통계학과 딥러닝

CNN을 활용한 마스크 착용 판별

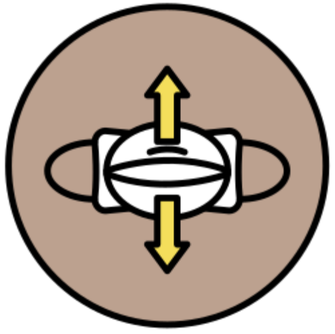
2018380508 김상진
2018380511 목진웅
2018380714 최동혁

00

목차

마스크 착용 판별 계획

1. 주제 선정 이유
2. 데이터 및 모델 설명
3. 연구 진행
4. Kaggle과 비교




01

주제 선정 이유


제대로 쓰지 않으면 예방효과가 없어요!
주의!
잘못된 마스크 착용,
안돼요!
배출
흡입
코와 입을 통해
자신과 타인에게
바이러스가
다 들어가요!
배출
흡입
❶ 코가 노출되는
마스크 착용
❷ 턱에 걸치는
마스크 착용


마스크를 착용하십시요.
GN CHECK
마스크를 착용하십시요.
GN CHECK
정상입니다.
GN CHECK
마스크 미착용
마스크 턱착용
마스크 착용

02

데이터 설명 - 1. Kaggle

데이터 출처
(유/무)

데이터 크기

kaggle

https://www.kaggle.com/datasets/tapakah68/medical-masks-part1?select=images

89.89GB

40.0k files

02

데이터 설명 - 1. Kaggle

- 3가지 타입 구분

TYPE 1 - 마스크를 완벽하게 착용한 경우

TYPE 2 - 마스크를 잘못 착용한 경우

TYPE 3 - 마스크를 미착용한 경우

02

데이터 설명 - 2. 새로 수집한 데이터

TYPE 1

TYPE 2

TYPE 3

02

데이터 설명 - 3. 투명 마스크

- 정확한 분류를 하지 못하는 사진

예측 값 - 마스크를 미착용한 경우

실제 값 - 마스크를 완벽하게 착용한 경우

TYPE 1

TYPE 2

TYPE 3

1. Kaggle

2. 수집 데이터

3. 투명 마스크

02

첫번째 모델

```
model_c1=Sequential()

model_c1.add(Conv2D(32,(3,3),activation='relu',input_shape=[299,299,3]))
model_c1.add(MaxPooling2D(2,2))

model_c1.add(Conv2D(64,(3,3),activation='relu',padding='same'))
model_c1.add(MaxPooling2D(2,2))

model_c1.add(Conv2D(128,(3,3),activation='relu',padding='same'))
model_c1.add(MaxPooling2D(2,2))

model_c1.add(Conv2D(256,(3,3),activation='relu',padding='same'))
model_c1.add(MaxPooling2D(2,2))

model_c1.add(Conv2D(512,(3,3),activation='relu',padding='same'))

model_c1.add(Flatten())
model_c1.add(Dense(128,activation='relu'))
model_c1.add(Dense(64,activation='relu'))
model_c1.add(Dense(3,activation='softmax'))
```

```
Model: "sequential_11"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_45 (Conv2D)           (None, 297, 297, 32)      896

max_pooling2d_31 (MaxPoolin  (None, 148, 148, 32)      0
g2D)

conv2d_46 (Conv2D)           (None, 148, 148, 64)      18496

max_pooling2d_32 (MaxPoolin  (None, 74, 74, 64)        0
g2D)

conv2d_47 (Conv2D)           (None, 74, 74, 128)       73856

max_pooling2d_33 (MaxPoolin  (None, 37, 37, 128)       0
g2D)

conv2d_48 (Conv2D)           (None, 37, 37, 256)       295168

max_pooling2d_34 (MaxPoolin  (None, 18, 18, 256)       0
g2D)

conv2d_49 (Conv2D)           (None, 18, 18, 512)       1180160

flatten_4 (Flatten)          (None, 165888)            0

dense_8 (Dense)              (None, 128)               21233792

dense_9 (Dense)              (None, 64)                8256

dense_10 (Dense)             (None, 3)                 195

=================================================================
Total params: 22,810,819
Trainable params: 22,810,819
Non-trainable params: 0
_________________________________________________________________
```

02

두번째 모델

```
m1=Sequential()
m1.add(Conv2D(32,(3,3),activation='relu',input_shape=[299,299,3]))
m1.add(AveragePooling2D(2,2))

m1.add(Conv2D(64,(3,3),activation='relu',padding='same'))
m1.add(AveragePooling2D(2,2))

m1.add(Conv2D(128,(3,3),activation='relu',padding='same'))
m1.add(AveragePooling2D(2,2))

m1.add(Conv2D(256,(3,3),activation='relu',padding='same'))
m1.add(MaxPooling2D(2,2))

m1.add(Conv2D(512,(3,3),activation='relu',padding='same'))
m1.add(MaxPooling2D(2,2))

m1.add(Conv2D(1024,(3,3),activation='relu',padding='same'))
m1.add(MaxPooling2D(2,2))

m1.add(Conv2D(2048,(3,3),activation='relu',padding='same'))

m1.add(Flatten())
m1.add(Dense(64,activation='relu'))
m1.add(Dense(3,activation='softmax'))
```

```
Model: "sequential_13"
_________________________________________________________________
 Layer (type)                 Output Shape              Param #
=================================================================
 conv2d_57 (Conv2D)           (None, 297, 297, 32)      896

 average_pooling2d_4 (Averag  (None, 148, 148, 32)      0
 ePooling2D)

 conv2d_58 (Conv2D)           (None, 148, 148, 64)      18496

 average_pooling2d_5 (Averag  (None, 74, 74, 64)        0
 ePooling2D)

 conv2d_59 (Conv2D)           (None, 74, 74, 128)       73856

 average_pooling2d_6 (Averag  (None, 37, 37, 128)       0
 ePooling2D)

 conv2d_60 (Conv2D)           (None, 37, 37, 256)       295168

 max_pooling2d_39 (MaxPoolin  (None, 18, 18, 256)       0
 g2D)

 conv2d_61 (Conv2D)           (None, 18, 18, 512)       1180160

 max_pooling2d_40 (MaxPoolin  (None, 9, 9, 512)         0
 g2D)

 conv2d_62 (Conv2D)           (None, 9, 9, 1024)        4719616

 max_pooling2d_41 (MaxPoolin  (None, 4, 4, 1024)        0
 g2D)

 conv2d_63 (Conv2D)           (None, 4, 4, 2048)        18876416

 flatten_6 (Flatten)          (None, 32768)             0

 dense_13 (Dense)             (None, 64)                2097216

 dense_14 (Dense)             (None, 3)                 195

=================================================================
Total params: 27,262,019
Trainable params: 27,262,019
Non-trainable params: 0
_________________________________________________________________
```

02

세번째 모델

```
m3=Sequential()
m3.add(Conv2D(32,(3,3),activation='relu',input_shape=[299,299,3]))
m3.add(MaxPooling2D(2,2))

m3.add(Conv2D(64,(3,3),activation='relu',padding='same'))
m3.add(MaxPooling2D(2,2))

m3.add(Conv2D(128,(3,3),activation='relu',padding='same'))
m3.add(MaxPooling2D(2,2))

m3.add(Conv2D(256,(3,3),activation='relu',padding='same'))
m3.add(MaxPooling2D(2,2))


m3.add(Conv2D(512,(3,3),activation='relu',padding='same'))
m3.add(MaxPooling2D(2,2))


m3.add(Conv2D(1024,(3,3),activation='relu',padding='same'))
m3.add(MaxPooling2D(2,2))

m3.add(Conv2D(2048,(3,3),activation='relu',padding='same'))

m3.add(Flatten())

m3.add(Dense(3,activation='softmax'))
```

```
Model: "sequential_15"
_________________________________________________________________
 Layer (type)                 Output Shape              Param #
=================================================================
 conv2d_71 (Conv2D)           (None, 297, 297, 32)      896

 max_pooling2d_48 (MaxPoolin  (None, 148, 148, 32)      0
 g2D)

 conv2d_72 (Conv2D)           (None, 148, 148, 64)      18496

 max_pooling2d_49 (MaxPoolin  (None, 74, 74, 64)        0
 g2D)

 conv2d_73 (Conv2D)           (None, 74, 74, 128)       73856

 max_pooling2d_50 (MaxPoolin  (None, 37, 37, 128)       0
 g2D)

 conv2d_74 (Conv2D)           (None, 37, 37, 256)       295168

 max_pooling2d_51 (MaxPoolin  (None, 18, 18, 256)       0
 g2D)

 conv2d_75 (Conv2D)           (None, 18, 18, 512)       1180160

 max_pooling2d_52 (MaxPoolin  (None, 9, 9, 512)         0
 g2D)

 conv2d_76 (Conv2D)           (None, 9, 9, 1024)        4719616

 max_pooling2d_53 (MaxPoolin  (None, 4, 4, 1024)        0
 g2D)

 conv2d_77 (Conv2D)           (None, 4, 4, 2048)        18876416

 flatten_8 (Flatten)          (None, 32768)             0

 dense_16 (Dense)             (None, 3)                 98307

=================================================================
Total params: 25,262,915
Trainable params: 25,262,915
Non-trainable params: 0
_________________________________________________________________
```

03

연구 진행

연구 진행 순서

- 1. Kaggle 데이터 모델 학습 및 과적합 개선
- 2. 수집 데이터 셋 테스트
- 3. 투명 마스크 셋 테스트
- 4. 투명 마스크 학습 및 테스트

1. Kaggle 데이터 모델 학습

데이터 만들기 훈련

In [78]:
```
training = pd.DataFrame(newdf[:5000]) # 전체 image데이터 셋에서 5000개의  training set
testing = pd.DataFrame(newdf[6000:]) # 전체 image데이터 셋에서 6000번째 부터 끝까지의 test set
```

In [79]:
```
x1=training[training['type']==1] # training set에서 타입 1,2,3,4를 1,2,3 으로 통합시킵니다.
x2=training[ (training['type']==2)| (training['type']==3)]
x3=training[training['type']==4]
```

In [80]:
```
x2['type']=2 # 2,3 타입의 데이터 셋은 'type'의 값을 2로 통일 시킵니다.
x3['type']=3 # type4는 'type'의 값을 3으로 바꿉니다.
```

In [81]:
```
x1=x1[:1000] # 통합시킨 타입 1,2,3 데이터 셋에서 각각 800개의 데이터만 가져옵니다.
x2=x2[:1000]
x3=x3[:1000]
```

In [82]:
```
# 각각 데이터를 XX1데이터 셋에 합칩니다
XX=pd.concat([x1[:1000],x2[:1000],x3[:1000]], ignore_index=True)
```

In [83]:
```
XX # xx의 데이터 형태입니다.
```

Out[83]:

	id	type	user_id	gender	age	name	size_mb
0	1	1	1	MALE	25	000001_1_000001_MALE_25.jpg	1.80
1	2	1	2	MALE	23	000002_1_000002_MALE_23.jpg	1.55
2	3	1	3	FEMALE	22	000003_1_000003_FEMALE_22.jpg	0.43
3	4	1	4	FEMALE	25	000004_1_000004_FEMALE_25.jpg	1.77
4	6	1	6	MALE	28	000006_1_000006_MALE_28.jpg	2.13
...	...	...	...	...	...	...	...
2995	1650	3	1650	MALE	31	001650_4_001650_MALE_31.jpg	2.51
2996	1653	3	1653	MALE	18	001653_4_001653_MALE_18.jpg	2.73
2997	1654	3	1654	MALE	20	001654_4_001654_MALE_20.jpg	0.49
2998	1655	3	1655	MALE	20	001655_4_001655_MALE_20.jpg	1.08
2999	1658	3	1658	MALE	22	001658_4_001658_MALE_22.jpg	2.28

3000 rows × 7 columns

	Training Accuracy	Test Accuracy
모델 1	0.9887	0.8094
모델 2	0.9703	0.8428
모델 3	0.9800	0.8595

1. 과적합 개선

```
model_c1_d=Sequential()

model_c1_d.add(Conv2D(32,(3,3),input_shape=[299,299,3]))
model_c1_d.add(BatchNormalization())
model_c1_d.add(Activation('relu'))
model_c1_d.add(MaxPooling2D(2,2))

model_c1_d.add(Conv2D(64,(3,3),padding='same'))
model_c1_d.add(BatchNormalization())
model_c1_d.add(Activation('relu'))
model_c1_d.add(MaxPooling2D(2,2))



model_c1_d.add(Conv2D(128,(3,3),padding='same'))
model_c1_d.add(BatchNormalization())
model_c1_d.add(Activation('relu'))
model_c1_d.add(MaxPooling2D(2,2))


model_c1_d.add(Conv2D(256,(3,3),padding='same'))
model_c1_d.add(BatchNormalization())
model_c1_d.add(Activation('relu'))
model_c1_d.add(MaxPooling2D(2,2))

model_c1_d.add(Conv2D(512,(3,3),padding='same'))
model_c1_d.add(BatchNormalization())
model_c1_d.add(Activation('relu'))

model_c1_d.add(Flatten())
model_c1_d.add(Dropout(0.2))
model_c1_d.add(Dense(128,activation='relu'))
model_c1_d.add(Dropout(0.2))
model_c1_d.add(Dense(64,activation='relu'))
model_c1_d.add(Dropout(0.2))
model_c1_d.add(Dense(3,activation='softmax'))
```

```
max_pooling2d_13 (MaxPoolin  (None, 18, 18, 256)      0
g2D)

conv2d_18 (Conv2D)           (None, 18, 18, 512)      1180160

batch_normalization_10 (Bat  (None, 18, 18, 512)      2048
chNormalization)

activation_9 (Activation)    (None, 18, 18, 512)      0

flatten_2 (Flatten)          (None, 165888)           0

dropout_3 (Dropout)          (None, 165888)           0

dense_4 (Dense)              (None, 128)              21233792

dropout_4 (Dropout)          (None, 128)              0

dense_5 (Dense)              (None, 64)               8256

dropout_5 (Dropout)          (None, 64)               0

dense_6 (Dense)              (None, 3)                195

=================================================================
Total params: 22,814,787
Trainable params: 22,812,803
Non-trainable params: 1,984
```

03

1. 과적합 개선

```
m1_d.add(Conv2D(32,(3,3),input_shape=[299,299,3]))
m1_d.add(BatchNormalization())
m1_d.add(Activation('relu'))
m1_d.add(AveragePooling2D(2,2))

m1_d.add(Conv2D(64,(3,3), padding='same')
m1_d.add(BatchNormalization())
m1_d.add(Activation('relu'))
m1_d.add(AveragePooling2D(2,2))

m1_d.add(Conv2D(128,(3,3), padding='same')
m1_d.add(BatchNormalization())
m1_d.add(Activation('relu'))
m1_d.add(AveragePooling2D(2,2))

m1_d.add(Conv2D(256,(3,3), padding='same')
m1_d.add(BatchNormalization())
m1_d.add(Activation('relu'))
m1_d.add(MaxPooling2D(2,2))


m1_d.add(Conv2D(512,(3,3), padding='same')
m1_d.add(BatchNormalization())
m1_d.add(Activation('relu'))
m1_d.add(MaxPooling2D(2,2))


m1_d.add(Conv2D(1024,(3,3), padding='same')
m1_d.add(BatchNormalization())
m1_d.add(Activation('relu'))
m1_d.add(MaxPooling2D(2,2))


m1_d.add(Conv2D(2048,(3,3),padding='same'))
m1_d.add(BatchNormalization())
m1_d.add(Activation('relu'))

m1_d.add(Flatten())
m1_d.add(Dropout(0.2))
m1_d.add(Dense(64,activation='relu'))
m1_d.add(Dropout(0.2))
m1_d.add(Dense(3,activation='softmax'))
```

```
batch_normalization_16 (Bat  (None, 18, 18, 512)      2048
chNormalization)

activation_15 (Activation)   (None, 18, 18, 512)      0

max_pooling2d_15 (MaxPoolin  (None, 9, 9, 512)        0
g2D)

conv2d_25 (Conv2D)           (None, 9, 9, 1024)       4719616

batch_normalization_17 (Bat  (None, 9, 9, 1024)       4096
chNormalization)

activation_16 (Activation)   (None, 9, 9, 1024)       0

max_pooling2d_16 (MaxPoolin  (None, 4, 4, 1024)       0
g2D)

conv2d_26 (Conv2D)           (None, 4, 4, 2048)       18876416

batch_normalization_18 (Bat  (None, 4, 4, 2048)       8192
chNormalization)

activation_17 (Activation)   (None, 4, 4, 2048)       0

flatten_3 (Flatten)          (None, 32768)            0

dropout_6 (Dropout)          (None, 32768)            0

dense_7 (Dense)              (None, 64)               2097216

dropout_7 (Dropout)          (None, 64)               0

dense_8 (Dense)              (None, 3)                195

=================================================================
Total params: 27,278,275
Trainable params: 27,270,147
Non-trainable params: 8,128
```

1. 과적합 개선

```
m3_d=Sequential()

m3_d.add(Conv2D(32,(3,3),input_shape=[299,299,3]))
m3_d.add(BatchNormalization())
m3_d.add(Activation('relu'))
m3_d.add(MaxPooling2D(2,2))

m3_d.add(Conv2D(64,(3,3), padding='same'))
m3_d.add(BatchNormalization())
m3_d.add(Activation('relu'))

m3_d.add(MaxPooling2D(2,2))

m3_d.add(Conv2D(128,(3,3), padding='same'))
m3_d.add(BatchNormalization())
m3_d.add(Activation('relu'))
m3_d.add(MaxPooling2D(2,2))

m3_d.add(Conv2D(256,(3,3), padding='same'))
m3_d.add(BatchNormalization())
m3_d.add(Activation('relu'))
m3_d.add(MaxPooling2D(2,2))


m3_d.add(Conv2D(512,(3,3), padding='same'))
m3_d.add(BatchNormalization())
m3_d.add(Activation('relu'))
m3_d.add(MaxPooling2D(2,2))

m3_d.add(Conv2D(1024,(3,3), padding='same'))
m3_d.add(BatchNormalization())
m3_d.add(Activation('relu'))
m3_d.add(MaxPooling2D(2,2))

m3_d.add(Conv2D(2048,(3,3), padding='same'))
m3_d.add(BatchNormalization())
m3_d.add(Activation('relu'))

m3_d.add(Flatten())
m3_d.add(Dropout(0.2))
m3_d.add(Dense(3,activation='softmax'))
```

```
conv2d_31 (Conv2D)           (None, 18, 18, 512)       1180160

batch_normalization_23 (Bat  (None, 18, 18, 512)       2048
chNormalization)

activation_22 (Activation)   (None, 18, 18, 512)       0

max_pooling2d_21 (MaxPoolin  (None, 9, 9, 512)         0
g2D)

conv2d_32 (Conv2D)           (None, 9, 9, 1024)        4719616

batch_normalization_24 (Bat  (None, 9, 9, 1024)        4096
chNormalization)

activation_23 (Activation)   (None, 9, 9, 1024)        0

max_pooling2d_22 (MaxPoolin  (None, 4, 4, 1024)        0
g2D)

conv2d_33 (Conv2D)           (None, 4, 4, 2048)        18876416

batch_normalization_25 (Bat  (None, 4, 4, 2048)        8192
chNormalization)

activation_24 (Activation)   (None, 4, 4, 2048)        0

flatten_4 (Flatten)          (None, 32768)             0

dropout_8 (Dropout)          (None, 32768)             0

dense_9 (Dense)              (None, 3)                 98307

=================================================================
Total params: 25,279,171
Trainable params: 25,271,043
Non-trainable params: 8,128
```

1. 과적합 개선

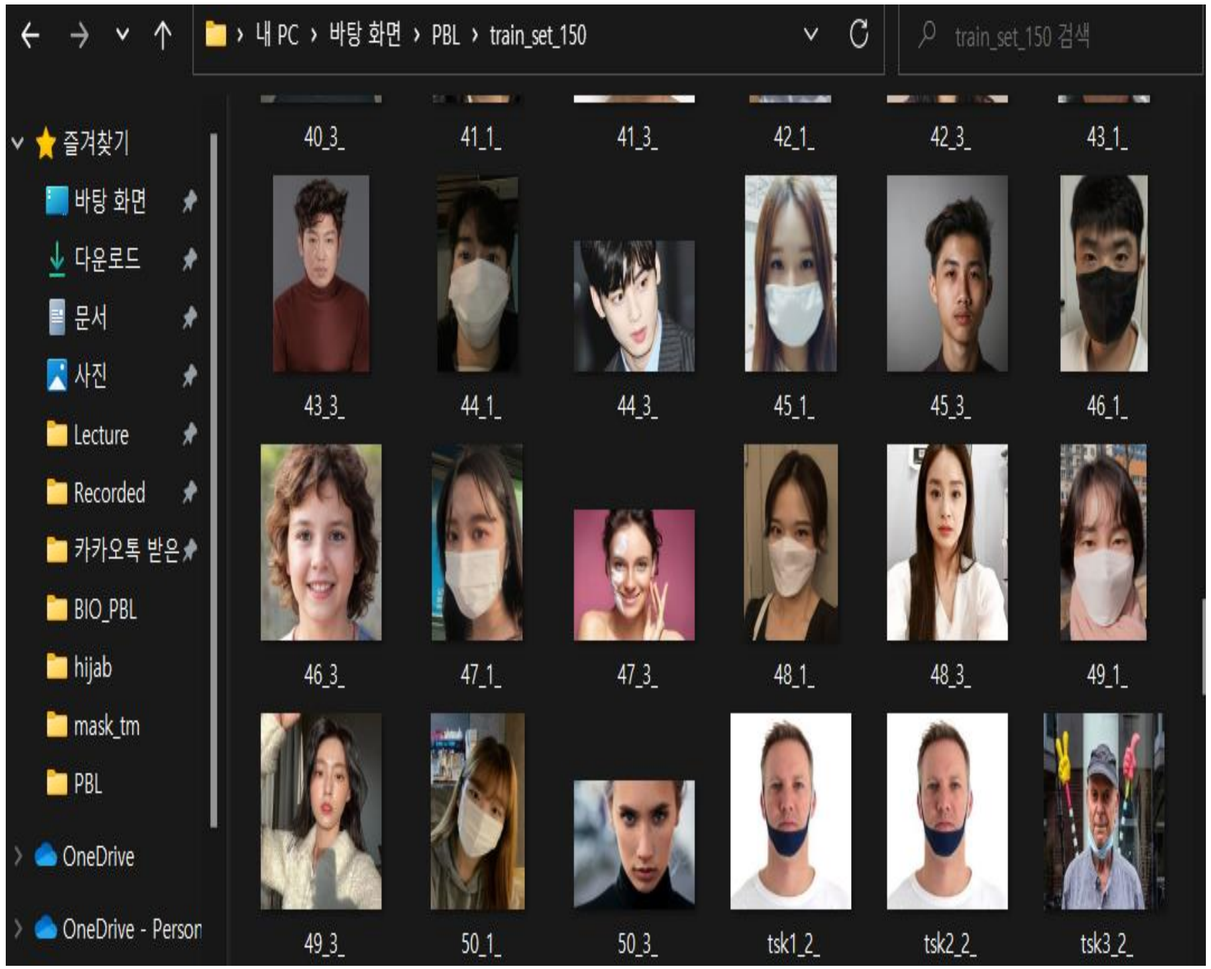
과적합 개선 전

	Training Accuracy	Test Accuracy
모델 1	0.9887	0.8094
모델 2	0.9703	0.8428
모델 3	0.9800	0.8595

과적합 개선 후

	Training Acc	Testing Acc
모델 1	0.8573	0.8214
모델 2	0.8820	0.8428
모델 3	0.9052	0.8741

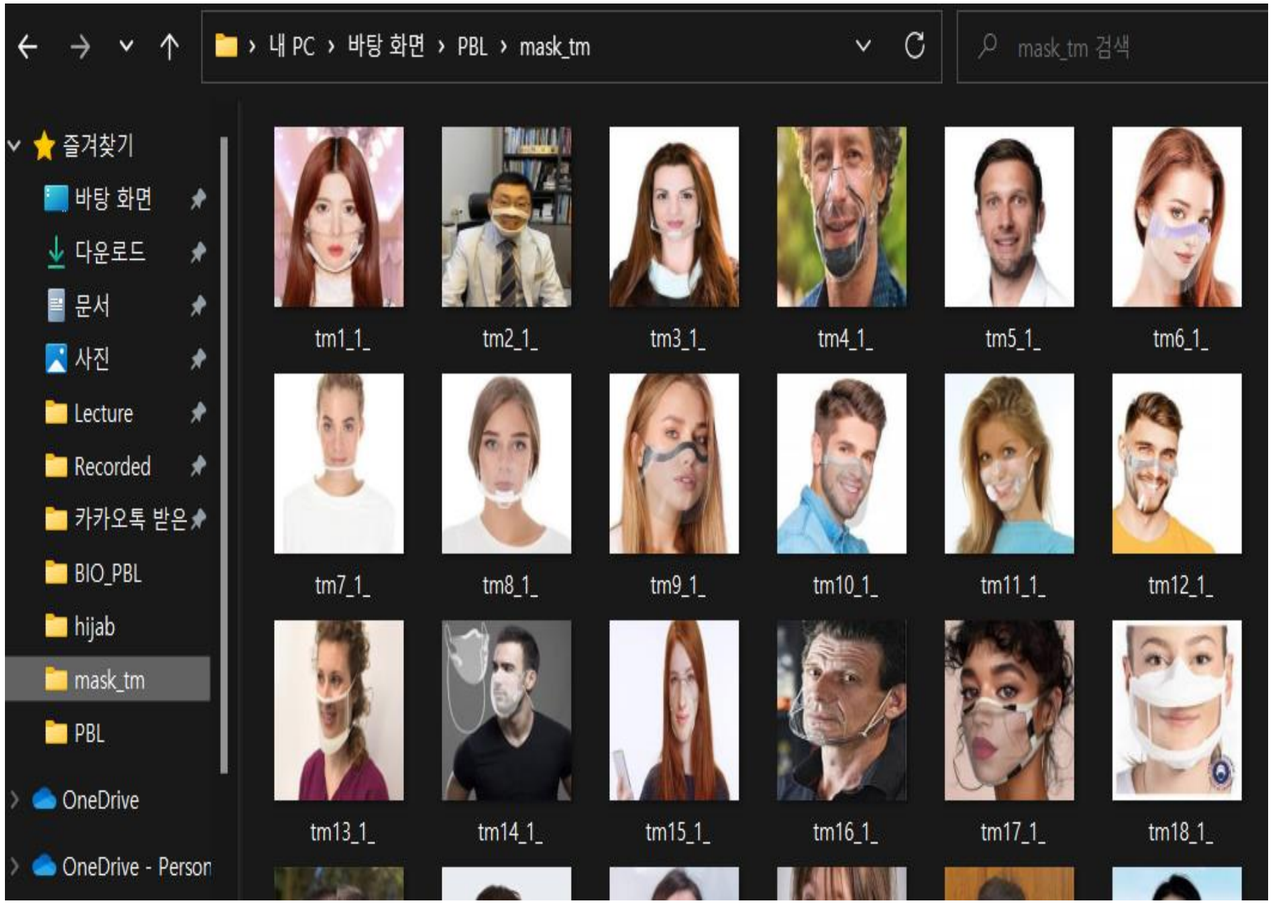
2. 수집 데이터 셋 테스트



	Testing Acc
모델 1	0.7545
모델 2	0.8293
모델 3	0.8538

03

3. 투명 마스크 셋 테스트



	Testing Acc
모델 1	0.2023
모델 2	0.5096
모델 3	0.5926

03

3. 투명 마스크 셋 테스트

```
# 이미지 증대를 위한 Image Generator

from keras.preprocessing.image import ImageDataGenerator


train_tm_datagen = ImageDataGenerator(rescale = 1/255.0,
                                      rotation_range=40,
                                      width_shift_range=0.2,
                                      height_shift_range = 0.2,
                                      shear_range = 0.2,
                                      zoom_range = 0.2,
                                      horizontal_flip = True,
                                      fill_mode = 'nearest')

train_tm_gen= train_tm_datagen.flow(x=X_tm_gen,y=Y_tm_gen)
```

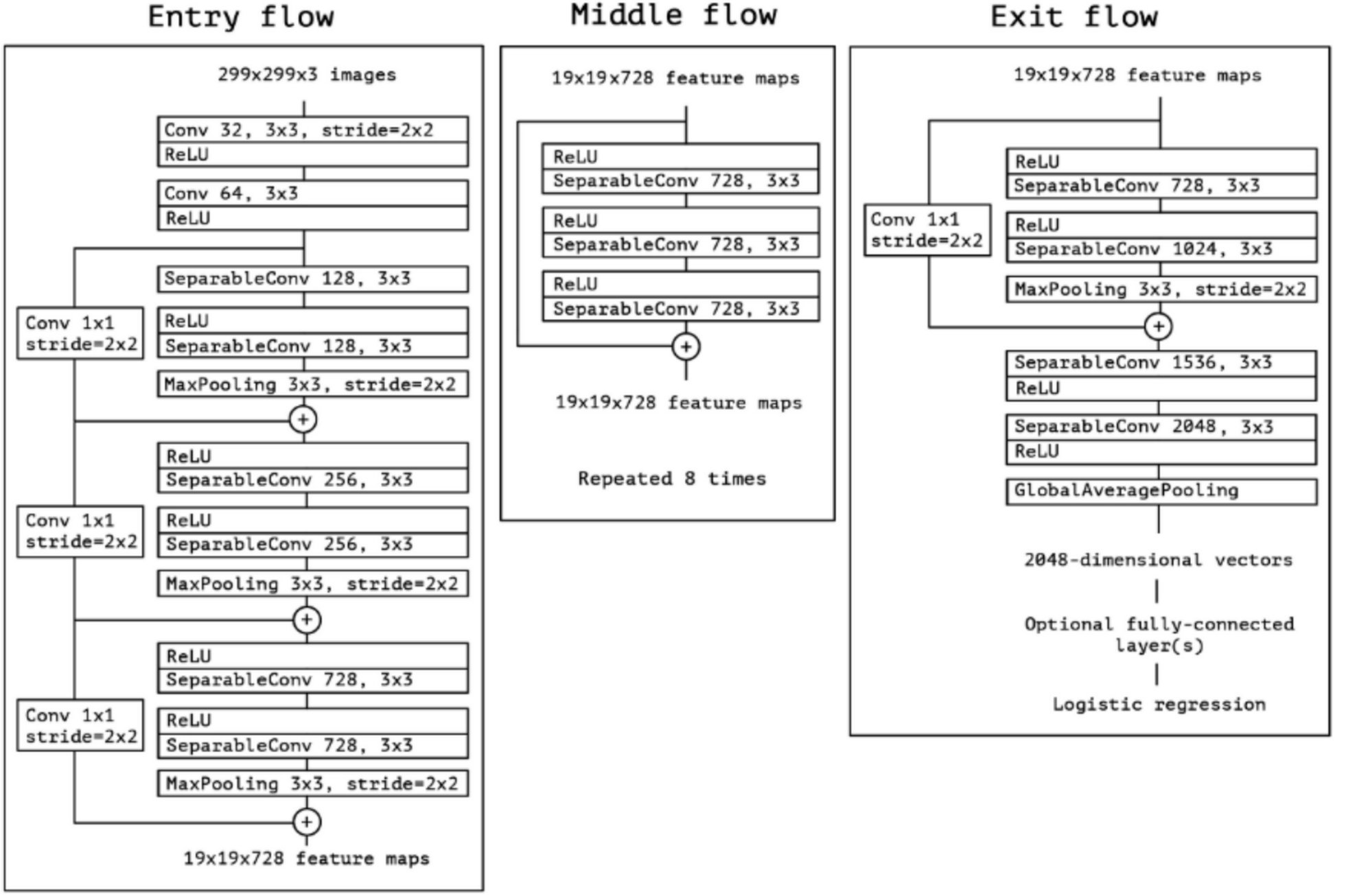
```
augs = train_tm_gen.__getitem__(2)

plt.figure(figsize=(16, 8))
for i, img in enumerate(augs[0]):
    plt.subplot(4, 8, i+1)
    plt.axis('off')
    img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
    plt.imshow(img.squeeze())
```

3. 투명 마스크 셋 테스트

	Testing Acc
모델 1	0.9744
모델 2	0.9744
모델 3	0.9744

02

Xception model



03 Xception 모델과 비교

```
import tensorflow as tf
import keras
from keras.applications.xception import Xception
model=Xception(include_top = False, weights = 'imagenet', input_shape = (299,299,3))
model.summary()
flattened = tf.keras.layers.Flatten()(model.output)

fc1 = tf.keras.layers.Dense(3, activation='softmax', name="AddedDense2")(flattened)

model = tf.keras.models.Model(inputs=model.input, outputs=fc1)
```

```
 block14_sepconv2 (SeparableCon  (None, 10, 10, 2048  3159552    ['block14_sepconv1_act[0][0]']
 v2D)                           )

 block14_sepconv2_bn (BatchNorm  (None, 10, 10, 2048  8192       ['block14_sepconv2[0][0]']
 alization)                     )

 block14_sepconv2_act (Activati  (None, 10, 10, 2048  0          ['block14_sepconv2_bn[0][0]']
 on)                            )

 flatten_17 (Flatten)           (None, 204800)       0           ['block14_sepconv2_act[0][0]']

 AddedDense2 (Dense)            (None, 3)            614403      ['flatten_17[0][0]']

==================================================================================================
Total params: 21,475,883
Trainable params: 21,421,355
Non-trainable params: 54,528
__________________________________________________________________________________________________
None
```

03

Xception 모델과 비교

	Model_3	Xception
Kaggle Training Acc	0.9052	0.9957
Kaggle Testing Acc	0.8741	0.9765
Self Testing Acc	0.8538	0.9599
TM Acc	0.5926	0.6369
TM Testing Acc	0.9744	0.9744

감사합니다.

통계학과 딥러닝 PBL_CNN을 이용한 마스크 착용 판별

3조