한국IT교육원

졸음운전 방지 예방을 위한 인공지능 기반 차량 내 전방주시 감지 여부

팀명 : 자율주행

팀원 : 조 수익, 김 덕열

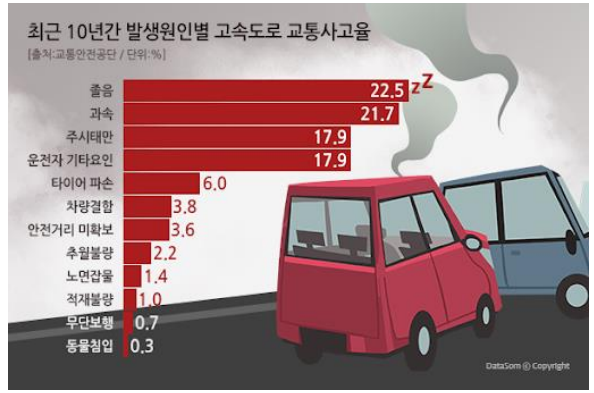
목차

01 프로젝트 개요

- 프로젝트 주제 및 선정 배경
  - 졸음운전 방지 예방을 위한 전방 주시 감지 여부
  - 졸음운전 경고 장치인 'DSM(Driver Status Monitoring) 시스템 개발

- 프로젝트 개발 환경
  - OpenCV : V 4.5.3
  - Python : V 3.8.10
  - Keras : V 2.4.3
  - Tensorflow : V 2.5.0

- 프로젝트 단계
  - 졸음 운전자 상태 이미지 수집
  - 수집한 이미지를 통한 학습
  - 결과 도출 및 적용



- 기대 효과
  - 전방을 주시하는지 확인하여 사고예방 가능
  - 어두운 환경, 안경 착용, 마스크, 선글라스 등 DSM 인식률 향상

02 프로젝트 팀 구성 및 역할

팀원 별 역할 및 담당 업무

훈련생	역할	담당 업무
조수익	팀장	▶ 알고리즘 개발 및 구현 ▶ 데이터 전처리 및 최적화 ▶ PPT 제작
김덕열	팀원	▶ 졸음운전 예방 데이터 이미지 수집 및 분류(전방주시 / 비 전방주시) ▶ PPT 제작 ▶ 발표

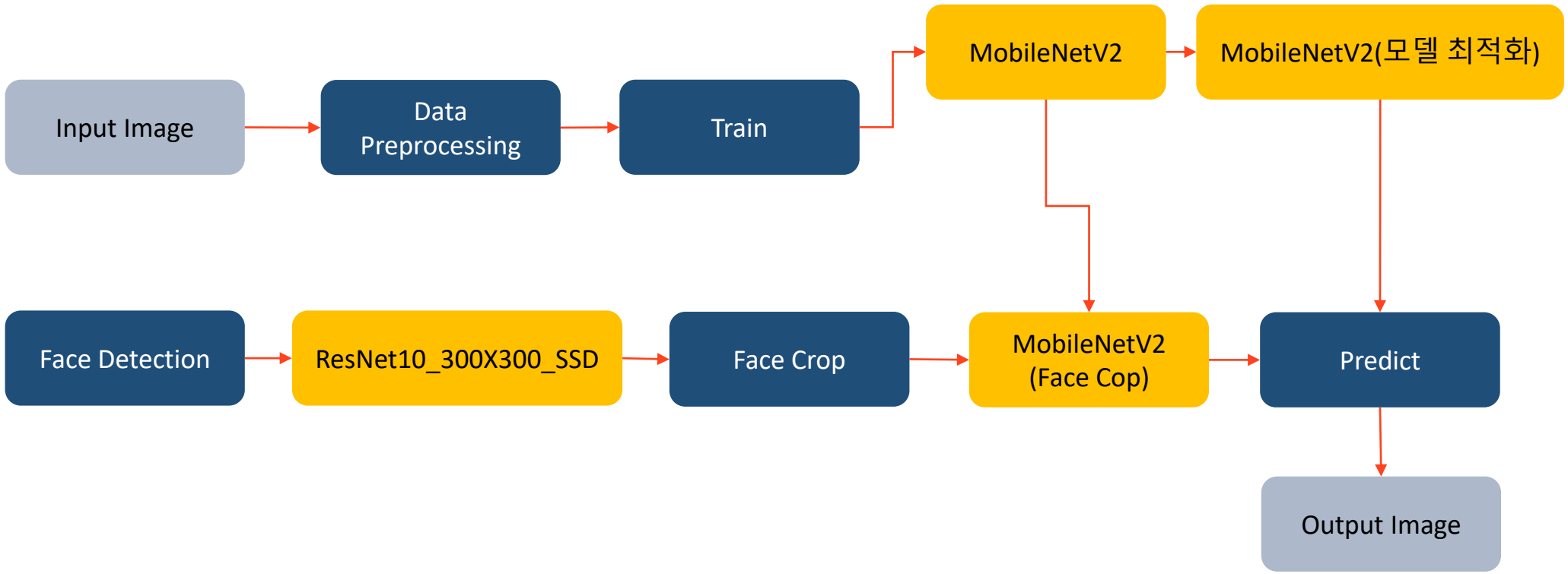
03 프로젝트 수행 절차 및 방법

프로젝트 기간 및 구체적 활동

구분	기간	활동	비고
사전 기획	11/1(월)~11/1(월)	프로젝트 기획 및 주제 선정 코드 구조 설계 (데이터셋, 알고리즘)	아이디어 선정
데이터 수집	11/2(화)~11/2(화)	필요 데이터 및 수집 절차 정의 데이터셋 확보	AI-Hub 운전자 상태정보 이미지 수집
코드 구현	11/3(수)~11/3(수)	데이터셋 전처리 모델 학습	최적화, 오류 수정
코드 함수화	11/4(목)~11/4(목)	인식률 개선 및 시연 ppt 제작	
발표 자료 정리	11/5(금)~11/5(금)	최종 발표	발표
총 개발기간	11/1(월)~11/5(금) (총 3주)	-	-

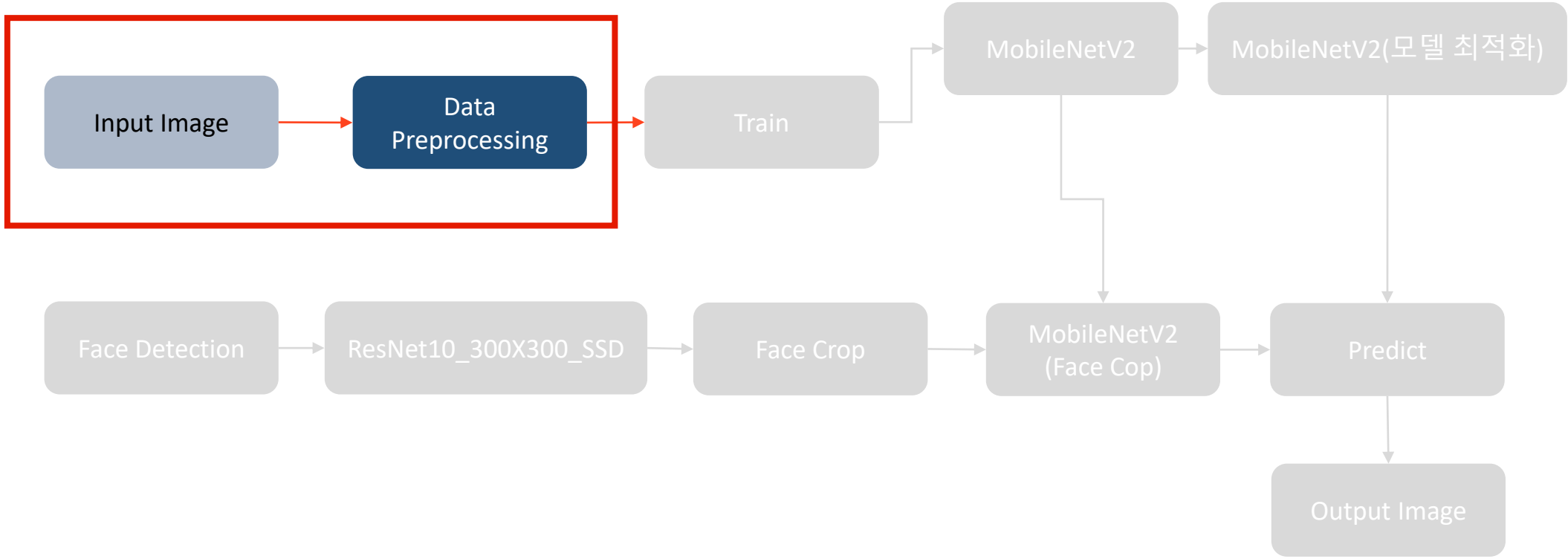
04 Flow Chart Algorithm

졸음운전 방지 예방을 위한 차량내 전방주시 감지 알고리즘



04-1 Flow Chart Algorithm

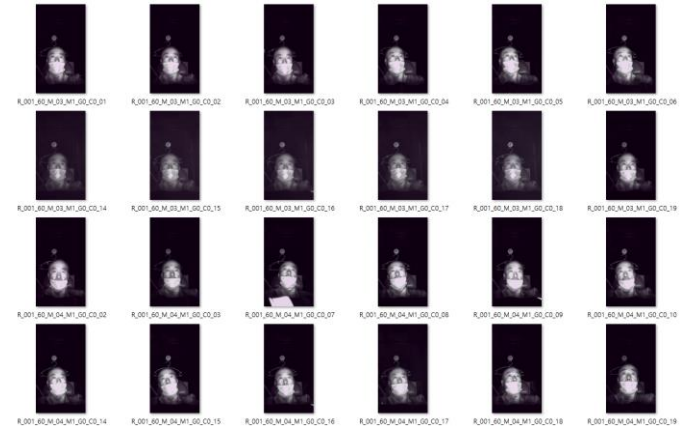
데이터 전처리



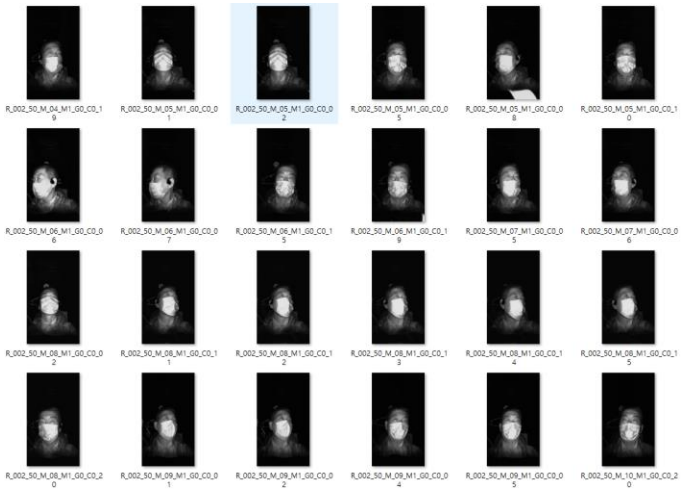
04-2 데이터셋 수집

졸음운전 예방을 위한 운전자 상태 정보 영상 데이터 수집

전방 주시



비전방 주시



- Train_Data : 4,600장
  - Normal(전방 주시) : 3,639장
  - Abnormal(비전방 주시) : 961장
- Image_size : 720 x 1280

04-3 데이터 전처리

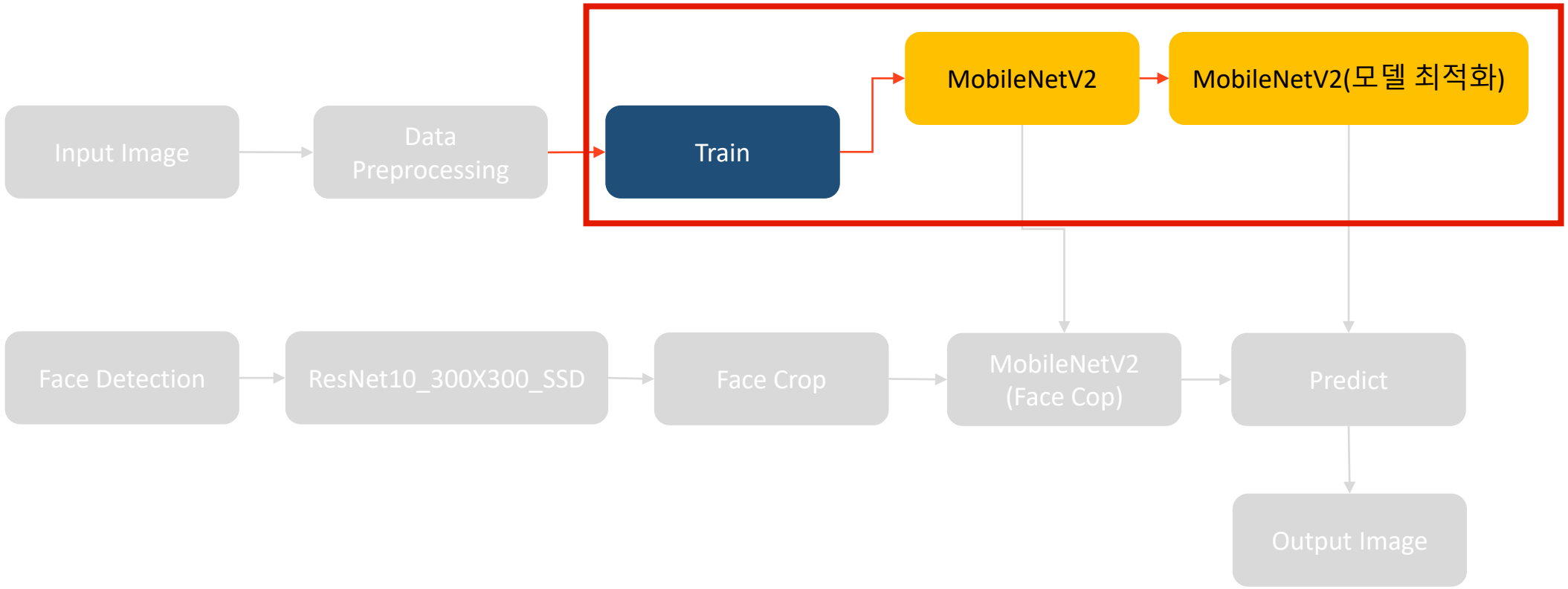
Input Image
(720 X 1280)

Image Resize
(224 X 224)

데이터 증식
(Data Augmentation)

04-4 Flow Chart Algorithm

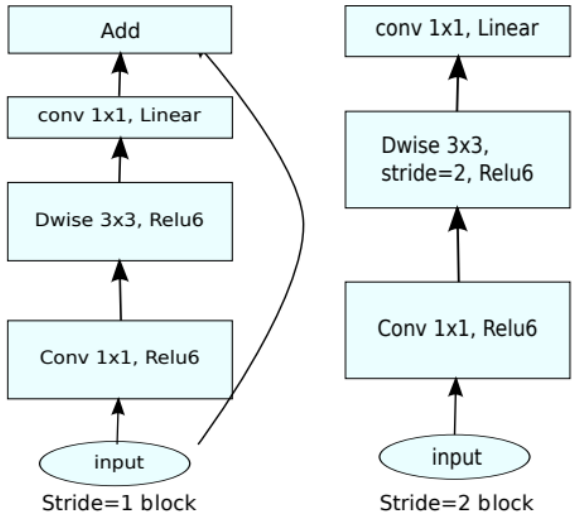
MboileNetV2 Train



04-5 모델

MobileNetV2

- MobileNetV2 Convolution block



(d) Mobilenet V2

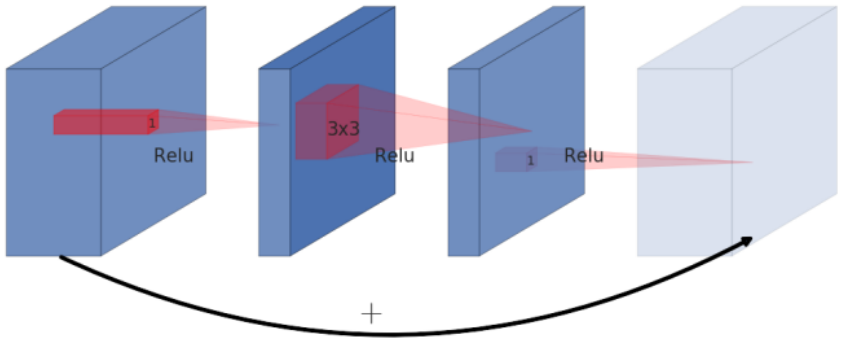
- Architecture

Input	Operator	t	c	n	s
$224^2 \times 3$	conv2d	-	32	1	2
$112^2 \times 32$	bottleneck	1	16	1	1
$112^2 \times 16$	bottleneck	6	24	2	2
$56^2 \times 24$	bottleneck	6	32	3	2
$28^2 \times 32$	bottleneck	6	64	4	2
$14^2 \times 64$	bottleneck	6	96	3	1
$14^2 \times 96$	bottleneck	6	160	3	2
$7^2 \times 160$	bottleneck	6	320	1	1
$7^2 \times 320$	conv2d 1x1	-	1280	1	1
$7^2 \times 1280$	avgpool 7x7	-	-	1	-
$1 \times 1 \times 1280$	conv2d 1x1	-	k	-	

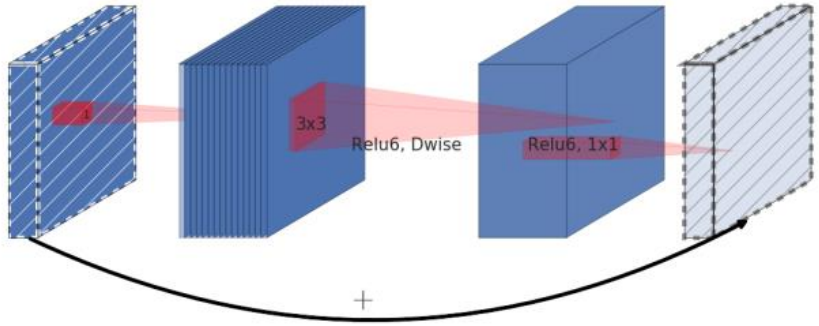
04-6 모델

MobileNetV2

- wide → narrow → wide 형태가 되어 가운데 narrow 형태가 bottleneck 구조 설계
- 처음에 들어오는 입력은 채널이 많은 wide한 형태이고 1x1 convolution을 이용하여 채널을 줄여 다음 layer에서 bottleneck을 설계
- Bottleneck에서는 3x3 convolution을 이용하여 convolution 연산을 하게 되고 다시 skip connection과 합쳐 지기 위하여 원래의 사이즈로 복원함



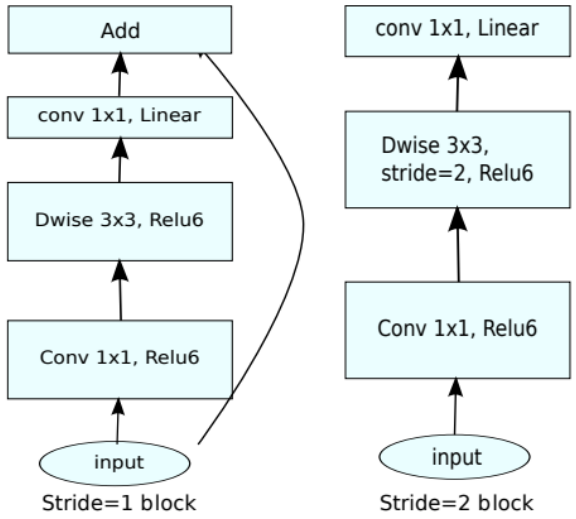
- 이 논문에서 제안된 inverted residual은 일반적은 residual block과 정반대로 움직임
- 처음 입력으로 그려진 점선 형태의 feature는 linear bottleneck 이다 (ReLU를 거치지 않음)
- 즉 narrow → wide → narrow 구조로 skip connectio을 합쳐짐
- 목적은 압축된 narrow layer를 skip connection으로 사용함으로써 메모리 사용량을 줄이기 위함이다



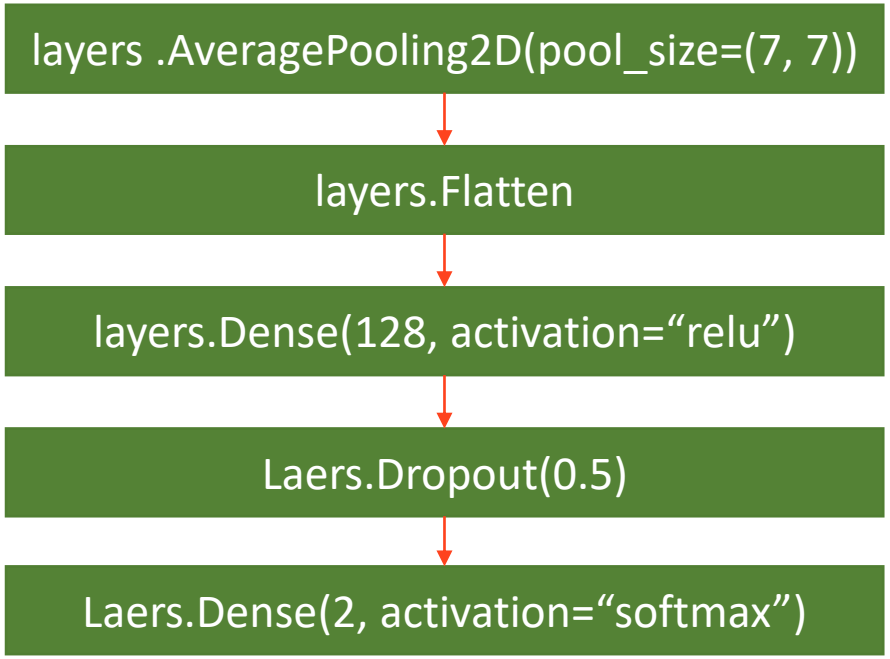
04-7 모델

MobileNetV2(모델 최적화)

- MobileNetV2 + User Layer Add



(d) Mobilenet V2



04-8 성능 평가

- Epoch : 100
- Accuracy : 93%
- Loss : 18%

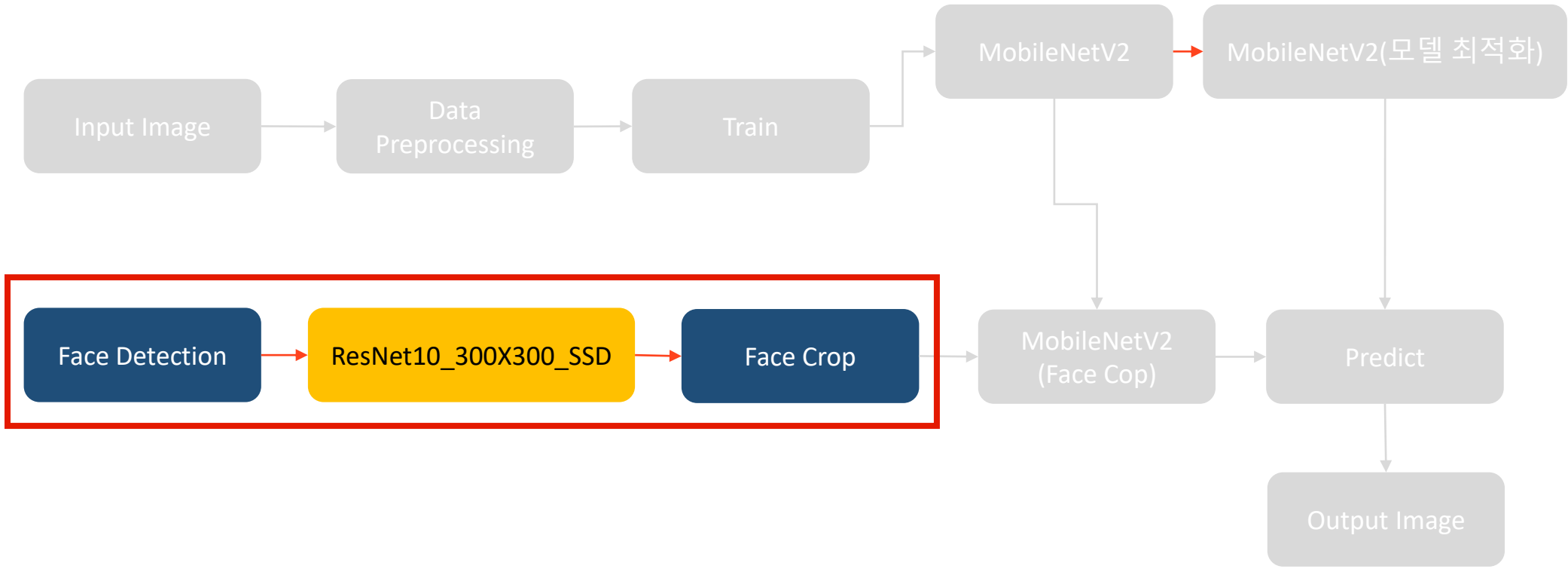

```
Epoch 90/100
115/115 [==============================] - 161s 1s/step - loss: 0.1854 - accuracy: 0.9274 - val_loss: 0.1812 - val_accuracy: 0.9370
Epoch 91/100
115/115 [==============================] - 167s 1s/step - loss: 0.1817 - accuracy: 0.9321 - val_loss: 0.1810 - val_accuracy: 0.9337
Epoch 92/100
115/115 [==============================] - 162s 1s/step - loss: 0.1841 - accuracy: 0.9307 - val_loss: 0.1848 - val_accuracy: 0.9293
Epoch 93/100
115/115 [==============================] - 164s 1s/step - loss: 0.1936 - accuracy: 0.9307 - val_loss: 0.1828 - val_accuracy: 0.9370
Epoch 94/100
115/115 [==============================] - 167s 1s/step - loss: 0.2020 - accuracy: 0.9198 - val_loss: 0.1816 - val_accuracy: 0.9337
Epoch 95/100
115/115 [==============================] - 165s 1s/step - loss: 0.1944 - accuracy: 0.9242 - val_loss: 0.1922 - val_accuracy: 0.9283
Epoch 96/100
115/115 [==============================] - 164s 1s/step - loss: 0.1920 - accuracy: 0.9255 - val_loss: 0.1789 - val_accuracy: 0.9359
Epoch 97/100
115/115 [==============================] - 163s 1s/step - loss: 0.1838 - accuracy: 0.9285 - val_loss: 0.1848 - val_accuracy: 0.9261
Epoch 98/100
115/115 [==============================] - 161s 1s/step - loss: 0.1845 - accuracy: 0.9307 - val_loss: 0.1837 - val_accuracy: 0.9348
Epoch 99/100
115/115 [==============================] - 160s 1s/step - loss: 0.1884 - accuracy: 0.9231 - val_loss: 0.1791 - val_accuracy: 0.9337
Epoch 100/100
115/115 [==============================] - 159s 1s/step - loss: 0.1848 - accuracy: 0.9304 - val_loss: 0.1812 - val_accuracy: 0.9391
```

```
model.evaluate(testX, testY)
```

```
29/29 [==============================] - 3s 98ms/step - loss: 0.1822 - accuracy: 0.9370
```

04-9 Flow Chart Algorithm

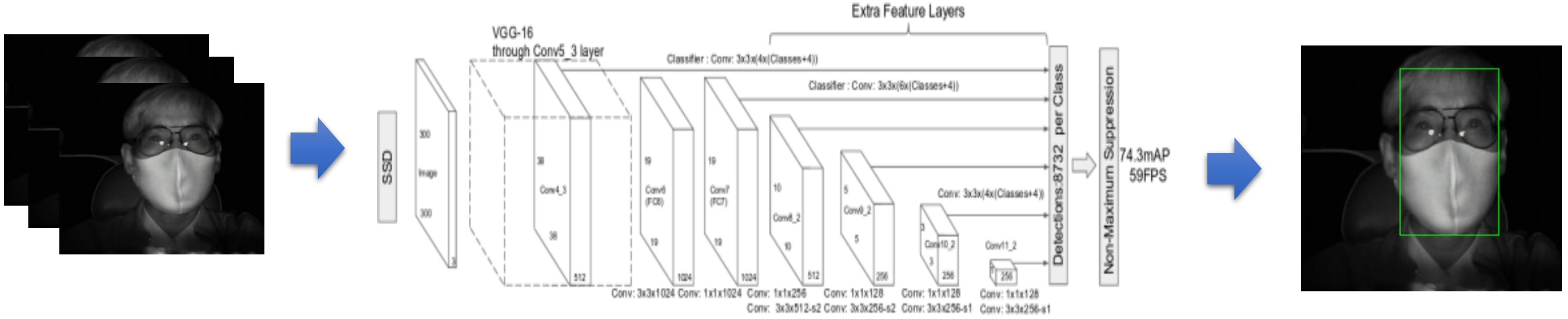
프로젝트 기간 및 구체적 활동



04-10 Face Detection

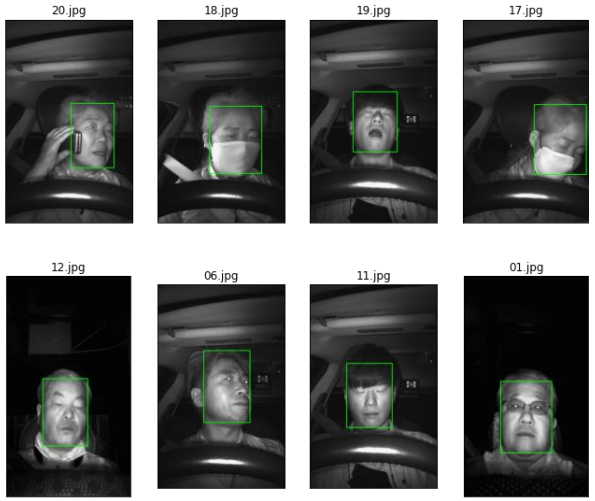
ResNet SSD 모델

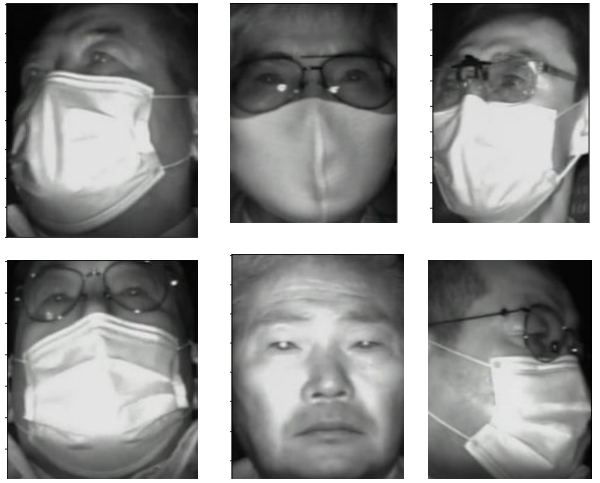
- SSD는 VGG19 네트워크를 Feature 추출기로 사용하는 Single Shot Detector이다. (Faster R-CNN의 CNN과 동일)



Test Image → ResNet10_300X300_SSD → Result

04-11 Face Detection

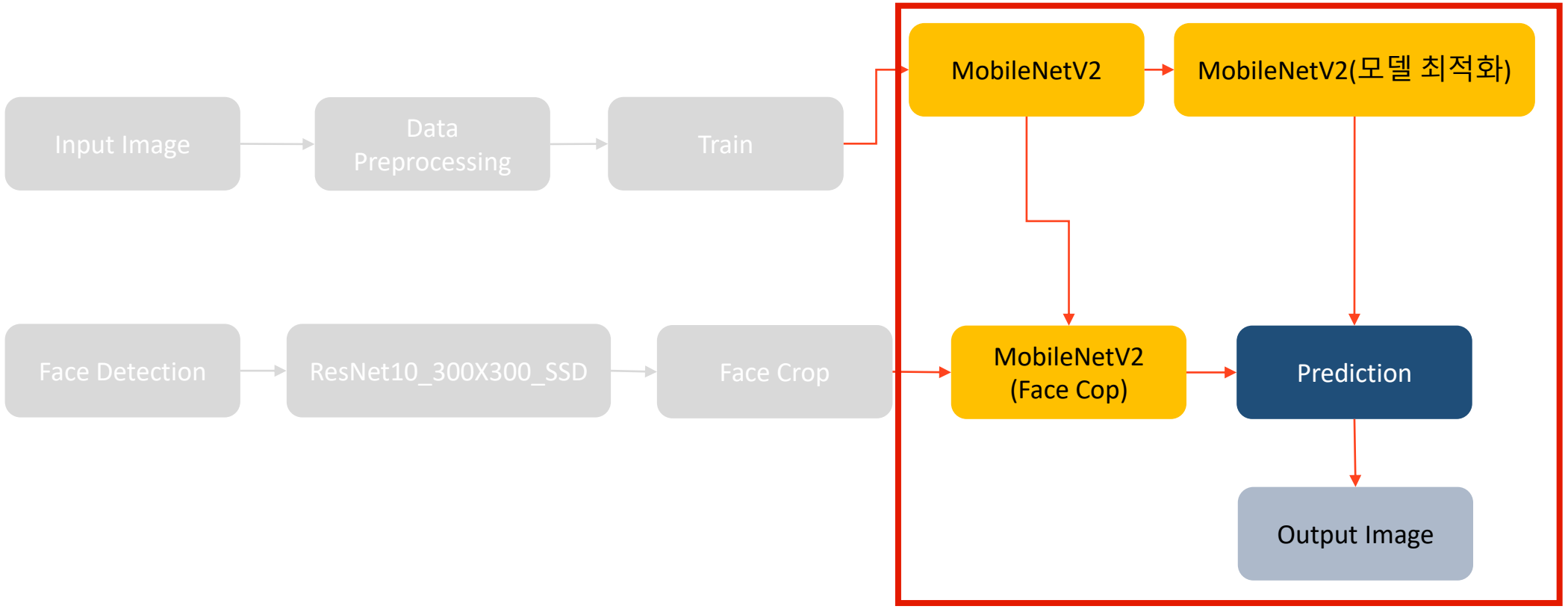




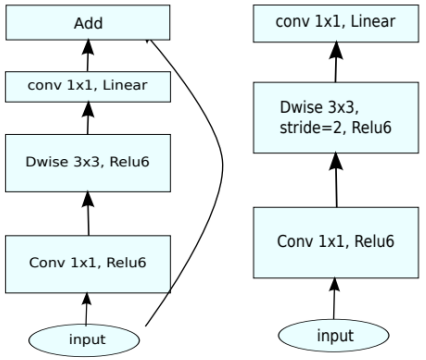
Face Detection → Face Crop

04-12 Flow Chart Algorithm

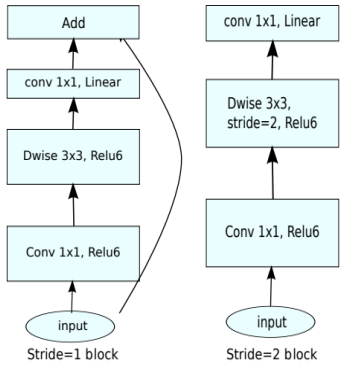
프로젝트 기간 및 구체적 활동



04-13 Prediction



(d) Mobilenet V2



(d) Mobilenet V2

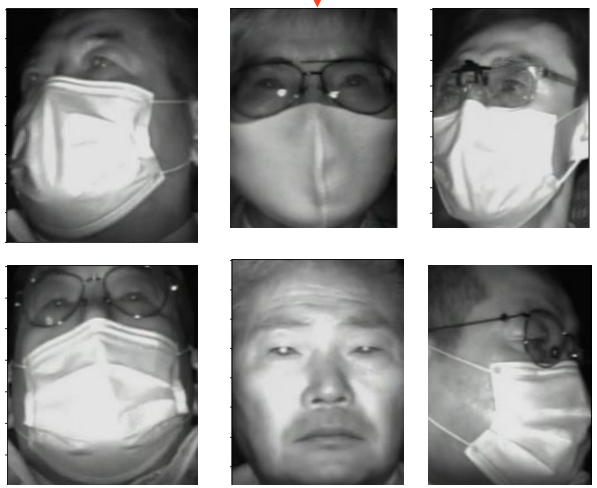
layers .AveragePooling2D(pool_size=(7, 7))

layers.Flatten

layers.Dense(128, activation="relu")

Laers.Dropout(0.5)

Laers.Dense(1, activation="softmax")

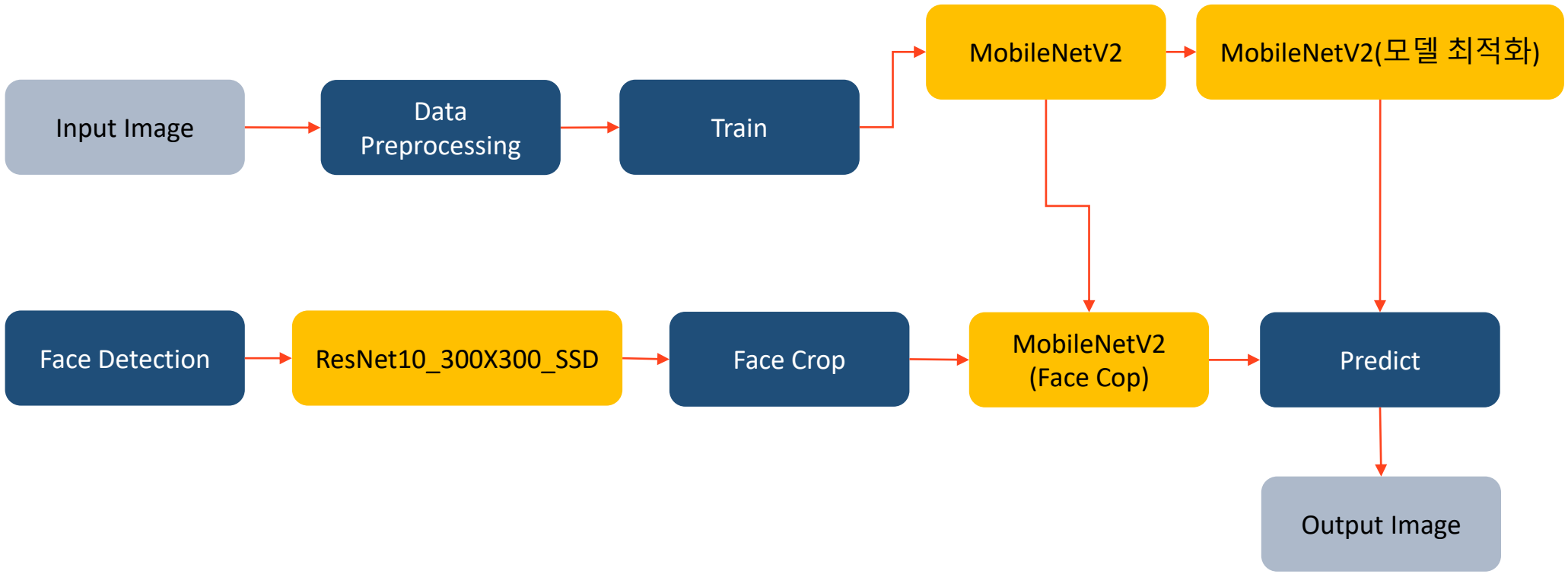


Prediction

Output Image

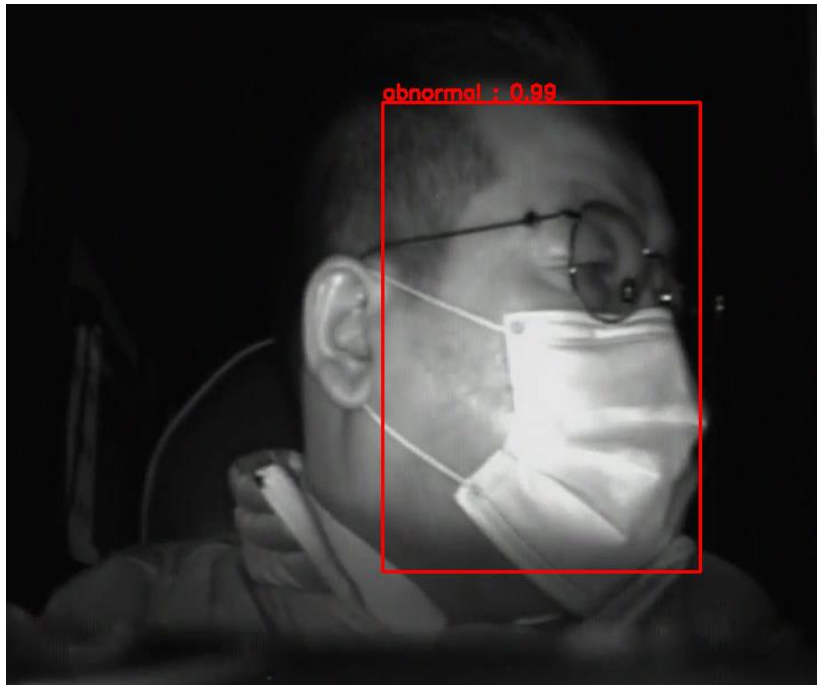
04-14 Flow Chart Algorithm

프로젝트 기간 및 구체적 활동



05 결과

전방 주시 : 93%



비 전방 주시 : 99%

06 참고 문헌

참고문헌 및 참고 사이트

[1] AI Hub ,「 졸음운전 예방을 위한 운전자 상태 정보 영상」,『 개방데이터/안전』, https://aihub.or.kr/aidata/30744, 2021.11.02 검색

[2] 까먹으면 다시 보려고 정리하는 블로그,「[논문 읽기] MobileNetV2(2018) 리뷰, MobileNetV2: Inverted Residuals and Linear Bottlenecks」, 『딥러닝 공부방』,https://deep-learning-study.tistory.com/541 , 2021.11.04 검색