 터미널에서 실행

- ❖ 이후 서버를 실행

```
ubuntu@ubuntu-machine:~/kafka_2.12-2.8.0$ sudo bin/kafka-server-start.sh  
config/server.properties
```

- ❖ Kafka의 Server(broker)를 실행하여 클라이언트(producer, consumer)로부터의 발행, 구독, 정보 요청을 처리하고, 데이터를 topic, partition으로 나누어 복제하고 저장
- ❖ server.properties에서는 broker의 주소를 수정해 NUC 외부 기기인 Pi에서도 접근이 가능하도록 처리

- ❖ Kafka Topic 생성

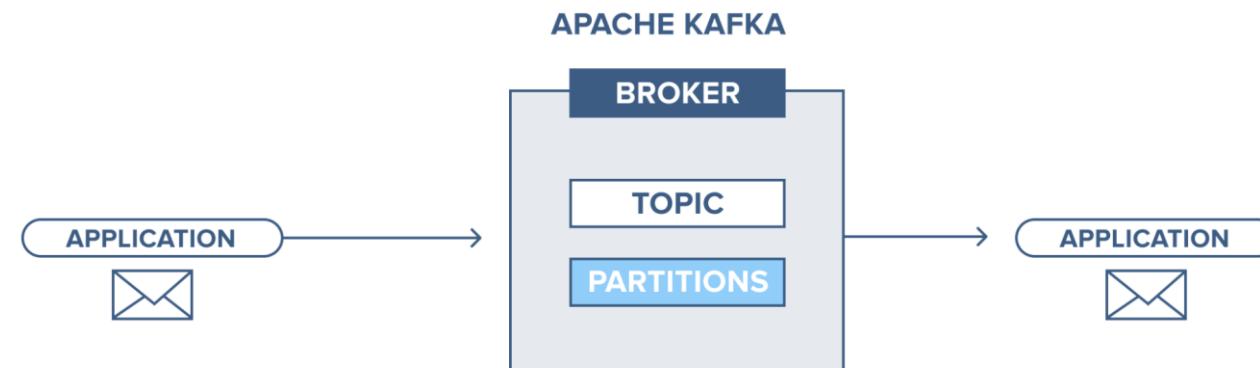
- ❖ Kafka는 메시지를 topic에 따라 관리하며, 메시지를 발행하는 producer도 특정 topic에 대해 메시지를 발행하고, consumer도 특정 topic을 지정해 그 topic에 대한 메시지들을 구독
- ❖ Topic은 partition으로 나뉘어 broker들에 분산, 복제되어 저장

# Kafka Topic 생성



- ❖ 터미널에서 새로운 탭 open 후 아래의 <NUC IP> 부분에 본인의 IP를 입력하고 진행

```
ubuntu@ubuntu-machine:~/kafka_2.12-2.8.0$ sudo bin/kafka-topics.sh --create --  
bootstrap-server <NUC IP>:9092 --replication-factor 1 --partitions 1 --topic chat  
--create 는 새로운 topic 생성, --topic으로 토픽의 이름을 진행  
--replication-factor는 메시지의 복제 개수를  
--partitions는 메시지를 몇 개의 broker로 쪼개어 저장할 것인지 지정  
ubuntu@ubuntu-machine: ~/kafka_2.12-2.8.0$ sudo bin/kafka-topics.sh --list --  
bootstrap-server <NUC IP>:9092  
chat  
--list 옵션으로 만들어진 topic의 목록을 확인할 수 있음  
방금 만든 chat을 확인할 수 있음
```



# Pi 콘솔창 입력 메시지 발행, NUC 구독

- ❖ 터미널에서 새로운 탭 open 후 아래의 <NUC IP> 부분에 본인의 IP를 입력하고 진행

```
pi@raspberrypi:~$ cd kafka_2.12-2.8.0  
pi@raspberrypi:~/kafka_2.12-2.8.0$ sudo  
bin/kafka-console-producer.sh  
--broker-list <NUC IP>:9092 --topic chat  
> hi from pi  
> 어떤 메시지를 입력해 보세요.
```

메시지를 만드는 프로듀서를 실행하면, 터미널  
에 입력된 내용은 브로커로 전달

SSH로 Pi에  
서 실행



```
ubuntu@ubuntu-machine:~$ cd  
kafka_2.12-2.8.0  
ubuntu@ubuntu-machine:~/kafka_2.12-  
2.8.0$ sudo bin/kafka-console-  
consumer.sh --bootstrap-server <NUC  
IP>:9092 --topic chat --from-beginning  
hi from pi  
어떤 메시지를 입력해보세요.
```

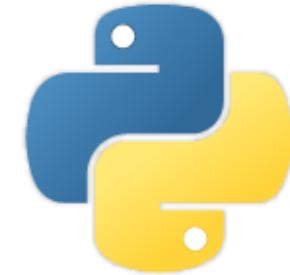
브로커에 전달된 메시지는 컨슈머에서 확인할  
수 있음

NUC의 터미널  
에서 실행



# Python Kafka 사용

Python의 Kafka 라이브러리를 활용해 Pi의 콘솔창 입력을 NUC으로 전송



# Pi, NUC에서 서버 주소 전역 환경변수 설정

- ❖ VIM으로 ~/.bashrc 파일 수정

```
ubuntu@ubuntu-machine:~$ vim ~/.bashrc
```

- ❖ VIM에서 G를 누르면 파일의 맨 끝으로 이동, 아래와 같이 SERVER\_IP에 NUC의 IP를 입력

```
export SERVER_IP=<NUC IP>
```

- ❖ 저장 후, 아래 명령어를 입력하여 환경 변수를 적용

```
ubuntu@ubuntu-machine:~$ source ~/.bashrc
```

환경변수란, 셸 세션과 작업 환경에 대한 정보를 저장하는 변수  
쉘에서 실행중인 모든 프로그램이나 스크립트가 이 변수에 접근할 수 있음



**NUC와 PI 양쪽에서 다 실행!!**

# Python Kafka Producer



SSH로 Pi에서  
실행

- ❖ 위에서 만들어 놓은 Zookeeper와 Broker가 NUC에 켜져 있는 상태에서 진행해야 함
- ❖ 아래 폴더로 이동

```
pi@raspberrypi:~$ cd ~/open_gist/kafka_transmission
```

- ❖ 이후 아래 커맨드로 producer를 실행하고, 터미널에서 메시지를 입력

```
pi@raspberrypi:~/open_gist/kafka_transmission$ python3
console_producer.py
메시지를 입력하세요: 안녕하세요 아무말이나 입력하세요
메시지를 입력하세요: 하이
```

- ❖ 파일 구성

1. 필요한 라이브러리들을 호출
2. 환경변수(SERVER\_IP)에 저장된 Broker 서버 주소를 불러옴
3. Python 실행 시 topic을 인자로 받고 Kafka producer를 정의
4. Ctrl + C를 눌러 종료하기 전까지, 입력 값을 받아 Kafka 메시지로 발행

# Python Kafka Consumer



NUC의 터미널에서 실행

- ❖ 마찬가지로 clone을 받고, 폴더로 이동
- ❖ consumer를 실행하고 발행된 메시지를 확인

```
ubuntu@ubuntu-machine:~$ cd ~/open_gist/kafka_transmission  
ubuntu@ubuntu-machine:~/open_gist/kafka_transmission$ python3  
console_consumer.py
```

도착한 메시지: {'1' : '안녕하세요 아무 말이나 입력하세요' }

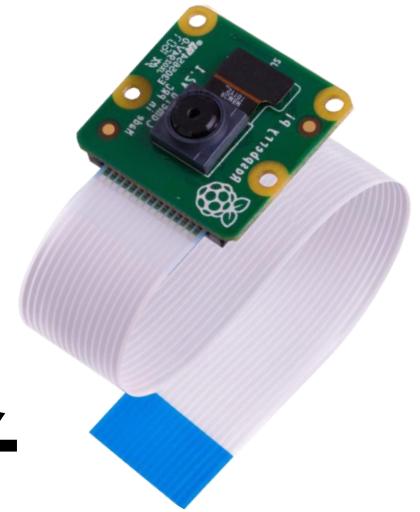
- ❖ 파일 구성

1. 필요한 라이브러리들을 호출
2. 환경변수(SERVER\_IP)에 저장된 Broker 서버 주소를 불러옴
3. Python 실행 시 topic을 인자로 받고, Kafka consumer를 정의
4. Ctrl + C를 눌러 종료하기 전까지, Broker에서 메시지를 받아오고 출력

# Pi → NUC 카메라 영상 전송

Pi에서 NUC으로 카메라 영상을 전송

Kafka, kafka-python, OpenCV를 사용



# Kafka Setup



- ❖ 위에서 만들어 놓은 Zookeeper와 Broker가 NUC에 켜져 있는 상태에서 진행
  - ❖ 켜져 있지 않다면 다시 실행
- ❖ 비디오 송수신을 위해 새로운 “pi-video”라는 새로운 Kafka topic을 생성

```
ubuntu@ubuntu-machine:~/kafka_2.12-2.8.0$ sudo bin/kafka-topics.sh --create  
--bootstrap-server <NUC IP>:9092 --replication-factor 1 --partitions 1  
--topic pi-video  
ubuntu@ubuntu-machine:~/kafka_2.12-2.8.0$ sudo bin/kafka-topics.sh --list  
--bootstrap-server <NUC IP>:9092  
chat  
pi-video
```

# OpenCV Python Setup(NUC)



NUC의 터미널에서 실행

- ❖ opencv-python : Python에서 OpenCV 라이브러리를 사용하기 위한 package
- ❖ 필요한 package 설치

```
ubuntu@ubuntu-machine:~$ sudo apt install -y libatlas-base-dev
ubuntu@ubuntu-machine:~$ sudo pip3 install -U pip
ubuntu@ubuntu-machine:~$ sudo pip3 install opencv-python
ubuntu@ubuntu-machine:~$ sudo pip3 install numpy==1.26.4
```

# Python Kafka Video Producer



- ❖ ~/open\_gist/kafka\_transmission 디렉토리로 이동
- ❖ 아래 명령어를 실행하여 Pi Camera를 통해 받아온 데이터를 OpenCV로 읽어온 후 비디오 프레임을 메시지로 발행

```
pi@raspberrypi:~$ cd ~/open_gist/kafka_transmission  
pi@raspberrypi:~/open_gist/kafka_transmission$ python3 video_producer.py --topic pi-video
```

- ❖ 파일 구성
  1. 필요한 라이브러리 호출
  2. 환경변수(SERVER\_IP)에 저장된 Broker 서버 주소를 불러옴
  3. Python 실행 시 topic을 인자로 받음
  4. Kafka produce를 정의
  5. Ctrl + C를 눌러 종료하기 전까지, 비디오 데이터를 읽어와서 Kafka 메시지로 발행

# Python Kafka Video Consumer



NUC의 터미널에서 실행

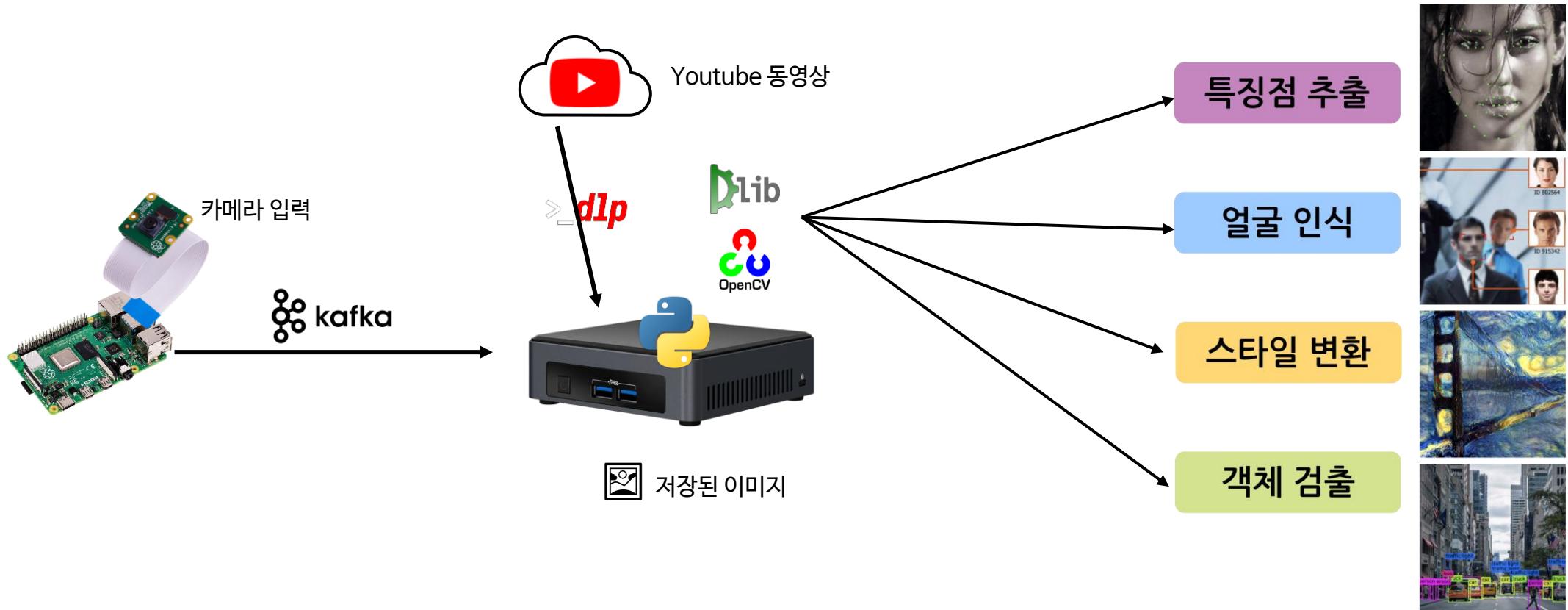
- ❖ ~/AI\_SUMMER\_2002/kafka\_transmission 디렉토리로 이동
- ❖ 아래 명령어를 실행하여 발행된 메시지(비디오 프레임)을 소비하고 확인 할 수 있음

```
ubuntu@ubuntu-machine:~$ cd ~/open_gist/kafka_transmission  
ubuntu@ubuntu-machine:~/open_gist/kafka_transmission$ python3 video_consumer.py
```

- ❖ 파일 구성

1. 필요한 라이브러리 호출
2. 환경변수(SERVER\_IP)에 저장된 Broker 서버 주소를 불러옴
3. Python 실행 시 topic을 인자로 받음
4. Kafka consumer를 정의
5. Ctrl + C를 눌러 종료하기 전까지, Broker에서 메시지(비디오 프레임)를 받아오고 이를 확인

# Computer Vision 기술을 이용한 영상 처리



# 얼굴 특징점 추출(Facial Landmark Detection)



NUC의 터미널에서 실행

- ❖ Python package 설치, 모델 다운로드
- ❖ ~/open\_gist/face\_landmark 디렉토리에서 진행

```
ubuntu@ubuntu-machine:~$ cd ~/open_gist/face_landmark
ubuntu@ubuntu-machine:~/open_gist/face_landmark$ sudo apt install cmake libgtk-3-dev
libboost-all-dev
ubuntu@ubuntu-machine:~/open_gist/face_landmark$ sudo pip3 install imutils yt-dlp
ubuntu@ubuntu-machine:~/open_gist/face_landmark$ sudo pip3 install dlib==19.22.0 -vvv
ubuntu@ubuntu-machine:~/open_gist/face_landmark$ wget
http://dlib.net/files/shape_predictor_68_face_landmarks.dat.bz2
ubuntu@ubuntu-machine:~/open_gist/face_landmark$ bunzip2
shape_predictor_68_face_landmarks.dat.bz2
```

필요한 모델을 받고 압축을 해제

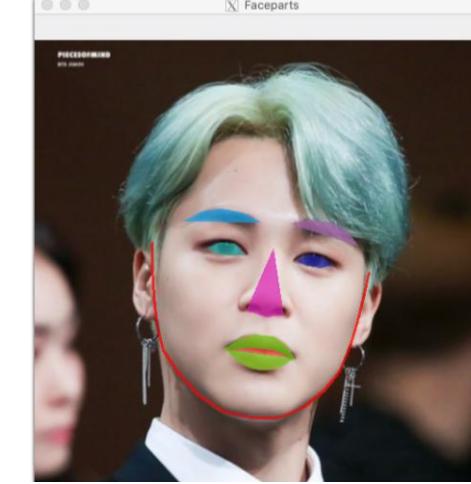
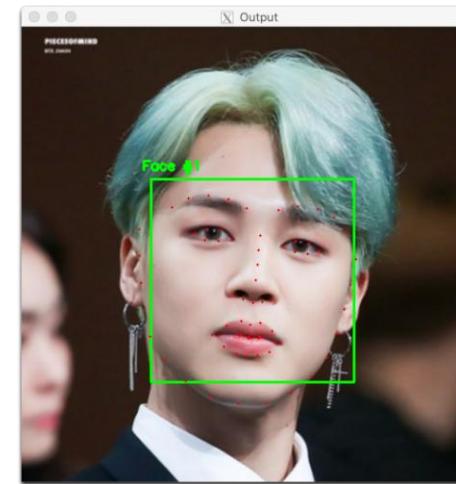
NUC의 터미널에서 실행



# 얼굴 특징점 추출(Facial Landmark Detection)

- ❖ 이미지로부터 얼굴 특징점 추출
  - ❖ 저장되어 있는 이미지에서 얼굴 특징점을 추출

```
ubuntu@ubuntu-machine:~/open_gist/face_landmark$ python3 img_face_landmark.py --img_path {원하는 이미지 경로} --show_parts {True 또는 False}  
--img_path 인자로 특징점을 검출할 이미지 경로를 입력할 수 있음  
기본값은 ../dataset/single_face/jimin.jpg  
--show_parts 인자에 True를 주면 얼굴의 주요 부위를 확인해 볼 수 있음  
기본 값은 False
```



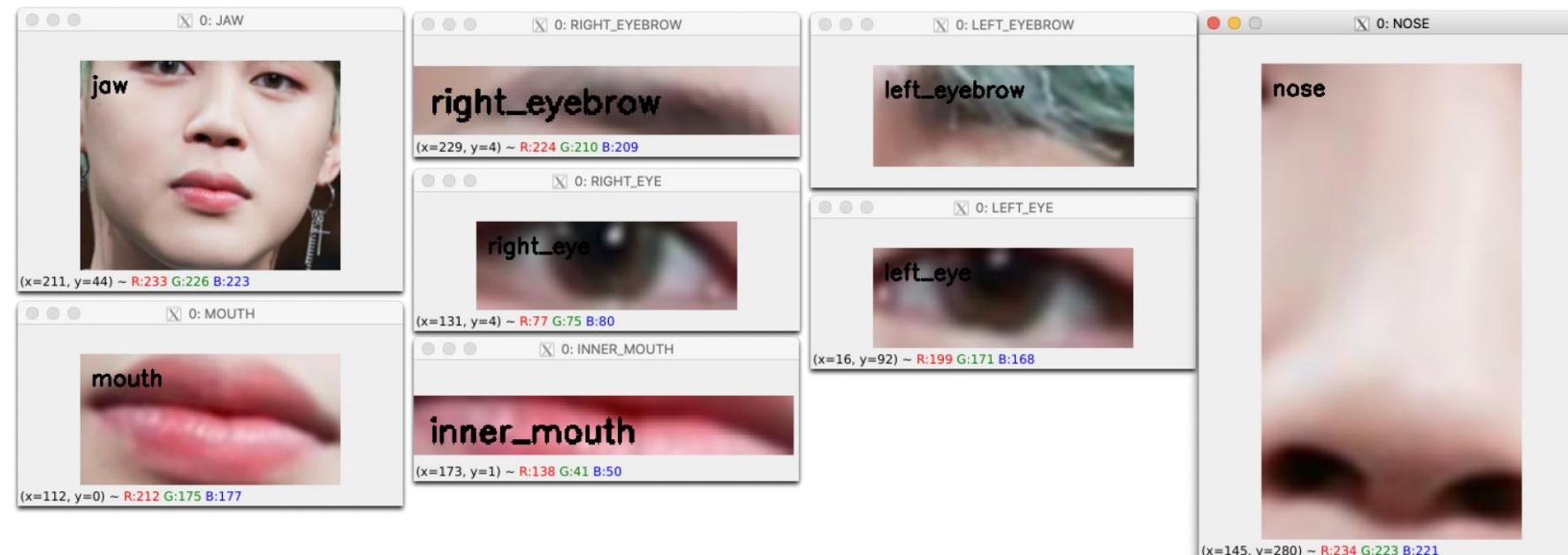
# 얼굴 특징점 추출(Facial Landmark Detection)



- ❖ 이미지로부터 얼굴 주요 부위 추출

- ❖ 입, 입라인, 왼쪽/오른쪽 눈썹, 왼쪽/오른쪽 눈, 코, 턱 등 8가지 주요 부분의 이미지를 확인 할 수 있음

```
ubuntu@ubuntu-machine:~/open_gist/face_landmark$ python3 img_face_landmark.py --  
show_parts True
```



# 얼굴 특징점 추출(Facial Landmark Detection)

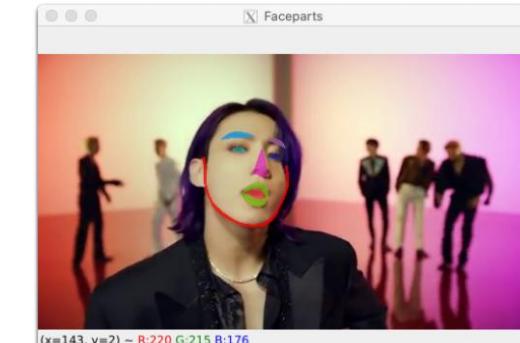
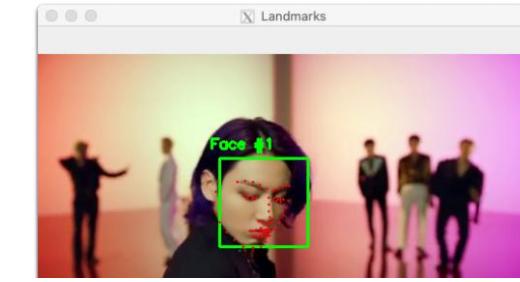


- ❖ Youtube 영상에서 얼굴 특징점 추출
  - ❖Youtube 영상을 저장하고, 얼굴 특징점을 추출

```
ubuntu@ubuntu-machine:~/open_gist/face_landmark$ python3  
youtube_face_landmark.py --youtube_url {Youtube 영상 링크}
```

--youtube\_url 인자로 원하는 영상 선택 가능  
생략하면 미리 설정된 기본값 실행

- ❖ 동영상 인식
  - ❖ 동영상은 여러 장의 정지된 사진이 순서대로 재생되면서 화면 속에서 실제로 움직이는 것처럼 보이는 것
  - ❖ 동영상을 이루는 한 장을 프레임(frame)이라고 하며, 동영상의 각 프레임마다 이미지에 적용할 때와 같은 함수를 적용하여 동영상에 컴퓨터 비전을 적용할 수 있음



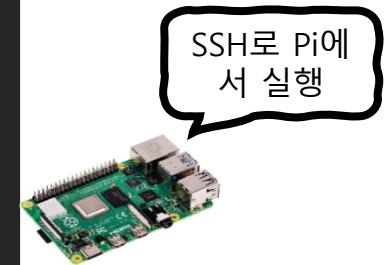
# 얼굴 특징점 추출(Facial Landmark Detection)

- ❖ Pi Camera 영상에서 얼굴 특징점 추출

- ❖ 위에서 만들어 놓은 Zookeeper와 Broker가 NUC에 켜져 있는 상태에서 진행

```
pi@raspberrypi:~$ cd ~/open_gist/kafka_transmission  
pi@raspberrypi:~/open_gist/kafka_transmission$ python3  
video_producer.py --topic {Kafka Topic}
```

--topic 인자로 발행할 Kafka topic을 설정  
생략할 시 이전에 만든 "pi-video" 토픽을 사용



- ❖ NUC에서 video를 받아 특징점 추출

```
ubuntu@ubuntu-machine:~/open_gist/face_landmark$ python3  
video_face_landmark.py --topic {Kafka Topic}
```

--topic 인자로 발행할 Kafka topic을 설정  
생략할 시 전에 만든 "pi-video" 토픽을 사용



# 얼굴 인식(Face Recognition)



NUC의 터미널에서 실행

- ❖ Python package 설치, 모델 다운로드
- ❖ ~/open\_gist/face\_recognition 디렉토리에서 진행

```
ubuntu@ubuntu-machine:~/open_gist/face_landmark$ cd ~/open_gist/face_recognition  
ubuntu@ubuntu-machine:~/open_gist/face_recognition$ sudo pip3 install face-recognition==1.3.0
```

# 얼굴 인식(Face Recognition)



SSH로 Pi에서  
실행

- ❖ 얼굴 등록하기 : Pi Camera 동작시키기
- ❖ 위에서 만들어 놓은 Zookeeper와 Broker가 NUC에 켜져 있는 상태에서 진행

```
pi@raspberrypi: ~ $ cd ~/open_gist/kafka_transmission
pi@raspberrypi: ~/open_gist/kafka_transmission $ 
python3 video_producer.py --topic {Kafka Topic}
# --topic 인자로 발행할 Kafka topic을 설정합니다.
# 생략할 시 전에 만든 "pi-video" 토픽을 씁니다.
```

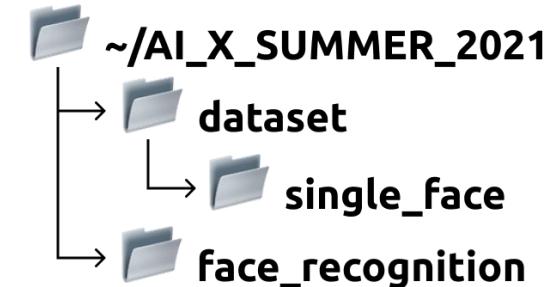
# 얼굴 인식(Face Recognition)

NUC의 터미널에서 실행



## 얼굴 등록하기

```
ubuntu@ubuntu-machine:~/open_gist/face_recognition$ python3 save_photo.py  
사진 저장을 시작합니다  
이름을 입력한 뒤, 키를 누르시면 사진이 저장됩니다. 종료하시려면 사람의 이름 대신  
'c'를 입력해주세요  
사람의 이름을 영어로 입력해주세요: hi  
./dataset/single_face/hi.jpg에 이미지가 저장되었습니다  
사람의 이름을 영어로 입력해주세요:
```



- ❖ save\_photo.py를 사용하여 한 명씩 Pi 카메라로 사진을 찍고(한 명의 얼굴만 나오도록), 이를 open\_gist/dataset/single\_face에 저장
  - ⚠️ 사람 이름은 영어로 입력
    - ❖ --save\_path 인자를 사용하여 사진 저장 위치를 바꿀 수 있음
  - ❖ 혹은 위 경로에 원하는 사람 혼자 나온 사진을 저장해도 됨
  - ❖ 해당 경로에 저장된 사람의 얼굴이 등록되고, 각 파일의 이름이 사람의 이름으로 등록

# 얼굴 인식(Face Recognition)

NUC의 터미널에서 실행



## ❖ 이미지 얼굴 인식

```
ubuntu@ubuntu-machine:~/open_gist/face_recognition$ python3  
img_face_recognition.py --img_path {데이터셋 경로} --threshold {기준점}  
--img_path 인자로 얼굴 인식을 진행할 이미지를 선택  
기본값은 ../dataset/multiple_face/bts2.jpg  
--threshold 인자의 기본값은 0.6
```

## ❖ Threshold란?

- ❖ 등록된 사람과의 비교로 그 사람이 맞다고 판별할 수 있는 차의 기준
- ❖ 여기서는 얼굴에 대한 인코딩 값의 차를 기준으로 판별
- ❖ threshold 값이 커질수록 더욱 관용적으로 판별해 등록되지 않은 사람을 등록된 사람으로 오판할 수 있음
- ❖ 작아질수록 등록된 사람도 Unknown으로 오판하게 될 수 있음
- ❖ 값을 조정하여 변화를 확인



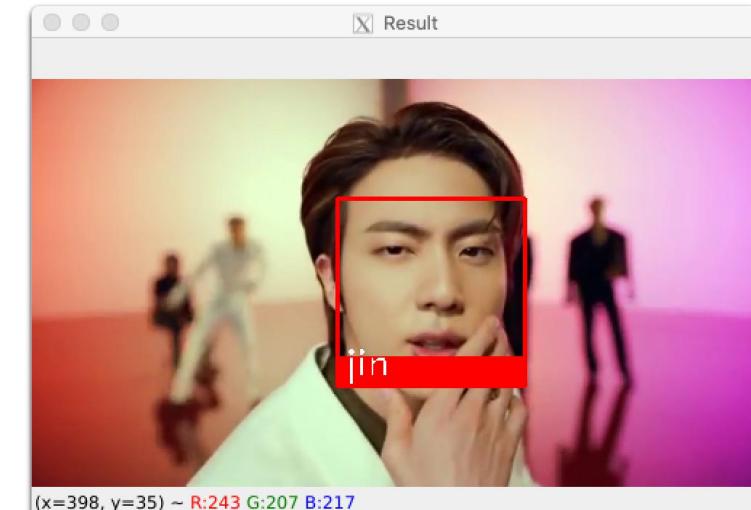
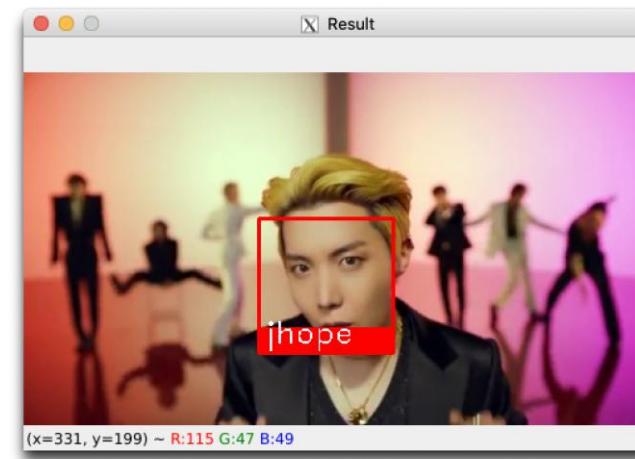
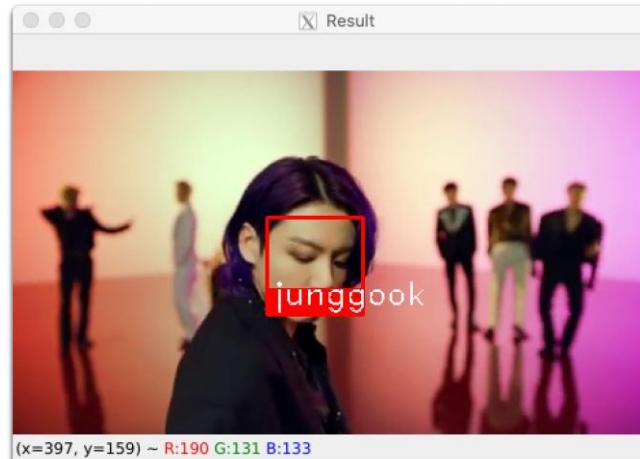
# 얼굴 인식(Face Recognition)

NUC의 터미널에서 실행



## ❖ Youtube 영상 얼굴 인식

```
ubuntu@ubuntu-machine:~/open_gist/face_recognition$ python3  
youtube_face_recognition.py --img_path {데이터셋 경로} --threshold {기준점}  
--youtube_url 인자로 원하는 youtube 영상 선택 가능  
--threshold 인자의 기본값은 0.6
```



# 얼굴 인식(Face Recognition)

- ❖ Pi Camera 영상에서 얼굴 인식
  - ❖ 위에서 만들어 놓은 Zookeeper와 Broker가 NUC에 켜져 있는 상태에서 진행

```
pi@raspberrypi:~$ cd ~/open_gist/kafka_transmission  
pi@raspberrypi:~/open_gist/kafka_transmission$ python3  
video_producer.py --topic {Kafka Topic}
```

--topic 인자로 발행할 Kafka topic을 설정  
생략할 시 전에 만든 "pi-video" 토픽을 사용



SSH로 Pi에  
서 실행

- ❖ NUC에서 비디오를 받아 얼굴 인식 진행

```
ubuntu@ubuntu-machine:~/open_gist/face_recognition$ python3  
video_face_recognition.py --topic {Kafka Topic} --threshold {기준점}
```

--topic 인자로 발행할 Kafka topic을 설정  
생략할 시 전에 만든 "pi-video" 토픽을 사용  
--threshold 옵션으로 사람의 식별 기준을 0 ~ 1 사이로 설정



NUC의 터미  
널에서 실행

# 객체 검출(Object Detection)



- ❖ Python package 설치, 모델 다운로드
  - ❖ GPU가 없는 제약된 상황이므로, Yolo v3 "Tiny" 모델을 사용
  - ❖ 추론의 정확도는 떨어지지만, 추론 속도가 크게 지연되지 않음
  - ❖ <https://pjreddie.com/media/files/yolov3-tiny.weights>

```
ubuntu@ubuntu-machine: cd ~/open_gist/object_detection
```

# 객체 검출(Object Detection)

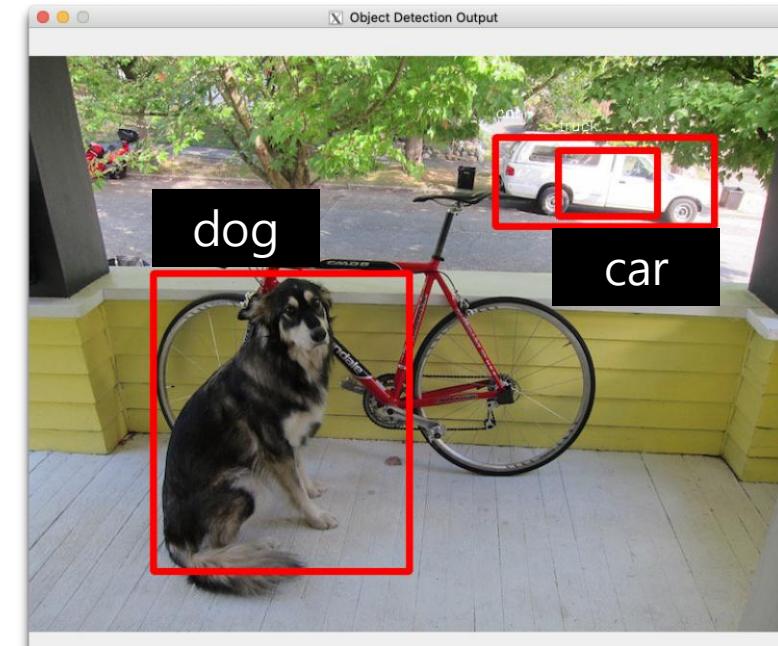


- ❖ 저장된 이미지로부터 객체 검출

```
ubuntu@ubuntu-machine:~/open_gist/object_detection$ python3
```

```
img_object_detection.py --img_path {원하는 이미지 경로}
```

--img\_path 인자로 특징점을 검출할 이미지 경로를 입력 가능  
기본값은 ../dataset/objects/dog\_bicycle.jpg

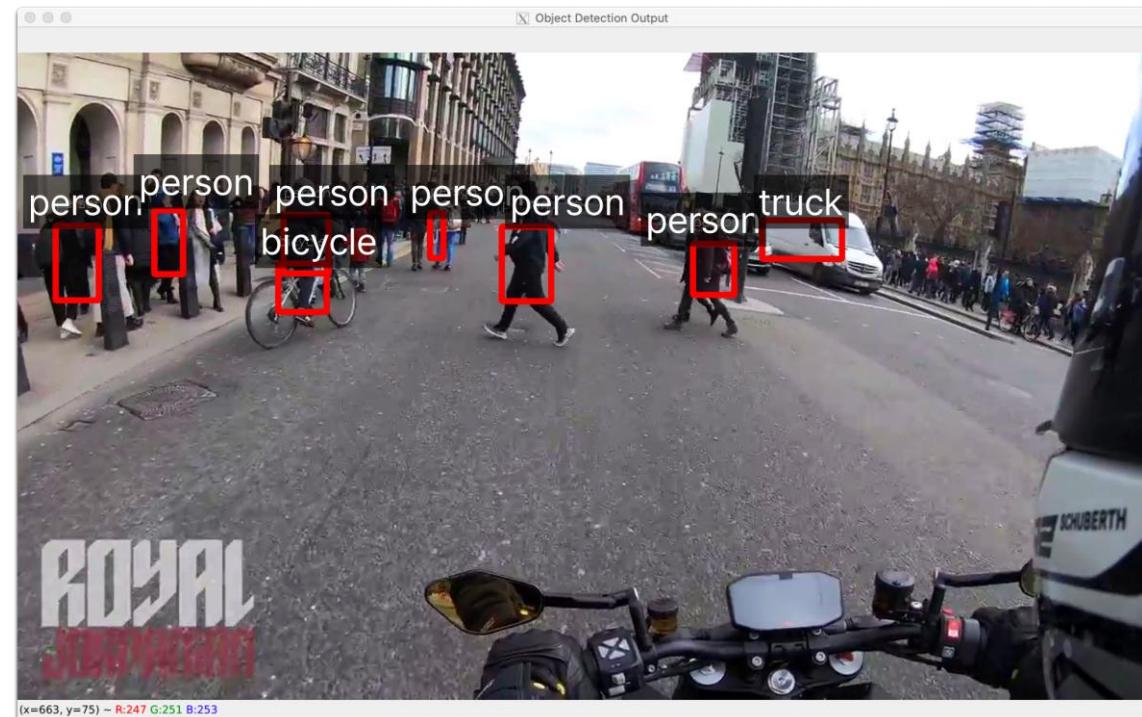


# 객체 검출(Object Detection)



- ❖ Youtube 영상에서 객체 검출

```
ubuntu@ubuntu-machine:~/open_gist/object_detection$ python3  
youtube_object_detection.py --youtube_url {원하는 영상 경로}  
--youtube_url 인자로 원하는 youtube 영상 선택
```



# 객체 검출(Object Detection)

- ❖ Pi Camera 영상에서 객체 검출

- ❖ 위에서 만들어 놓은 Zookeeper와 Broker가 NUC에 켜져 있는 상태에서 진행

```
pi@raspberrypi:~$ cd ~/open_gist/kafka_transmission  
pi@raspberrypi:~/open_gist/kafka_transmission$ python3  
video_producer.py --topic {Kafka Topic}
```

--topic 인자로 발행할 Kafka topic을 설정  
생략할 시 전에 만든 "pi-video" 토픽을 사용



- ❖ NUC에서 비디오를 받아 객체 검출

```
ubuntu@ubuntu-machine:~/open_gist/object_detection$ python3  
video_object_detection.py --topic {Kafka Topic}
```

--topic 인자로 발행할 Kafka topic을 설정



# 스타일 변환(Style Transfer)



- ❖ Python package 설치, 모델 다운로드
  - ❖ 미리 스타일이 학습된 모델들을 다운받아 사용
  - ❖ <https://www.dropbox.com/sh/2z3hyrewinnmubf/AACUAazQxfKpiMBzjHUVXFRDa> --content-disposition

```
ubuntu@ubuntu-machine:$ cd ~/open_gist/style_transfer
ubuntu@ubuntu-machine:~/open_gist/style_transfer$ wget
https://www.dropbox.com/sh/2z3hyrewinnmubf/AACUAazQxfKpiMBzjHUVXFRDa --content-dis
position
ubuntu@ubuntu-machine:~/open_gist/style_transfer$ mkdir models
ubuntu@ubuntu-machine:~/open_gist/style_transfer$ 
unzip models.zip -d ./models
```

NUC의 터미널에서 실행



# 스타일 변환(Style Transfer)

## ❖ 이미지 스타일 변환

```
ubuntu@ubuntu-machine:~/open_gist/style_transfer$ python3 img_style_transfer.py -  
-img_path {원하는 이미지 경로} --style_path {원하는 스타일 학습된 모델 경로} --  
img_path 옵션으로 스타일 변환할 이미지 경로를 변경 가능  
기본적으로 ../dataset/objects/dog_bicycle.jpg  
--style_path 옵션으로 적용할 스타일이 학습된 모델을 선택할 수 있음  
기본적으로 ./models/mosaic.onnx
```



wave\*.onnx



starry\*.onnx



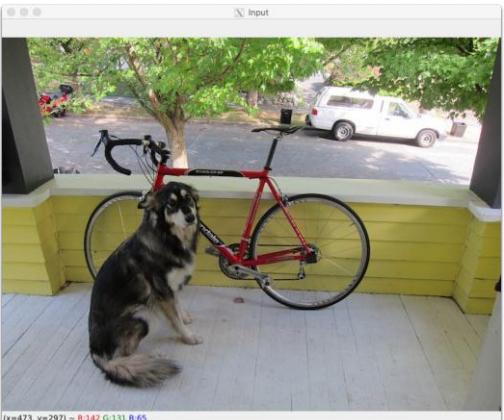
mosaic\*.onnx



udnei\*.onnx

# 스타일 변환(Style Transfer)

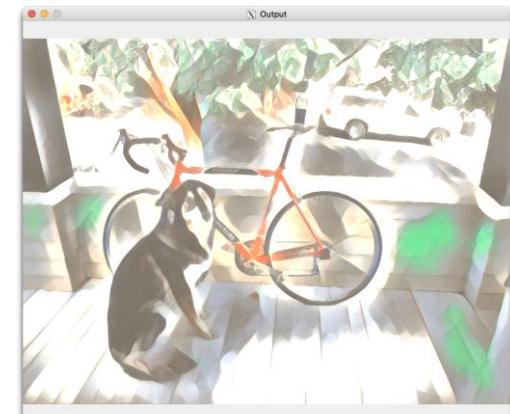
- ❖ 저장된 이미지 스타일 변환 결과



원본 이미지



wave150.onnx



udnie\_aggressive.onnx



mosaic.onnx



starry150.onnx

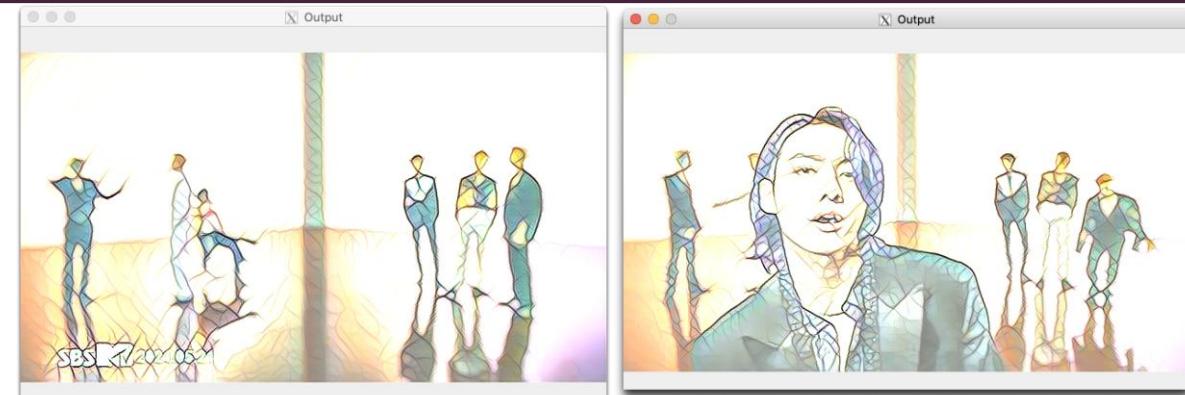
# 스타일 변환(Style Transfer)



## ❖ Youtube 영상 스타일 변환

```
ubuntu@ubuntu-machine:~/open_gist/style_transfer$ python3
youtube_style_transfer.py --youtube_url {원하는 영상 링크} --style_path {원하는
스타일 모델} --skip_ratio {비디오 프레임 스kip 수}
--youtube_url 인자로 원하는 youtube 영상 선택 가능
--style_path 옵션으로 적용할 스타일이 학습된 모델을 선택 가능
기본적으로 ./models/mosaic.onnx
--skip_ratio 옵션으로 몇 개의 프레임을 건너 뛰며 추론할 것인지 선택
모든 프레임을 변환하면 매우 느리므로, 일부를 스kip
```

mosic.onnx의 적용 예시



# 스타일 변환(Style Transfer)

- ❖ Pi Camera 영상에서 스타일 변환

- ❖ 위에서 만들어 놓은 Zookeeper와 Broker가 NUC에 켜져 있는 상태에서 진행

```
pi@raspberrypi:~$ cd ~/open_gist/kafka_transmission  
pi@raspberrypi:~/open_gist/kafka_transmission$ python3  
video_producer.py --topic {Kafka Topic}
```

--topic 인자로 발행할 Kafka topic을 설정  
생략할 시 전에 만든 "pi-video" 토픽을 사용



SSH로 Pi에  
서 실행

- ❖ NUC에서 비디오를 받아 스타일 변환 진행

```
ubuntu@ubuntu-machine:~/open_gist/style_transfer$ python3  
video_style_transfer.py --topic {Kafka Topic} --style_path {원하는 스타  
일 모델} --skip_ratio {비디오 프레임 스킵 수}
```

--topic 인자로 발행할 Kafka topic을 설정  
--style\_path 옵션으로 적용할 스타일이 학습된 모델을 선택할 수 있음  
--skip\_ratio 옵션으로 몇 개의 프레임을 건너 뛰며 추론할 것인지 선택



NUC의 터미  
널에서 실행