**Fashion Image Data Classifier**

**Execution Environment**

Local – CPU : AMD Ryzen 9 4900HS with Radeon Graphics

Anaconda – 4.12.0

python – 3.7.11

tensorflow. version: 2.2.0

numpy version : 1.18.5

pandas version : 1.3.5

keras version : 2.3.1

sklearn version :1.0.2

matplotlib version : 3.5.3

keras\_preprocessing version : 1.1.2

**PreProcessing**

**Image Augmentation**

keras\_preprocessing의 ImageDataGenerator 사용

Dimension, Pointofview를 바꿔가며 데이터 증식

**Resizing**

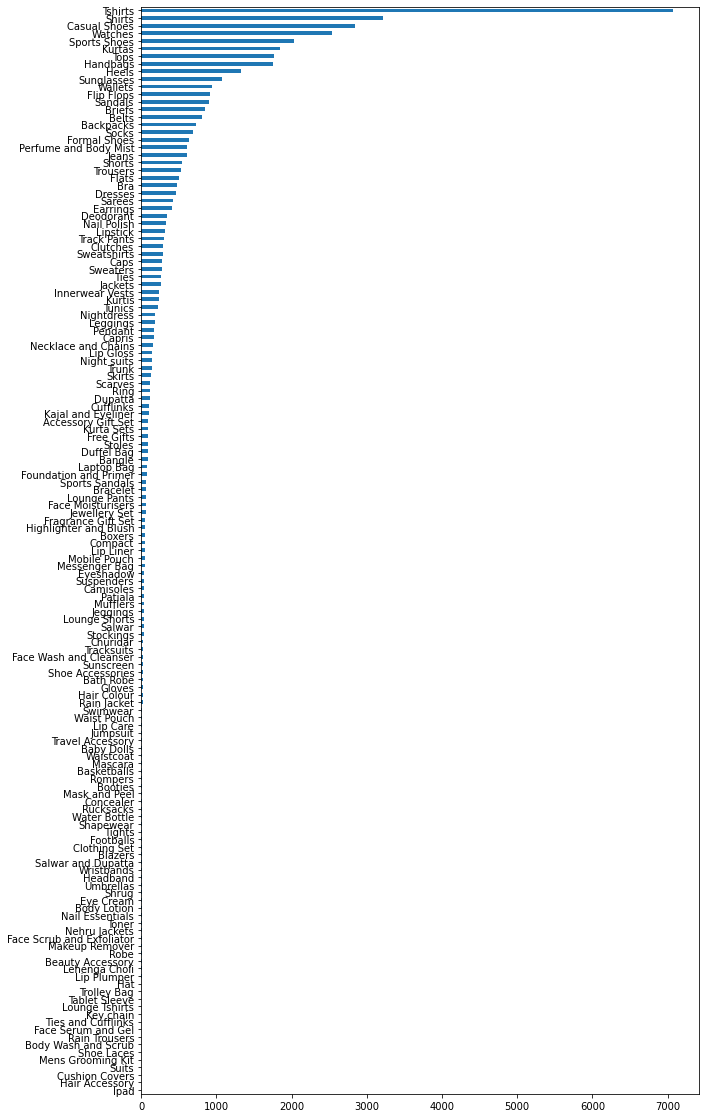
96,96,3 형태로 변환

(96, 96)짜리 정방 행렬 + RGB

**Data Preview**

class : 36 - "Accessories","Apparel Set","Bags","Bath and Body","Beauty Accessories","Belts","Bottomwear","Dress","Eyes","Eyewear","Flip Flops","Fragrance","Free Gifts","Gloves","Headwear","Innerwear","Jewellery","Lips","Loungewear and Nightwear","Makeup","Mufflers","Nails","Sandal","Saree","Scarves","Shoe Accessories","Shoes","Skin","Skin Care","Socks","Sports Accessories","Ties","Topwear","Wallets","Watches","Water Bottle"





**Training**

Sequential Simple CNN model

**Model Summary**

Model: "sequential"

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Layer (type) Output Shape Param #

=================================================================

conv2d (Conv2D) (None, 94, 94, 32) 896

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

conv2d\_1 (Conv2D) (None, 92, 92, 64) 18496

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

max\_pooling2d (MaxPooling2D) (None, 30, 30, 64) 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

dropout (Dropout) (None, 30, 30, 64) 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

conv2d\_2 (Conv2D) (None, 26, 26, 128) 204928

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

max\_pooling2d\_1 (MaxPooling2 (None, 8, 8, 128) 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

dropout\_1 (Dropout) (None, 8, 8, 128) 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

conv2d\_3 (Conv2D) (None, 4, 4, 128) 409728

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

max\_pooling2d\_2 (MaxPooling2 (None, 1, 1, 128) 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

dropout\_2 (Dropout) (None, 1, 1, 128) 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

flatten (Flatten) (None, 128) 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

dense (Dense) (None, 512) 66048

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

dropout\_3 (Dropout) (None, 512) 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

dense\_1 (Dense) (None, 256) 131328

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

dropout\_4 (Dropout) (None, 256) 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

dense\_2 (Dense) (None, 128) 32896

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

dense\_3 (Dense) (None, 128) 16512

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

dense\_4 (Dense) (None, 36) 4644

=================================================================

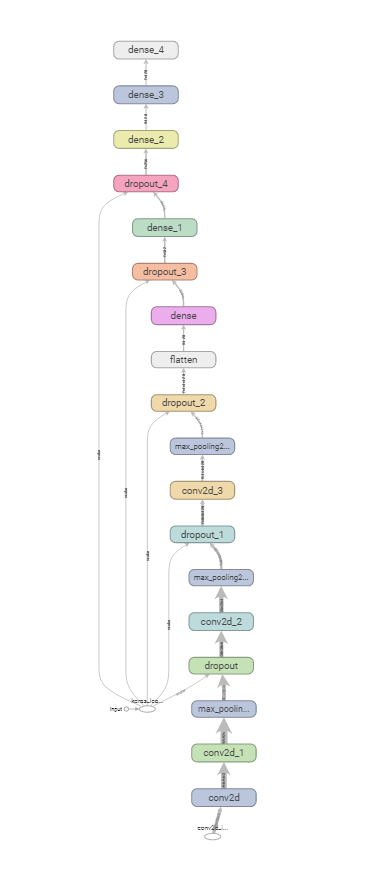
Total params: 885,476

Trainable params: 885,476

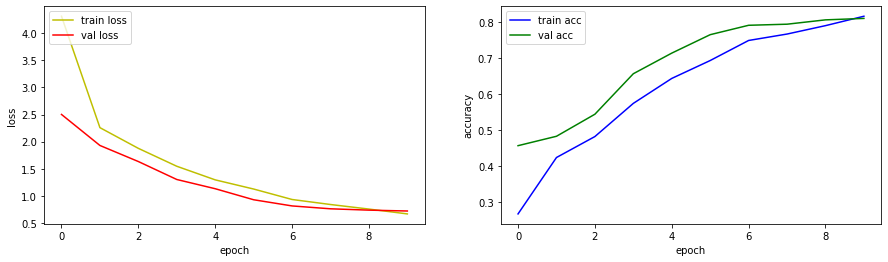
Non-trainable params: 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Visualizing Model Structure by tensorboard**



In Epoch 10

epoch가 지날 수록, train loss, validation loss가 감소, train accuracy와 validation accuracy가 증가함으로 효율의 증대가 추후 epoch에서 이루어 질 것으로 보고, 실 사용 모델은 epoch를 40번 한 모델을 사용

**Cautions**

# 학습시 keras,tensorflow 버전에 주의

ran out of data로 멈출 수 있음.

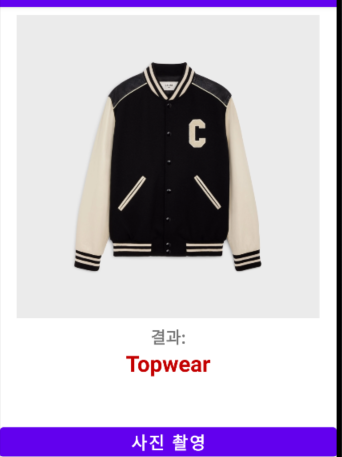
# tf.keras로 모델 구성 할 것

# 'Sequential' object has no attribute '\_get\_save\_spec' error 발생 가능

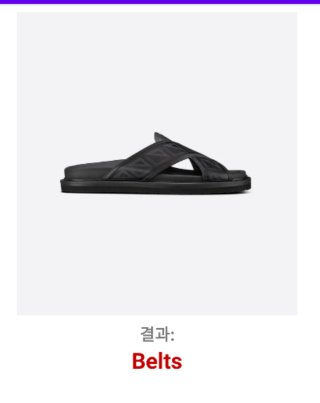
# **Prediction**

최종 모델의 Validation Accuracy : 89.62%

**Sample 어플 제작 후 테스트**



주변 환경 요소의 색상이 균일 하면, 상당한 정확도를 보임.

환경 요소의 색상이 균일 하지 않거나, 애매하면 잘못된 결과를 도출하기도 함.

**TensorflowLite**

converter = tf.lite.TFLiteConverter.from\_keras\_model(classifier)

tflite\_model = converter.convert()

with open("Clothing\_ClassifierTFlite\_model.tflite",'wb') as f:

f.write(tflite\_model)

해당 코드를 통해 학습된 모델을 TensorflowLite를 통해 앱에 내장 할 수 있었음.

서버 자원의 절약 및 속도 측면에서 사용 + 인터넷이 없어도 인식 가능

**References List**

KRIZHEVSKY, Alex; SUTSKEVER, Ilya; HINTON, Geoffrey E. Imagenet classification with deep convolutional neural networks. *Communications of the ACM*, 2017, 60.6: 84-90.

TensorflowLite - [Image Classification App | Teachable Machine + TensorFlow Lite](https://www.youtube.com/watch?v=jhGm4KDafKU&ab_channel=IJApps)

[](https://www.youtube.com/watch?v=jhGm4KDafKU&ab_channel=IJApps)

+ <https://github.com/IJ-Apps/Image-Classification-App-with-Custom-TensorFlow-Model>