한국IT교육원

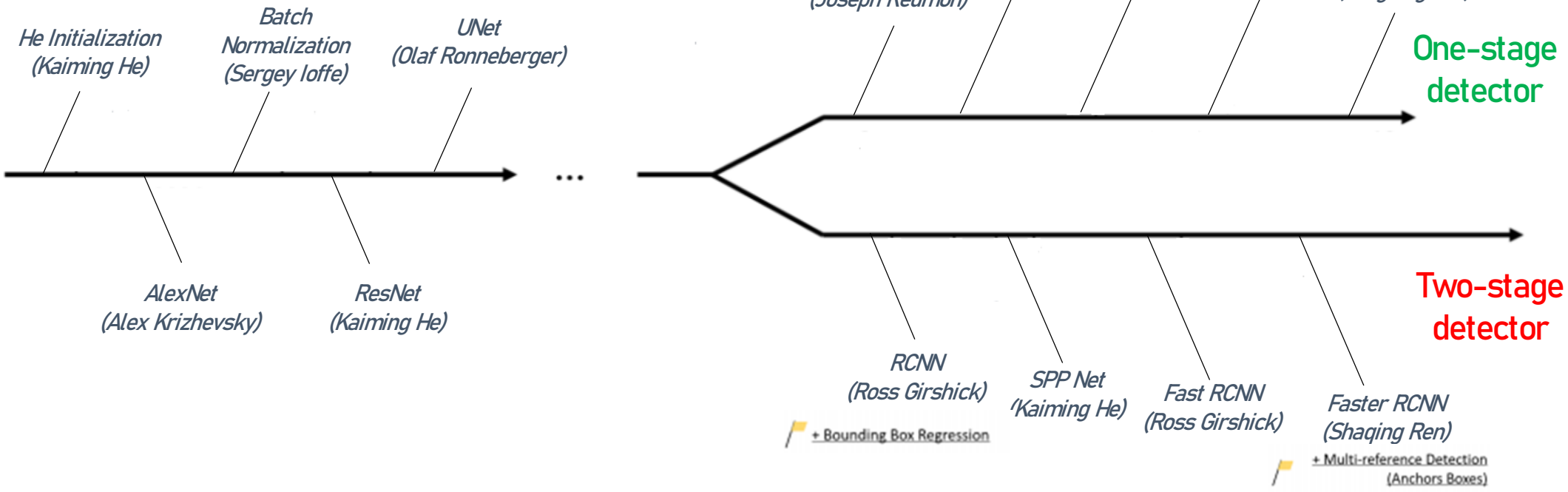
자율주행을 위한 U-Net구현

01 프로젝트 수행을 위한 사전 스터디

프로그래밍 학습

- Python 및 TensorFlow, Keras 등 모델 생성 및 학습을 위한 프로그래밍 Tool 학습
- 사전 훈련된 모델을 사용하여 전이학습 및 Fine-Tuning 실습

논문 리뷰



02 프로젝트 개요

기획의도

- 최근 자율주행 차량에 대한 수요와 관심이 폭발적으로 증가하고 있으며, 완성차 업체마다 자율주행 기능이 탑재된 전기차를 경쟁적으로 출시
- 정부는 2025년까지 운전자 개입이 없는 완전 자율주행(Level 4)버스와 택시를, 2027년까지는 승용차를 출시하겠다는 계획안 발표
- 지난 수년간 자율주행에 필수적인 정확한 객체 인식(Object detection)을 위해 수많은 딥러닝 모델 (RCNN, Yolo, SSD 등) 및 알고리즘들이 제시 되어 왔음.
- U-Net은 생체의학 분야에서 이미지 분할(image segmentation)을 목적으로 제안된 모델로 자율주행에서 정확한 객체 인식을 위해 사용되어질 수 있음.

기대효과

- U-Net의 이미지 분할효과를 통해 더욱 정확하고 안전한 자율주행 알고리즘 및 모델 구현에 도움을 줄 것으로 기대함.
- 훈련생들이 직접 모델을 구현하고 학습시키는 프로젝트를 통해 지난 4주간 익힌 딥러닝 관련 지식을 더욱 견고히 할 수 있음.

03 프로젝트 개요

프로젝트 개요

- U-Net을 활용한 이미지 분할(image segmentation) 구현

활용 장비 및 재료(개발 환경 등)

- Google Colab, Python, Tensorflow, Keras, android studio 등

프로젝트 구조

- Ground Truth가 포함된 Dataset 획득 -> 프로젝트 모델 구현 및 학습 -> 이미지 분할(Image segmentation) 구현 -> 모바일 환경(android)에서 이미지 분할(Image segmentation) 구현

04 프로젝트 수행 결과

학습 데이터 소개(Training)


dog cat horse


tv/monitor boat bus


train bottle airplane person bird


cow sofa dining table sheep chair


bicycle potted plant motorbike car

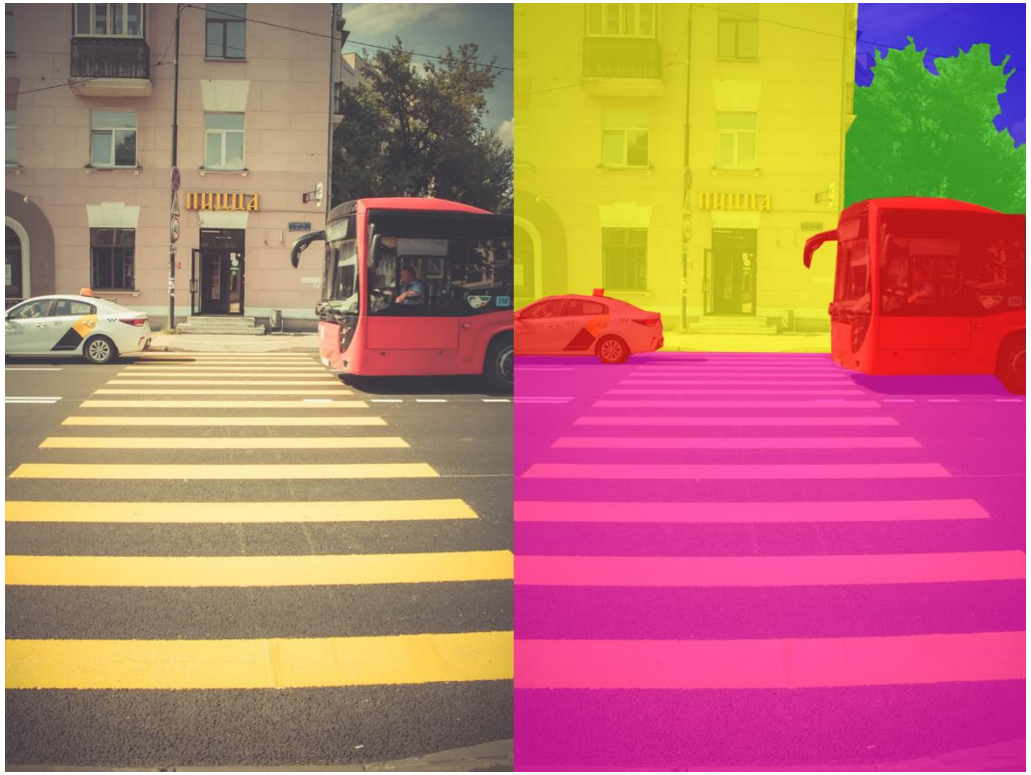
사용 데이터셋 : VOCtrainval_11-May-2012

- Person : person
- Animal : bird, cat, cow, dog, horse, sheep
- Vehicle : airplane, bicycle, boat, bus, car, motorbike, train
- Indoor : bottle, chair, dining table, potted plant, sofa, tv/monitor

<총 17,125개의 Data>

04 프로젝트 수행 결과

Image semantic segmentation

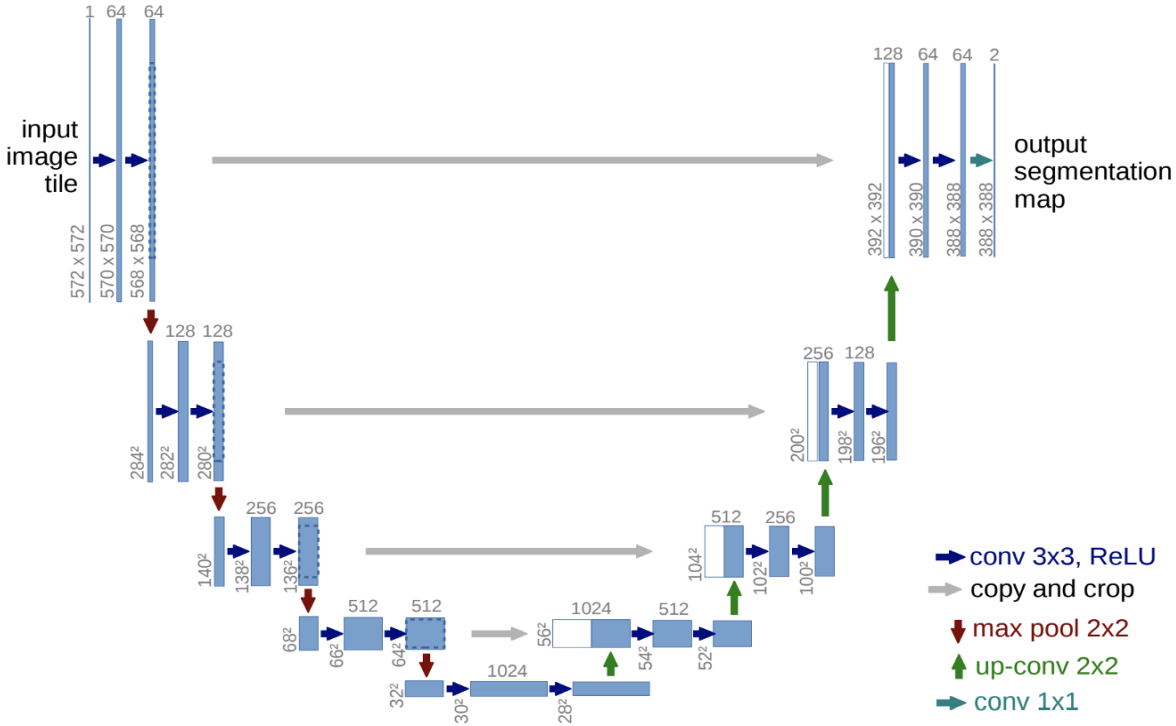


[출처] http://datagen.tech/guides/image-annotation/image-segmentation/#

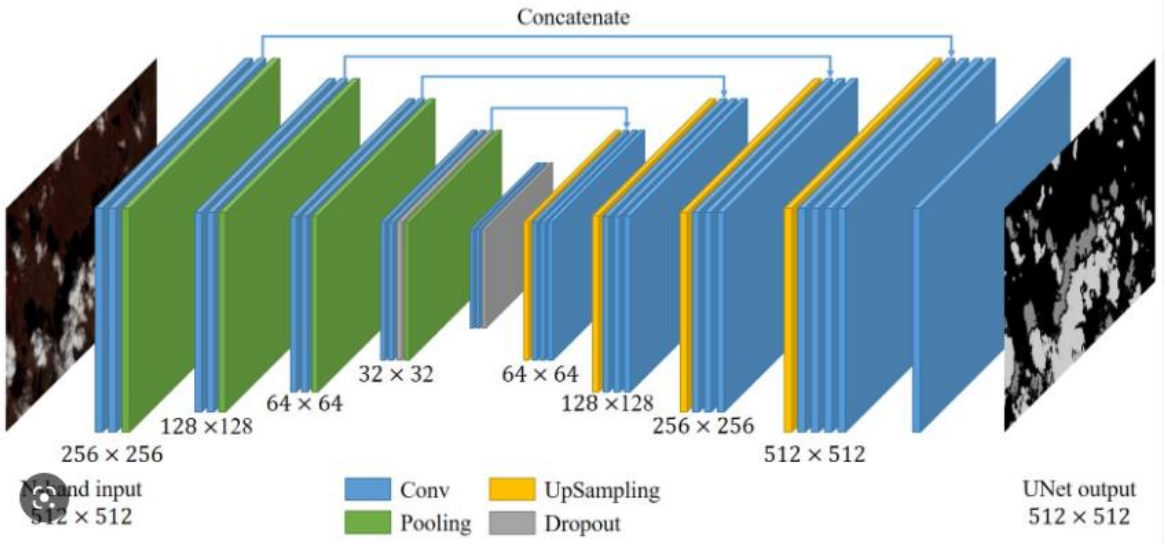
[출처] https://thegradient.pub/semantic-segmentation

04 프로젝트 수행 결과

U-Net모델의 구조



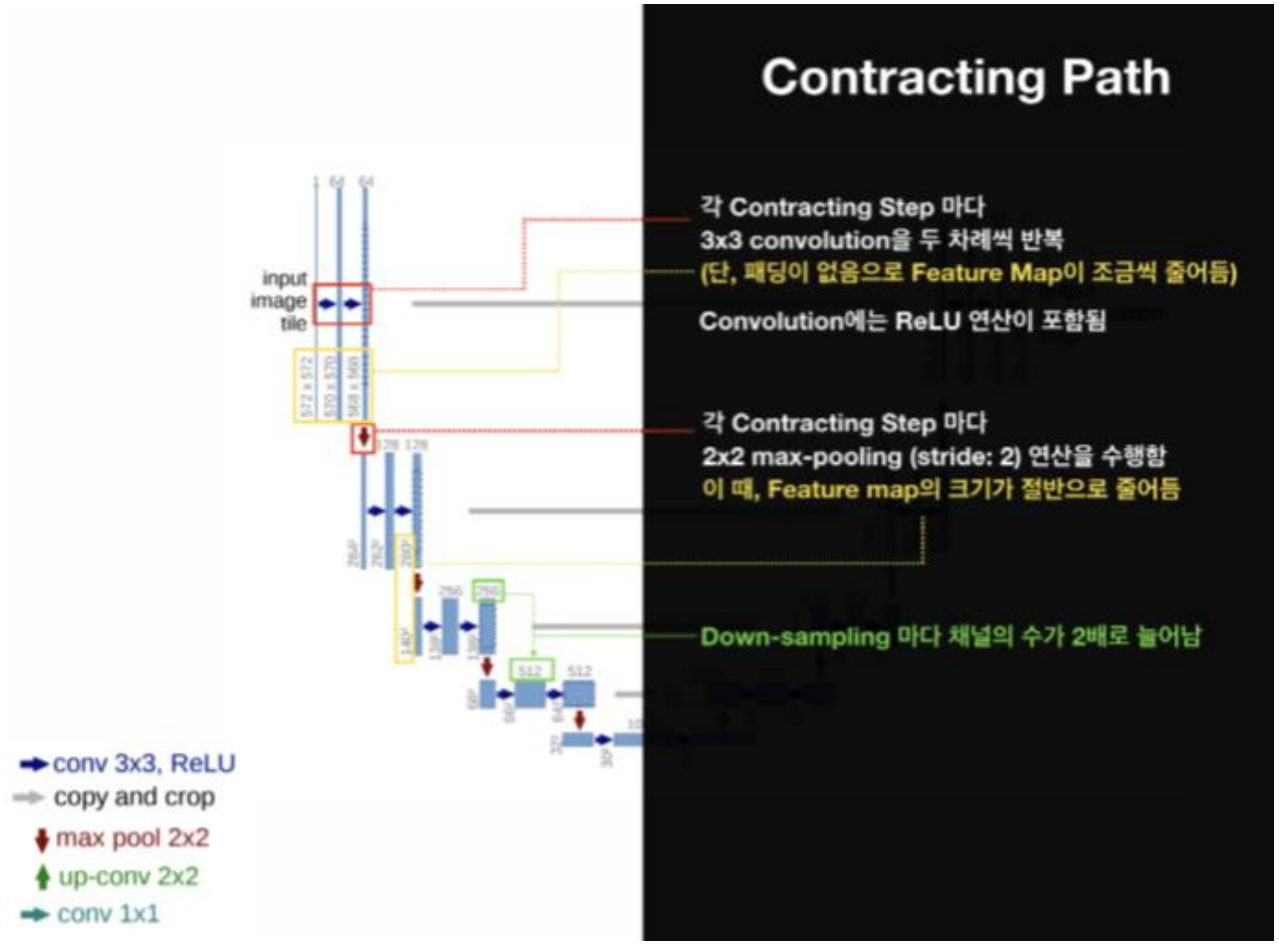
[출처] https://medium.com/@msmapark2/u-net-%EB%85%BC%EB%AC%B8-%EB%A6%AC%EB%B7%B0-u-net-convolutional-networks-for-biomedical-image-segmentation-456d6901b28a



[출처] https://www.mdpi.com/2072-4292/12/12/2001

04 프로젝트 수행 결과

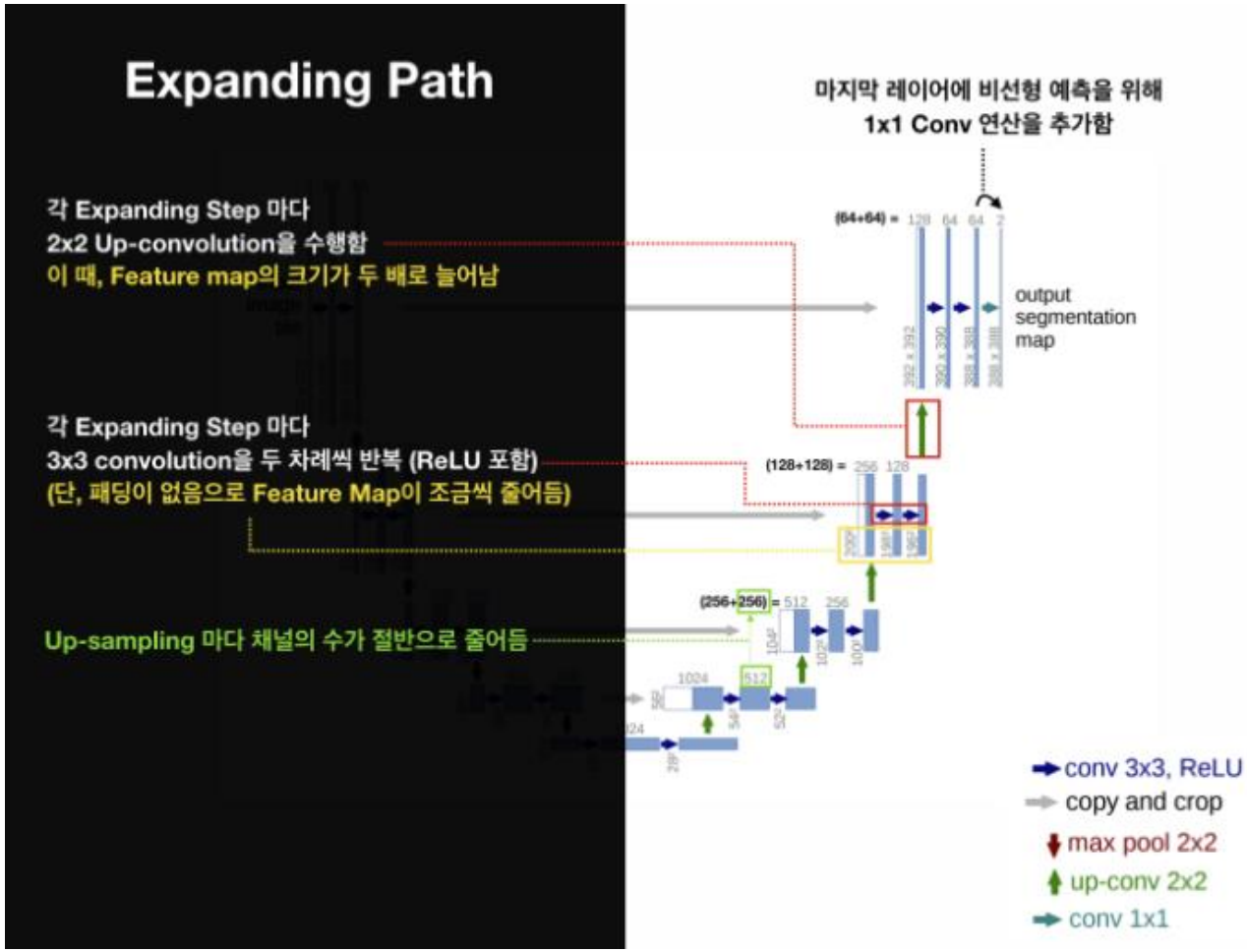
- U-Net모델의 구조



[출처] https://medium.com/@msmapark2/u-net-%EB%85%BC%EB%AC%B8-%EB%A6%AC%EB%B7%B0-u-net-convolutional-networks-for-biomedical-image-segmentation-456d6901b28a

04 프로젝트 수행 결과

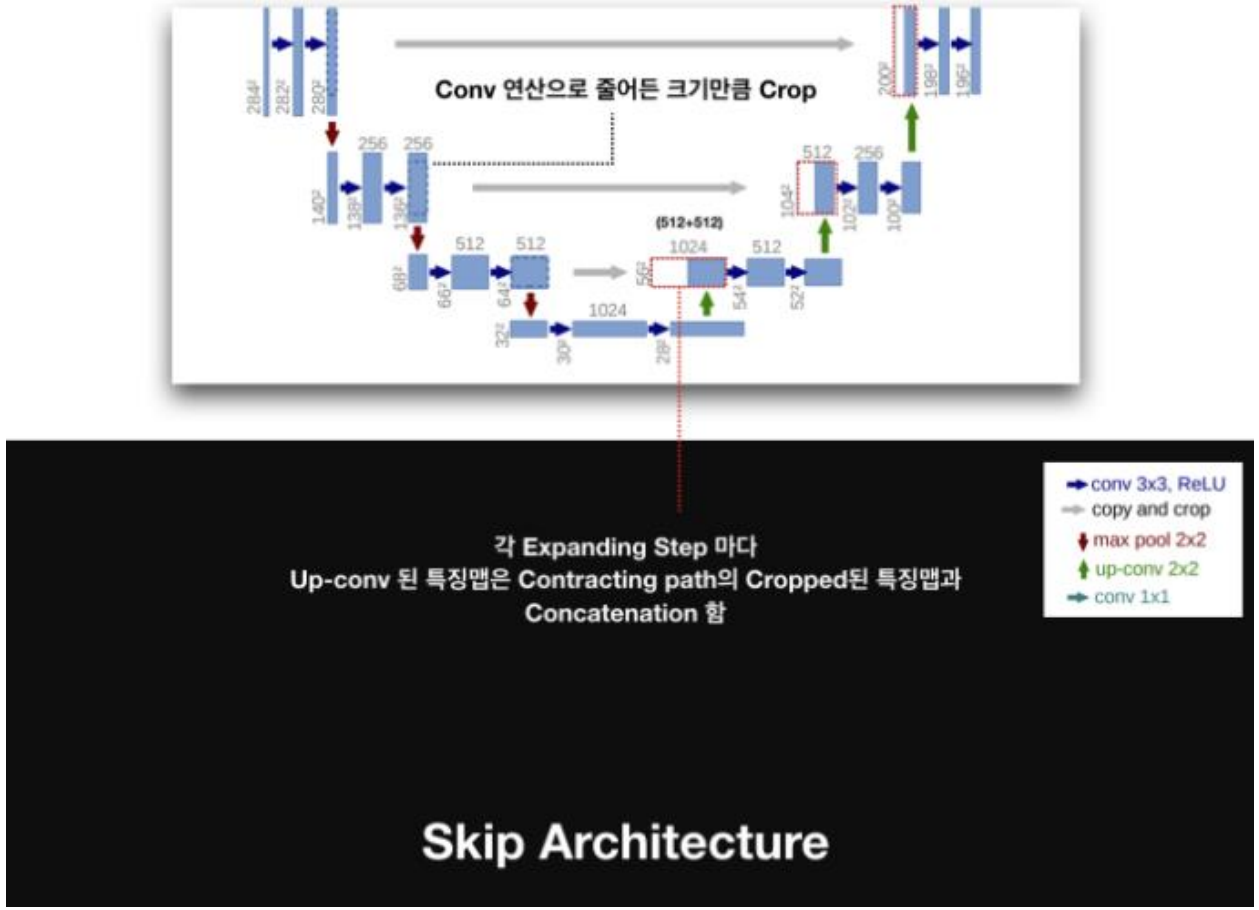
U-Net모델의 구조



[출처] https://medium.com/@msmapark2/u-net-%EB%85%BC%EB%AC%B8-%EB%A6%AC%EB%B7%B0-u-net-convolutional-networks-for-biomedical-image-segmentation-456d6901b28a

04 프로젝트 수행 결과

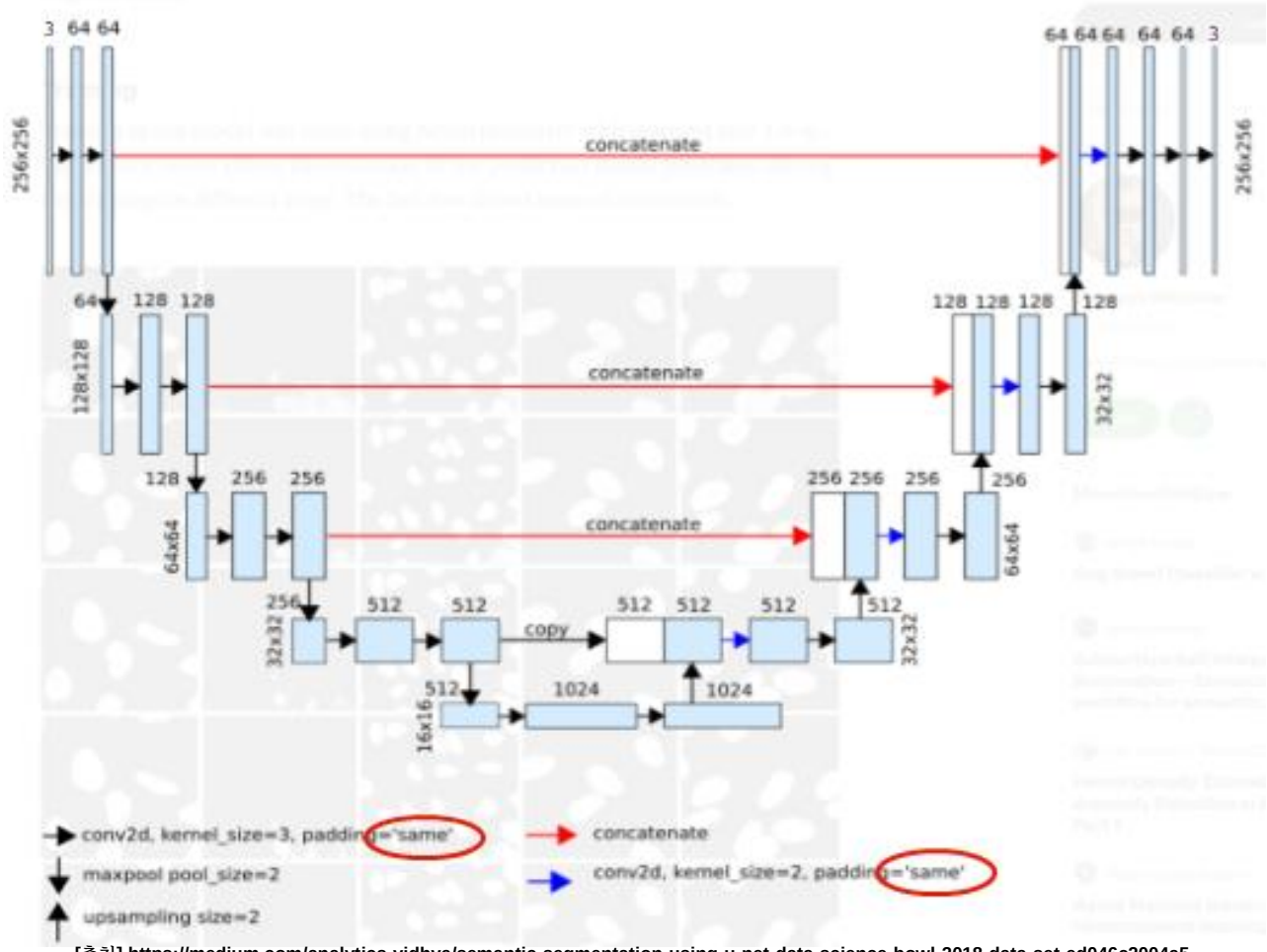
- U-Net모델의 구조



[출처] https://medium.com/@msmapark2/u-net-%EB%85%BC%EB%AC%B8-%EB%A6%AC%EB%B7%B0-u-net-convolutional-networks-for-biomedical-image-segmentation-456d6901b28a

04 프로젝트 수행 결과

- 프로젝트 구현 U-Net 모델(“same” convolution 적용)



[출처] https://medium.com/analytics-vidhya/semantic-segmentation-using-u-net-data-science-bowl-2018-data-set-ed046c2004a5

04 프로젝트 수행 결과

U-Net 모델 구현

```
import tensorflow as tf
import numpy as np

from tensorflow.keras.layers import Input
from tensorflow.keras.layers import Conv2D
from tensorflow.keras.layers import MaxPooling2D
from tensorflow.keras.layers import Dropout
from tensorflow.keras.layers import Conv2DTranspose
from tensorflow.keras.layers import concatenate

from test_utils import summary, comparator
```

Import module1

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

import imageio

import matplotlib.pyplot as plt
%matplotlib inline
```

Import module2

04 프로젝트 수행 결과

U-Net 모델 구현

```
path = ''
image_path = os.path.join(path, '/content/gdrive/MyDrive/Colab Notebooks/coursera programming practice/class4/week3/Image Segmentation with UNet/data/CameraRGB/')
mask_path = os.path.join(path, '/content/gdrive/MyDrive/Colab Notebooks/coursera programming practice/class4/week3/Image Segmentation with UNet/data/CameraMask/')
image_list = sorted(os.listdir(image_path)) #listdir : 경로에 있는 모든 파일명으로 리스트를 만듬
mask_list = sorted(os.listdir(mask_path))
image_list = sorted([image_path+i for i in image_list])
mask_list = sorted([mask_path+i for i in mask_list])
```

```
image_filenames = tf.constant(image_list) #image_list라는 리스트를 image_filenames라는 이름의 1차원 텐서로 만듬
masks_filenames = tf.constant(mask_list)

dataset = tf.data.Dataset.from_tensor_slices((image_filenames, masks_filenames)) #주어진 데이터소스를 여러 텐서로 자른다.

for image, mask in dataset.take(1):
    print(image)
    print(mask)
```

Dataset 객체 생성

04 프로젝트 수행 결과

U-Net 모델 구현

```python
def process_path(image_path, mask_path):
    img = tf.io.read_file(image_path)
    img = tf.image.decode_png(img, channels=3) #png image파일을 tensor로 변환
    img = tf.image.convert_image_dtype(img, tf.float32) #0~1값으로 normalize
    print("img.shape=", img.shape)

    mask = tf.io.read_file(mask_path)
    mask = tf.image.decode_png(mask, channels=3)
    print("mask.shape=", mask.shape)
    mask = tf.math.reduce_max(mask, axis=-1, keepdims=True)
    print("mask.shape=", mask.shape)
    return img, mask
```

```python
def preprocess(image, mask):
    input_image = tf.image.resize(image, (96, 128), method='nearest')
    input_mask = tf.image.resize(mask, (96, 128), method='nearest')

    return input_image, input_mask
```

```python
image_ds = dataset.map(process_path)

processed_image_ds = image_ds.map(preprocess)
```

학습에 사용 될 dataset 생성

04 프로젝트 수행 결과

U-Net 모델 구현

```
def conv_block(inputs=None, n_filters=32, dropout_prob=0, max_pooling=True):

    conv = Conv2D(n_filters,
                  3,
                  activation='relu',
                  padding='same',
                  kernel_initializer='he_normal')(inputs)
    conv = Conv2D(n_filters,
                  3,
                  activation='relu',
                  padding='same',
                  kernel_initializer='he_normal')(conv)

    if dropout_prob > 0:
        conv = Dropout(dropout_prob)(conv)

    if max_pooling:
        next_layer = MaxPooling2D(2,strides=2)(conv)

    else:
        next_layer = conv

    skip_connection = conv

    return next_layer, skip_connection
```

Contracting Block

```
def upsampling_block(expansive_input, contractive_input, n_filters=32):

    up = Conv2DTranspose(
                 n_filters,
                 3,
                 strides=2,
                 padding='same')(expansive_input)

    print("up=", up)
    print("contractive_input=", contractive_input)

    merge = concatenate([up, contractive_input], axis=3)
    print("merge=", merge)
    conv = Conv2D(n_filters,
                  3,
                  activation='relu',
                  padding='same',
                  kernel_initializer='he_normal')(merge)
    conv = Conv2D(n_filters,
                  3,
                  activation='relu',
                  padding='same',
                  kernel_initializer='he_normal')(conv)

    return conv
```

Expanding Block

04 프로젝트 수행 결과

U-Net 모델 구현

```
def unet_model(input_size=(96, 128, 3), n_filters=32, n_classes=23):

    inputs = Input(input_size)

    cblock1 = conv_block(inputs=inputs, n_filters=n_filters*1)
    cblock2 = conv_block(inputs=cblock1[0], n_filters=n_filters*2)
    cblock3 = conv_block(inputs=cblock2[0], n_filters=n_filters*4)
    cblock4 = conv_block(inputs=cblock3[0], n_filters=n_filters*8,dropout_prob=0.3)
    cblock5 = conv_block(inputs=cblock4[0], n_filters=n_filters*16,dropout_prob=0.3, max_pooling=False)

    ublock6 = upsampling_block(cblock5[0], cblock4[1], n_filters*8)
    ublock7 = upsampling_block(ublock6, cblock3[1], n_filters*4)
    ublock8 = upsampling_block(ublock7, cblock2[1], n_filters*2)
    ublock9 = upsampling_block(ublock8, cblock1[1], n_filters*1)

    conv9 = Conv2D(n_filters,
                3,
                activation='relu',
                padding='same',
                kernel_initializer='he_normal')(ublock9)

    conv10 = Conv2D(n_classes, 1, padding='same')(conv9)

    model = tf.keras.Model(inputs=inputs, outputs=conv10)

    return model
```

전체 모델 구조

```
img_height = 96
img_width = 128
num_channels = 3

unet = unet_model((img_height, img_width, num_channels))
```

모델 생성

04 프로젝트 수행 결과

U-Net 모델 구현

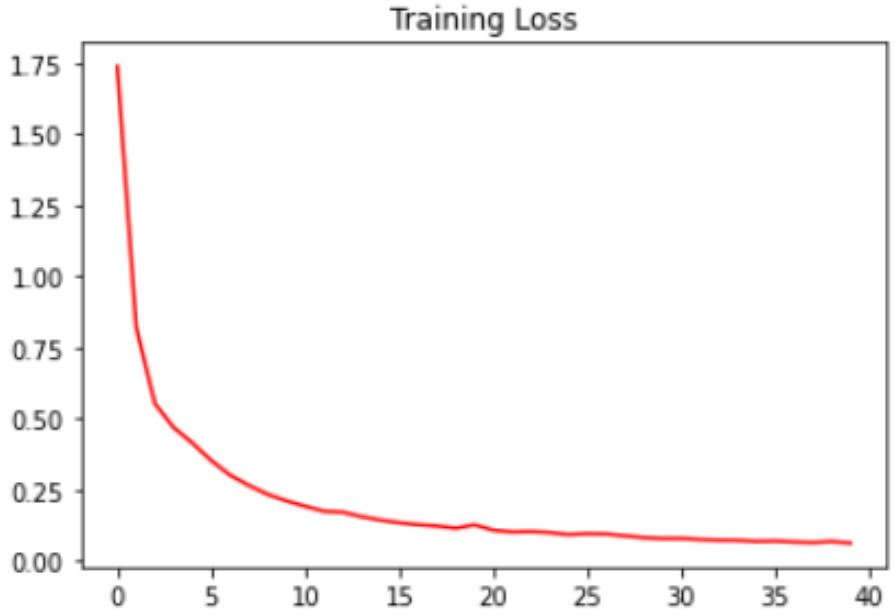
```
unet.compile(optimizer='adam',
             loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True),
             metrics=['accuracy'])
```

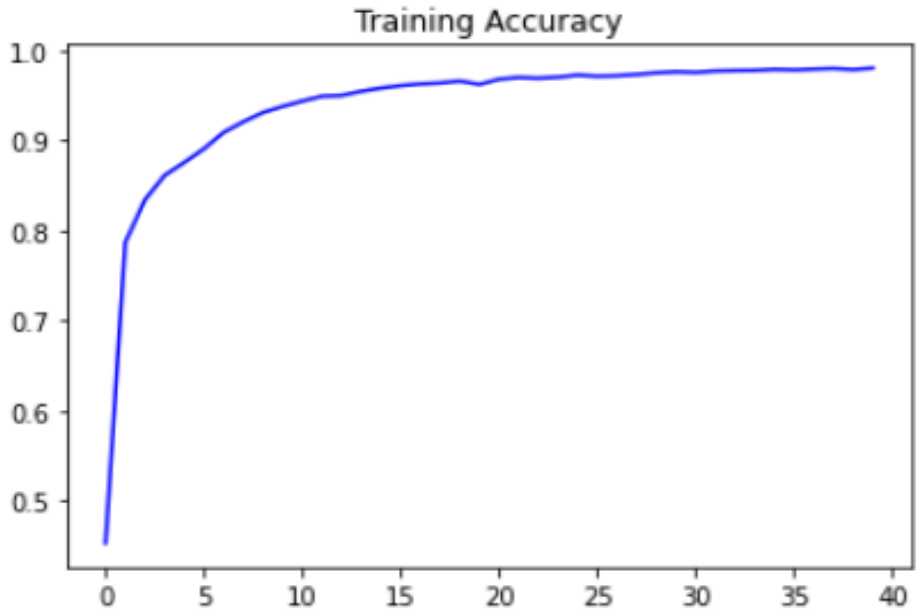
```
EPOCHS = 40
VAL_SUBSPLITS = 5
BUFFER_SIZE = 500
BATCH_SIZE = 32
processed_image_ds.batch(BATCH_SIZE)
train_dataset = processed_image_ds.cache().shuffle(BUFFER_SIZE).batch(BATCH_SIZE)
print(processed_image_ds.element_spec)
model_history = unet.fit(train_dataset, epochs=EPOCHS)
```

모델 학습

04 프로젝트 수행 결과

- U-Net 모델 구현

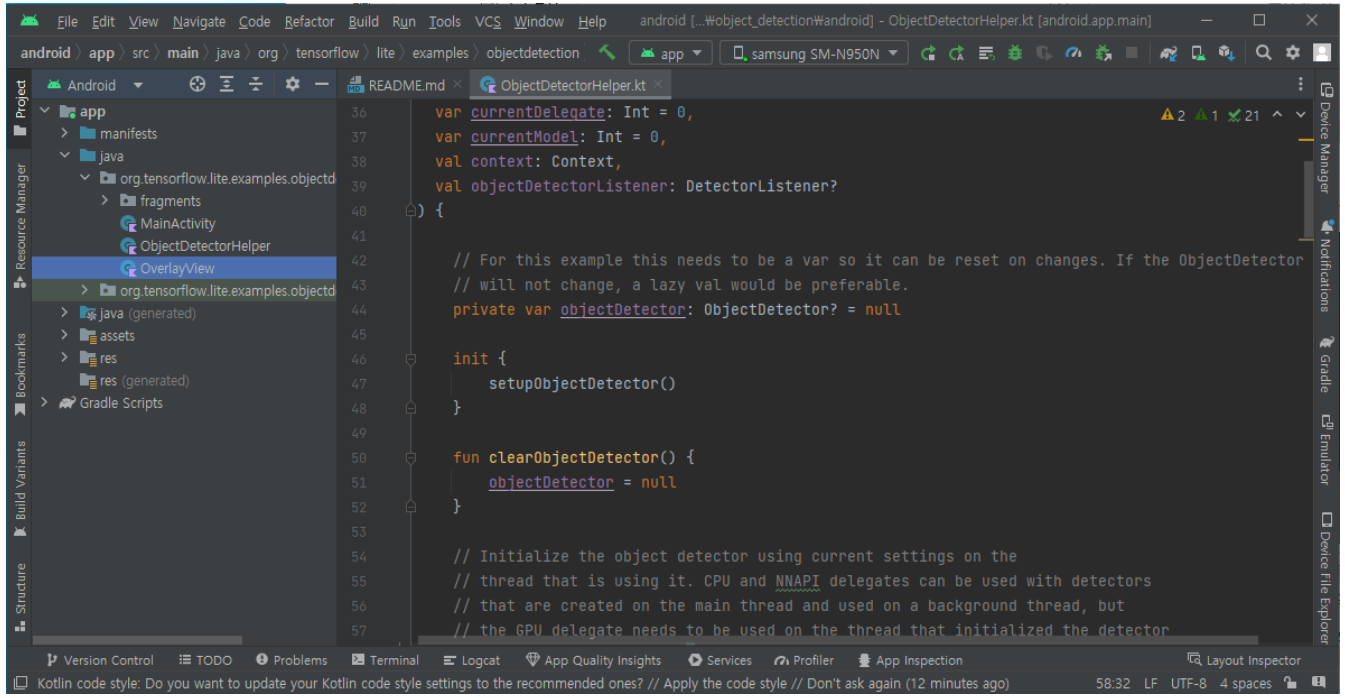


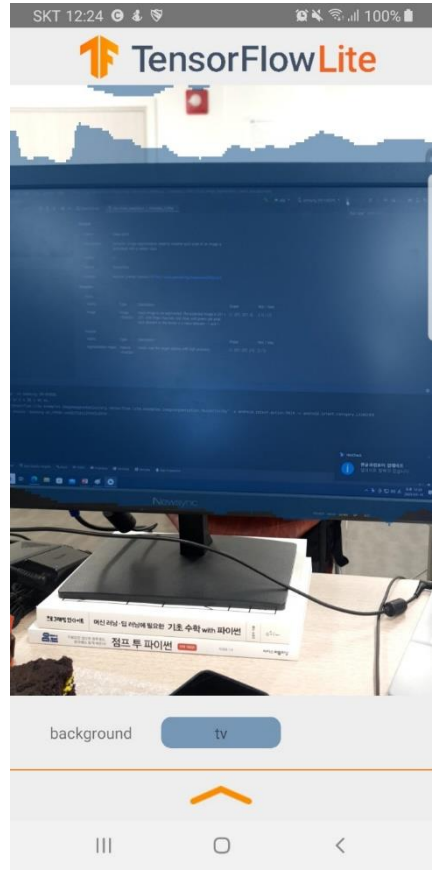


04 프로젝트 수행 결과

Tensorflow Lite 모델 생성 및 Android studio 빌드

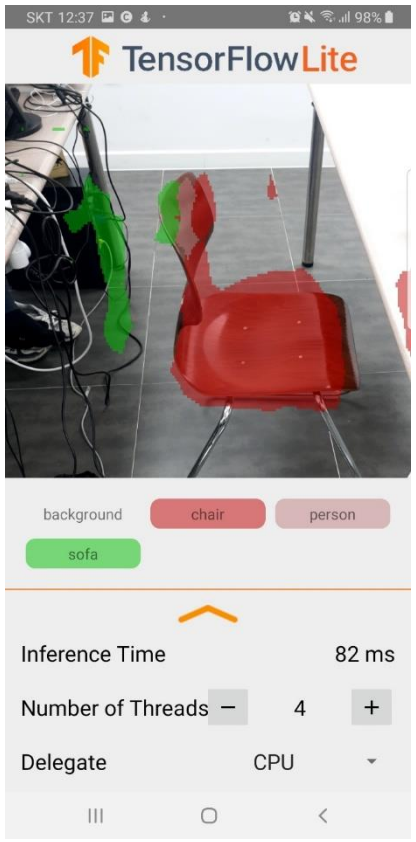
```
converter = tf.lite.TFLiteConverter.from_keras_model(unet)
tflite_model = converter.convert()

with open('unetmodel.tflite', 'wb') as f:
  f.write(tflite_model)
```

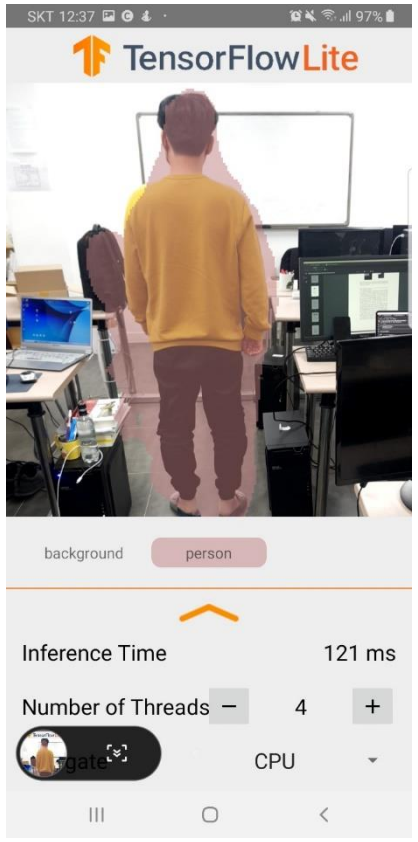


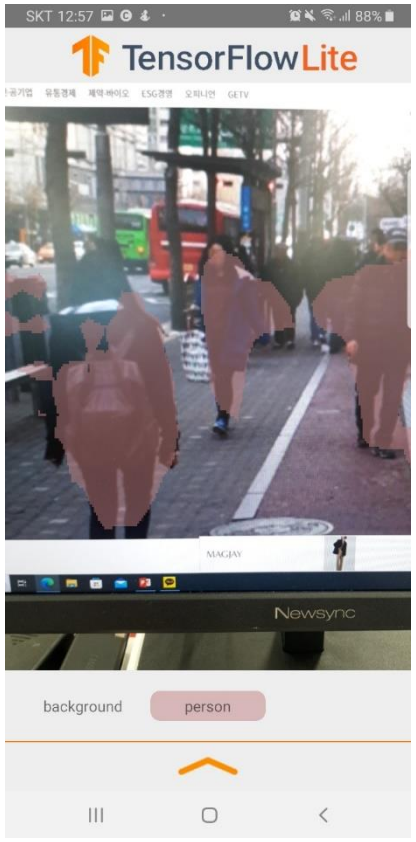


04 프로젝트 수행 결과

테스트 결과







04 프로젝트 수행 결과

학습 데이터 소개(Training)

04 프로젝트 수행 결과

- 모델 시연 영상