KT GenieLabs Dev-Challenge 2022

과제3 음식 이미지 분류 모델 설계

SDA(Statistics Data Analysis)

경상국립대학교 정보통계학과 허지혜
경상국립대학교 정보통계학과 성언승

Contents

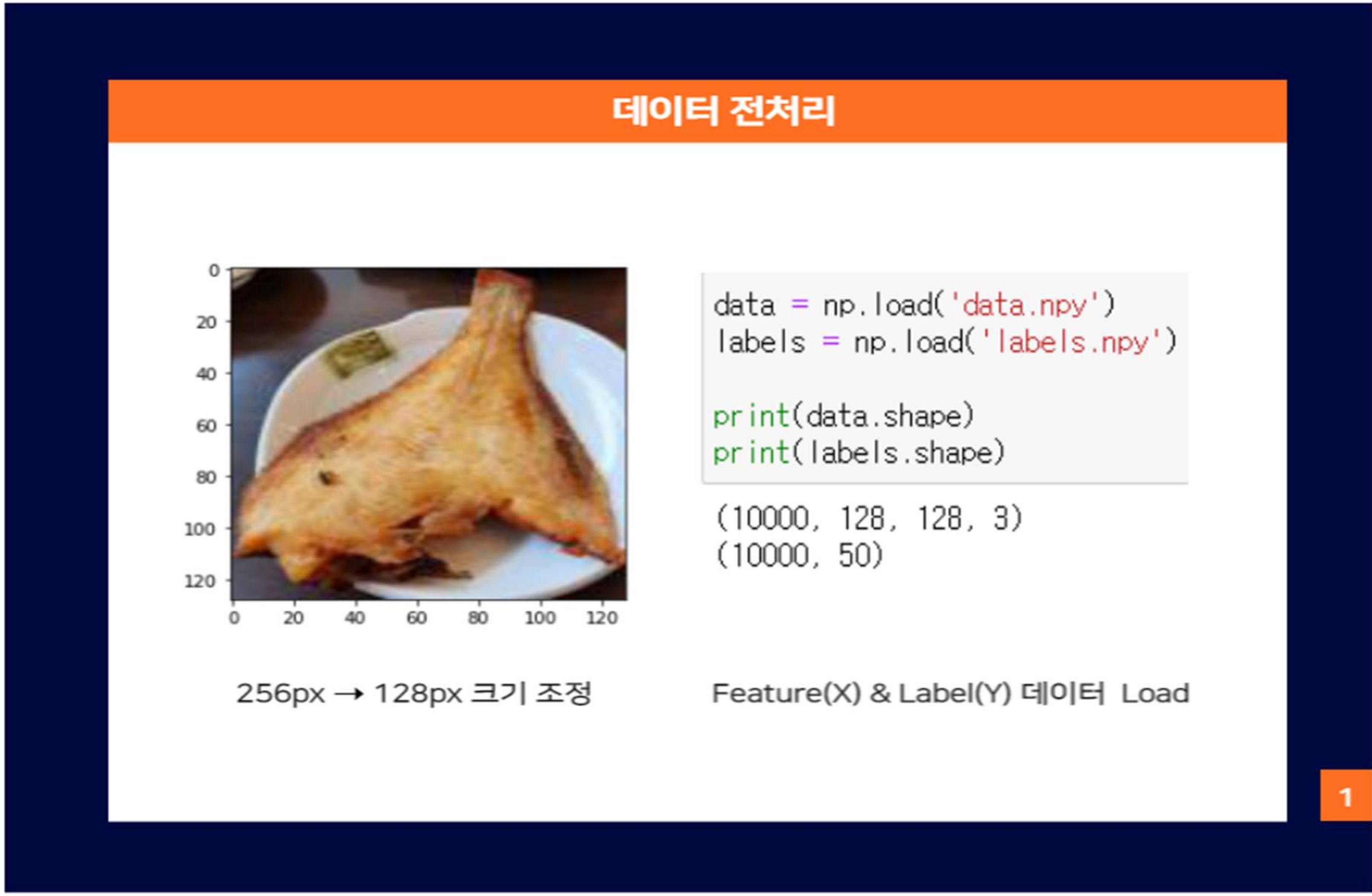
1. 프로젝트 수행방안

2. 제안 모델

3. 하이퍼 파라미터

4. 분류 결과

프로젝트 수행방안



데이터 전처리

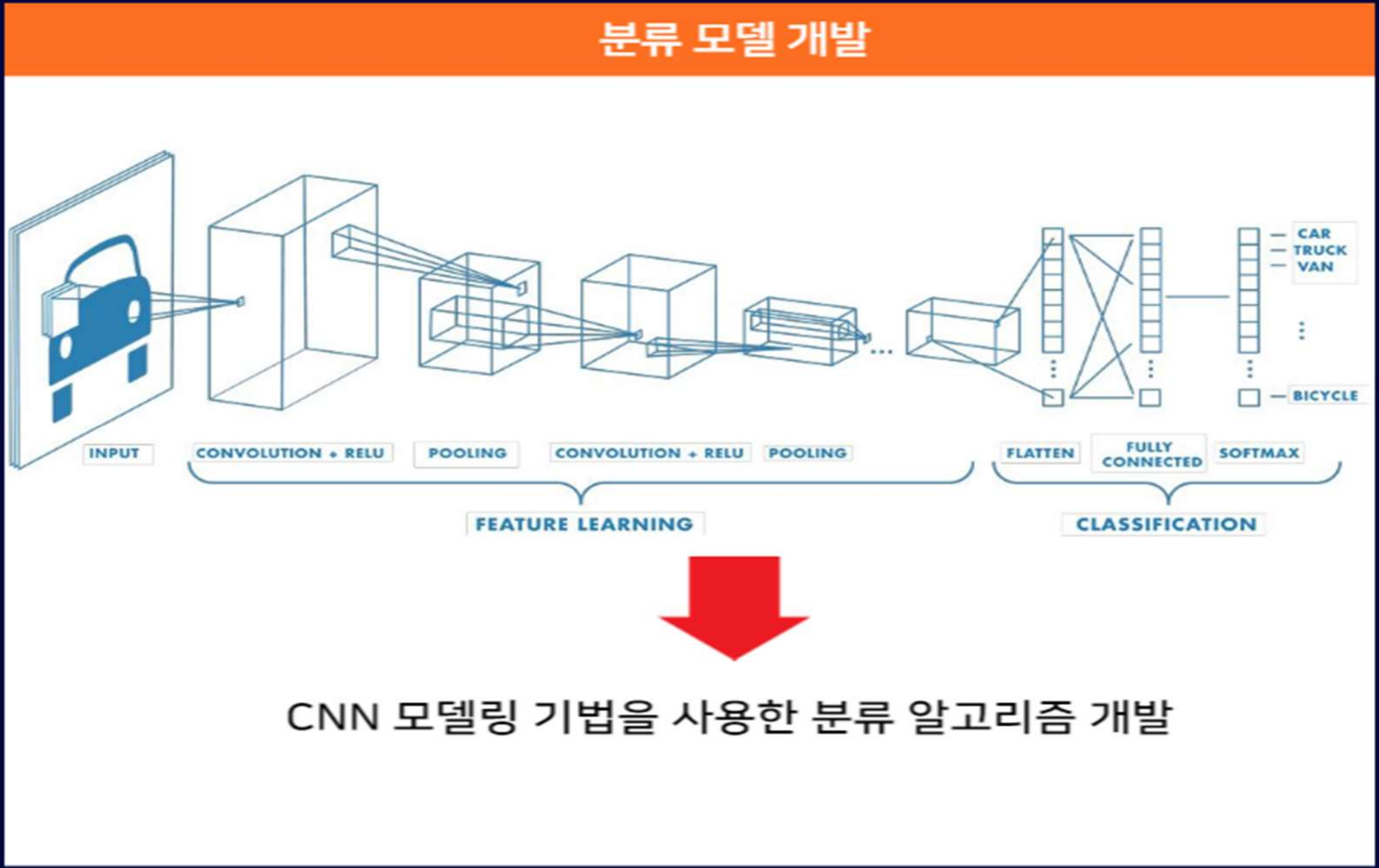
```
data = np.load('data.npy')
labels = np.load('labels.npy')

print(data.shape)
print(labels.shape)
```

```
(10000, 128, 128, 3)
(10000, 50)
```

256px → 128px 크기 조정

Feature(X) & Label(Y) 데이터 Load

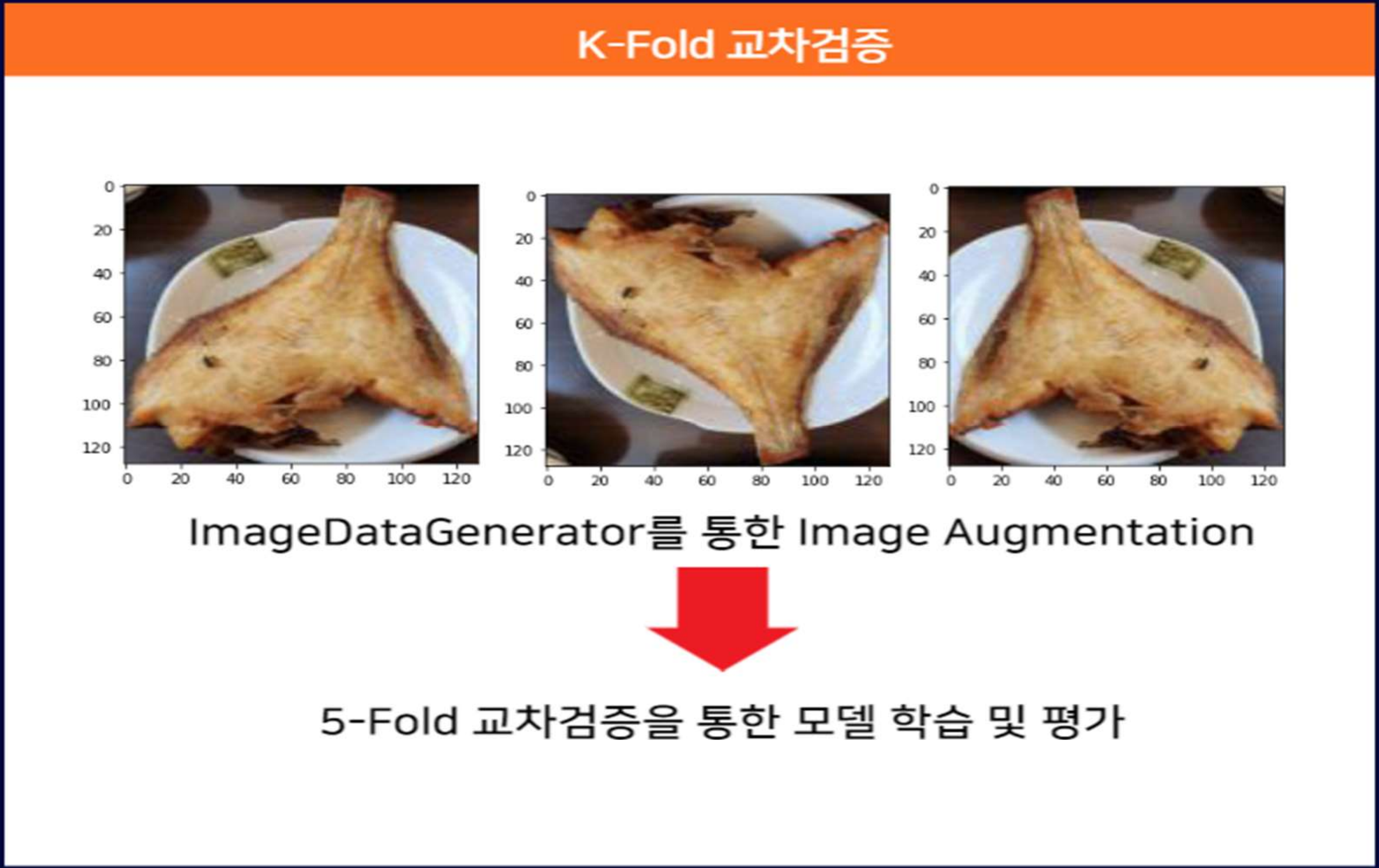
프로젝트 수행방안



분류 모델 개발

CNN 모델링 기법을 사용한 분류 알고리즘 개발

프로젝트 수행방안

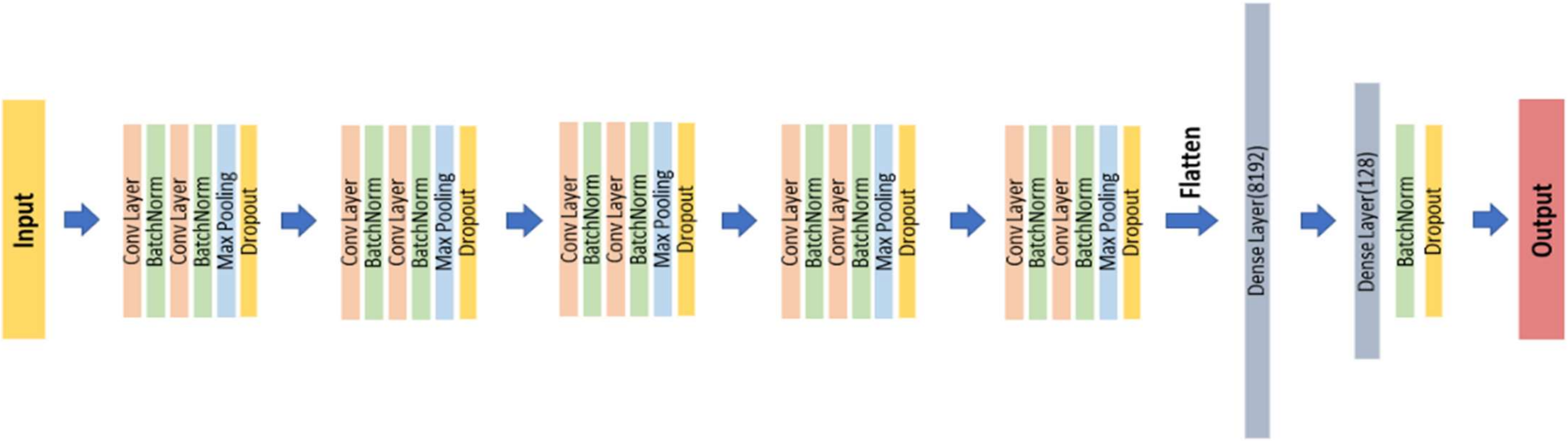


K-Fold 교차검증

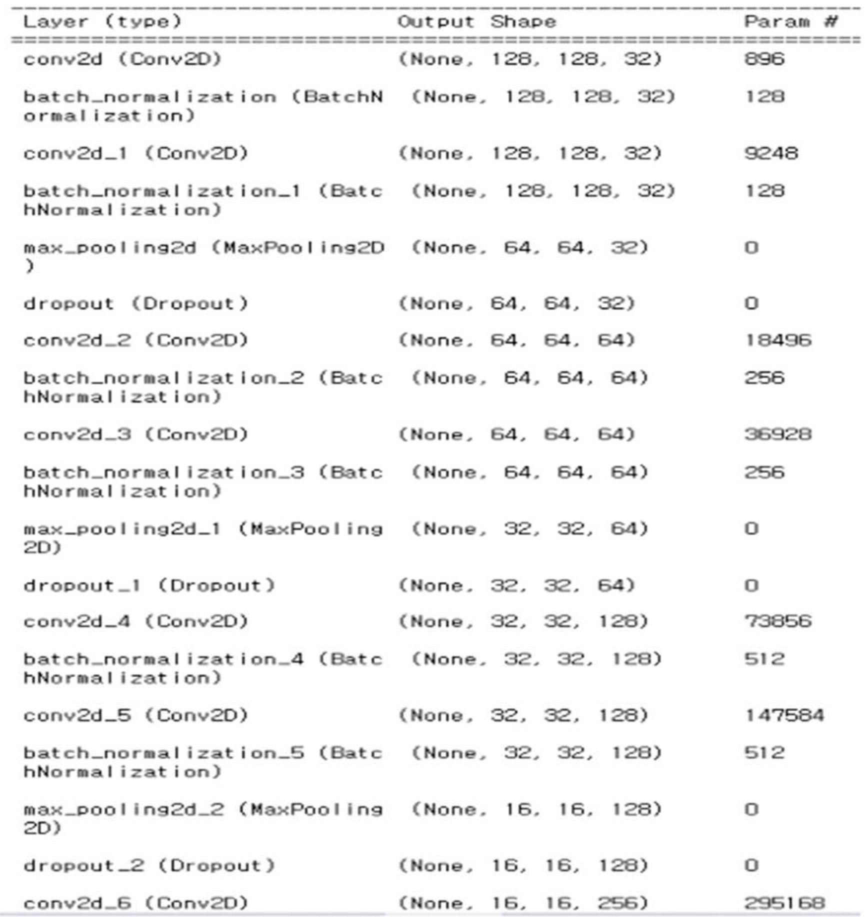
ImageDataGenerator를 통한 Image Augmentation

5-Fold 교차검증을 통한 모델 학습 및 평가

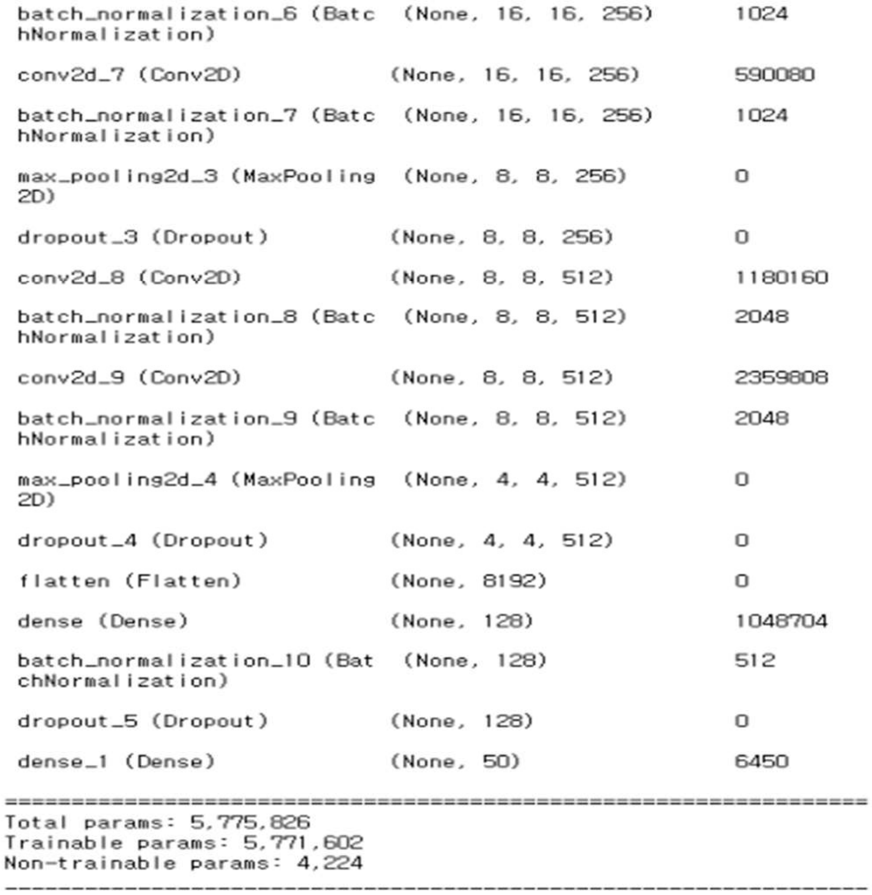
제안 모델


Input
Conv Layer
BatchNorm
Conv Layer
BatchNorm
Max Pooling
Dropout
Conv Layer
BatchNorm
Conv Layer
BatchNorm
Max Pooling
Dropout
Conv Layer
BatchNorm
Conv Layer
BatchNorm
Max Pooling
Dropout
Conv Layer
BatchNorm
Conv Layer
BatchNorm
Max Pooling
Dropout
Conv Layer
BatchNorm
Conv Layer
BatchNorm
Max Pooling
Dropout
Flatten
Dense Layer(8192)
Dense Layer(128)
BatchNorm
Dropout
Output

제안 모델

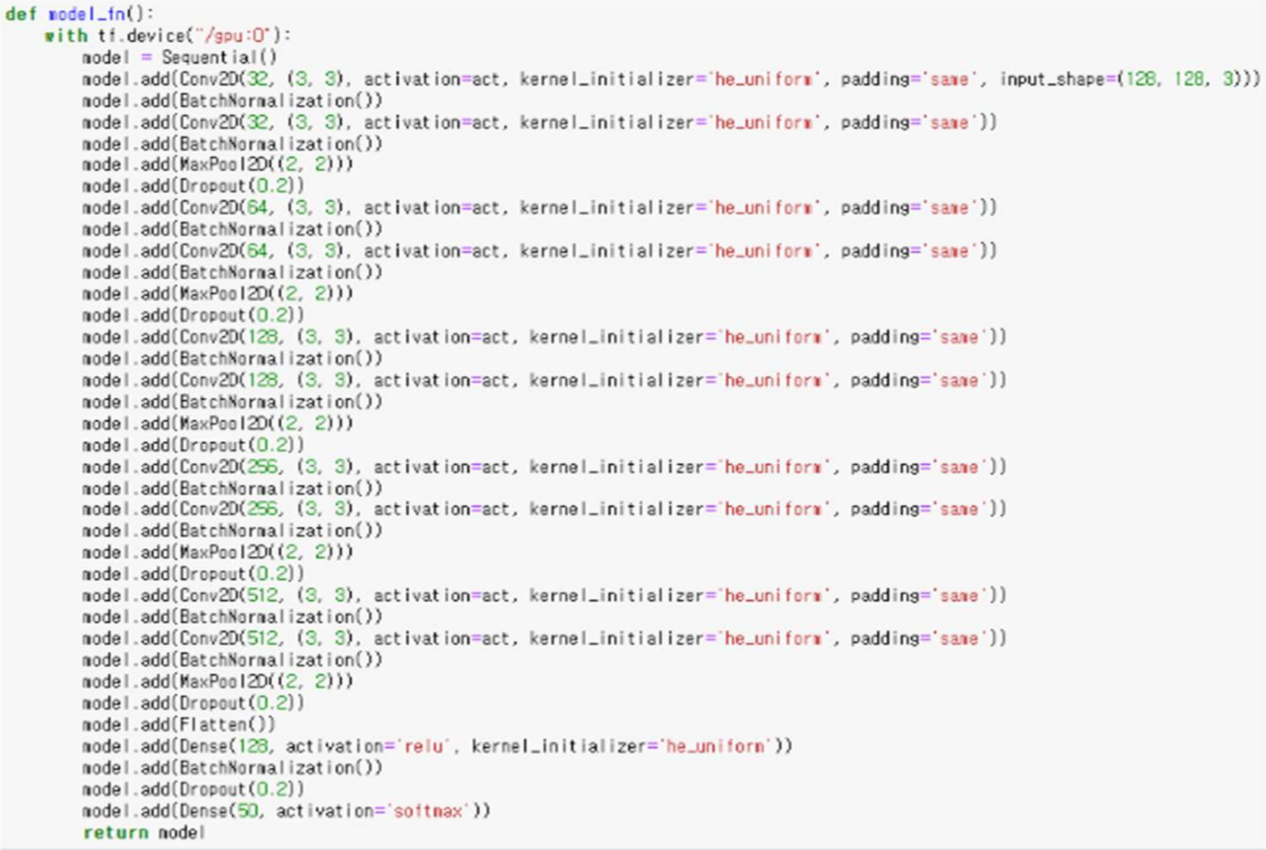


```
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 128, 128, 32)      896

 batch_normalization (BatchN  (None, 128, 128, 32)     128
 ormalization)

 conv2d_1 (Conv2D)           (None, 128, 128, 32)      9248

 batch_normalization_1 (Batc  (None, 128, 128, 32)     128
 hNormalization)

 max_pooling2d (MaxPooling2D  (None, 64, 64, 32)       0
 )

 dropout (Dropout)           (None, 64, 64, 32)        0

 conv2d_2 (Conv2D)           (None, 64, 64, 64)        18496

 batch_normalization_2 (Batc  (None, 64, 64, 64)       256
 hNormalization)

 conv2d_3 (Conv2D)           (None, 64, 64, 64)        36928

 batch_normalization_3 (Batc  (None, 64, 64, 64)       256
 hNormalization)

 max_pooling2d_1 (MaxPooling  (None, 32, 32, 64)       0
 2D)

 dropout_1 (Dropout)         (None, 32, 32, 64)        0

 conv2d_4 (Conv2D)           (None, 32, 32, 128)       73856

 batch_normalization_4 (Batc  (None, 32, 32, 128)      512
 hNormalization)

 conv2d_5 (Conv2D)           (None, 32, 32, 128)       147584

 batch_normalization_5 (Batc  (None, 32, 32, 128)      512
 hNormalization)

 max_pooling2d_2 (MaxPooling  (None, 16, 16, 128)      0
 2D)

 dropout_2 (Dropout)         (None, 16, 16, 128)       0

 conv2d_6 (Conv2D)           (None, 16, 16, 256)       295168
```



```
 batch_normalization_6 (Batc  (None, 16, 16, 256)      1024
 hNormalization)

 conv2d_7 (Conv2D)           (None, 16, 16, 256)       590080

 batch_normalization_7 (Batc  (None, 16, 16, 256)      1024
 hNormalization)

 max_pooling2d_3 (MaxPooling  (None, 8, 8, 256)        0
 2D)

 dropout_3 (Dropout)         (None, 8, 8, 256)         0

 conv2d_8 (Conv2D)           (None, 8, 8, 512)         1180160

 batch_normalization_8 (Batc  (None, 8, 8, 512)        2048
 hNormalization)

 conv2d_9 (Conv2D)           (None, 8, 8, 512)         2359808

 batch_normalization_9 (Batc  (None, 8, 8, 512)        2048
 hNormalization)

 max_pooling2d_4 (MaxPooling  (None, 4, 4, 512)        0
 2D)

 dropout_4 (Dropout)         (None, 4, 4, 512)         0

 flatten (Flatten)           (None, 8192)              0

 dense (Dense)               (None, 128)               1048704

 batch_normalization_10 (Bat  (None, 128)              512
 chNormalization)

 dropout_5 (Dropout)         (None, 128)               0

 dense_1 (Dense)             (None, 50)                6450

=================================================================
Total params: 5,775,826
Trainable params: 5,771,602
Non-trainable params: 4,224
_________________________________________________________________
```

제안 모델

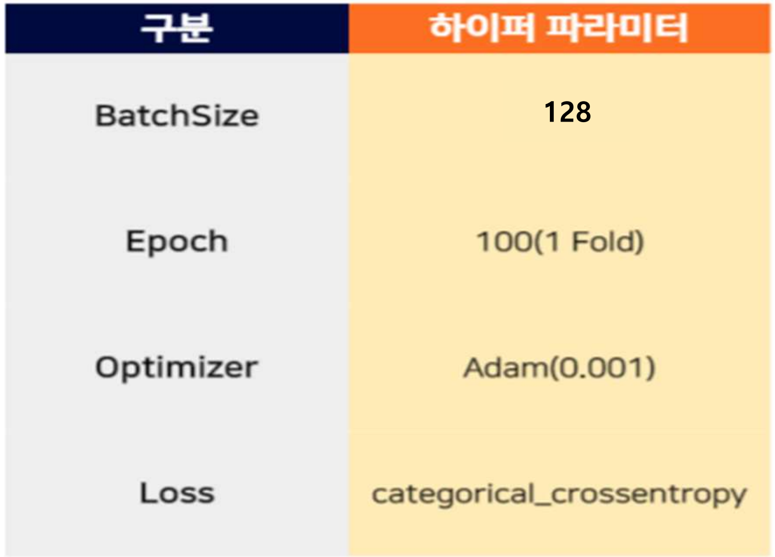


```python
def model_fn():
    with tf.device("/gpu:0"):
        model = Sequential()
        model.add(Conv2D(32, (3, 3), activation=act, kernel_initializer='he_uniform', padding='same', input_shape=(128, 128, 3)))
        model.add(BatchNormalization())
        model.add(Conv2D(32, (3, 3), activation=act, kernel_initializer='he_uniform', padding='same'))
        model.add(BatchNormalization())
        model.add(MaxPool2D((2, 2)))
        model.add(Dropout(0.2))
        model.add(Conv2D(64, (3, 3), activation=act, kernel_initializer='he_uniform', padding='same'))
        model.add(BatchNormalization())
        model.add(Conv2D(64, (3, 3), activation=act, kernel_initializer='he_uniform', padding='same'))
        model.add(BatchNormalization())
        model.add(MaxPool2D((2, 2)))
        model.add(Dropout(0.2))
        model.add(Conv2D(128, (3, 3), activation=act, kernel_initializer='he_uniform', padding='same'))
        model.add(BatchNormalization())
        model.add(Conv2D(128, (3, 3), activation=act, kernel_initializer='he_uniform', padding='same'))
        model.add(BatchNormalization())
        model.add(MaxPool2D((2, 2)))
        model.add(Dropout(0.2))
        model.add(Conv2D(256, (3, 3), activation=act, kernel_initializer='he_uniform', padding='same'))
        model.add(BatchNormalization())
        model.add(Conv2D(256, (3, 3), activation=act, kernel_initializer='he_uniform', padding='same'))
        model.add(BatchNormalization())
        model.add(MaxPool2D((2, 2)))
        model.add(Dropout(0.2))
        model.add(Conv2D(512, (3, 3), activation=act, kernel_initializer='he_uniform', padding='same'))
        model.add(BatchNormalization())
        model.add(Conv2D(512, (3, 3), activation=act, kernel_initializer='he_uniform', padding='same'))
        model.add(BatchNormalization())
        model.add(MaxPool2D((2, 2)))
        model.add(Dropout(0.2))
        model.add(Flatten())
        model.add(Dense(128, activation='relu', kernel_initializer='he_uniform'))
        model.add(BatchNormalization())
        model.add(Dropout(0.2))
        model.add(Dense(50, activation='softmax'))
        return model
```

모델 특징

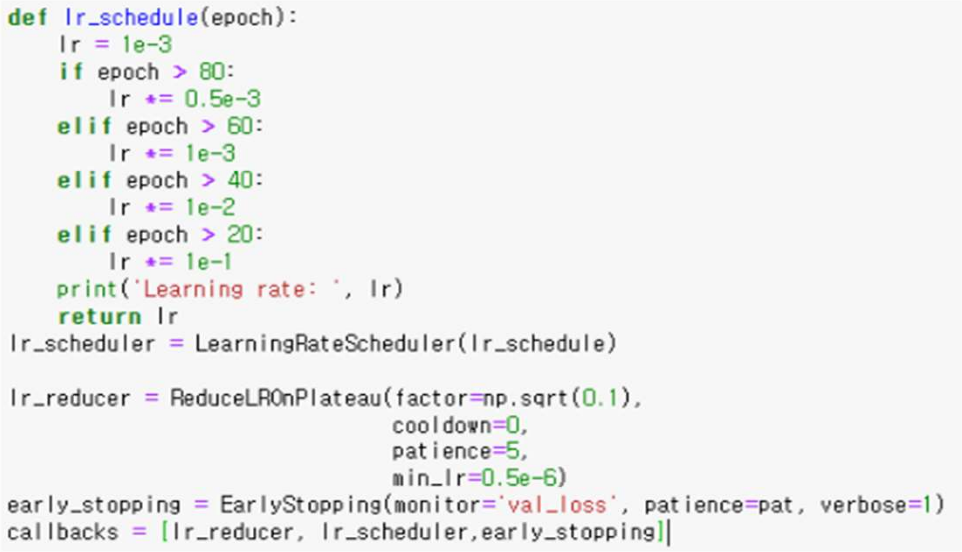
1. VGG16 모델 Fine-Tuning → 후반부 층 간소화
2. 활성화 함수 변경 → LeackyReLU(alpha=0.2)
3. 각 컨볼루션 층 → He 초기화(he_uniform) 사용
4. 각 Block별 Dropout을 통한 과적합 방지

하이퍼 파라미터



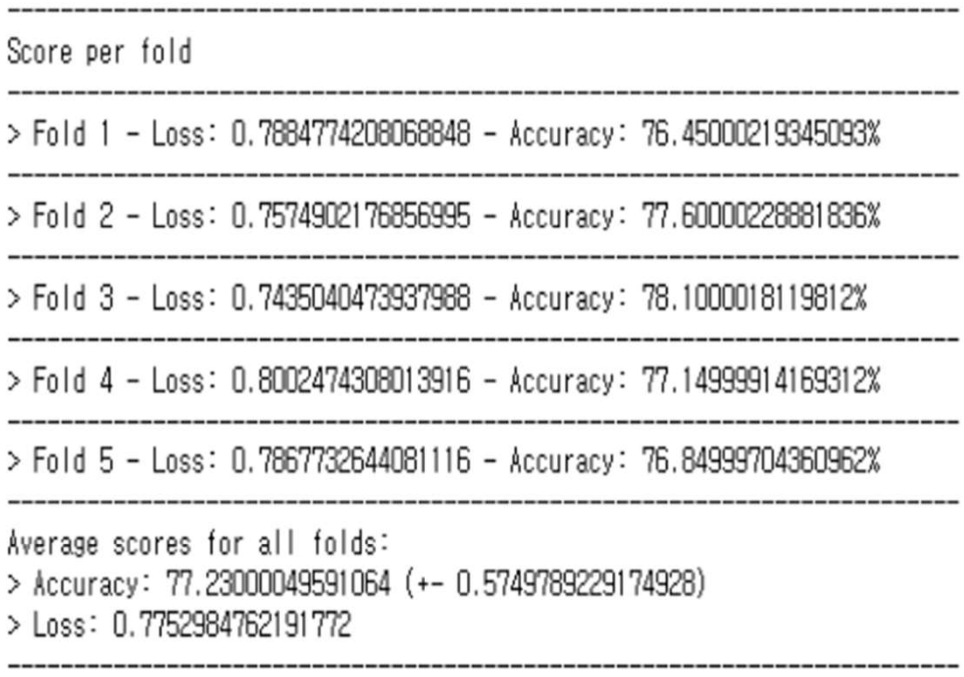
구분	하이퍼 파라미터
BatchSize	128
Epoch	100(1 Fold)
Optimizer	Adam(0.001)
Loss	categorical_crossentropy

CallBack 함수 정의



```
def lr_schedule(epoch):
    lr = 1e-3
    if epoch > 80:
        lr *= 0.5e-3
    elif epoch > 60:
        lr *= 1e-3
    elif epoch > 40:
        lr *= 1e-2
    elif epoch > 20:
        lr *= 1e-1
    print('Learning rate: ', lr)
    return lr
lr_scheduler = LearningRateScheduler(lr_schedule)

lr_reducer = ReduceLROnPlateau(factor=np.sqrt(0.1),
                               cooldown=0,
                               patience=5,
                               min_lr=0.5e-6)
early_stopping = EarlyStopping(monitor='val_loss', patience=pat, verbose=1)
callbacks = [lr_reducer, lr_scheduler,early_stopping]
```

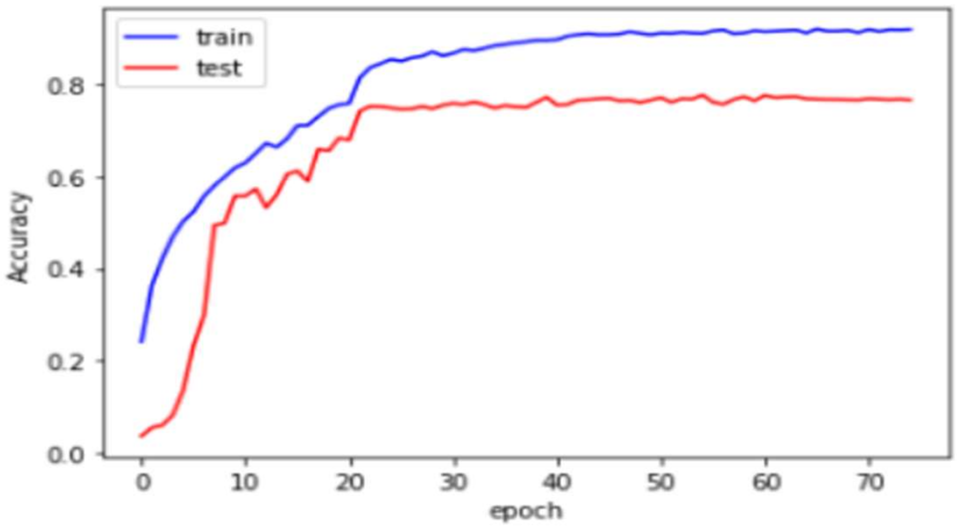
분류 결과

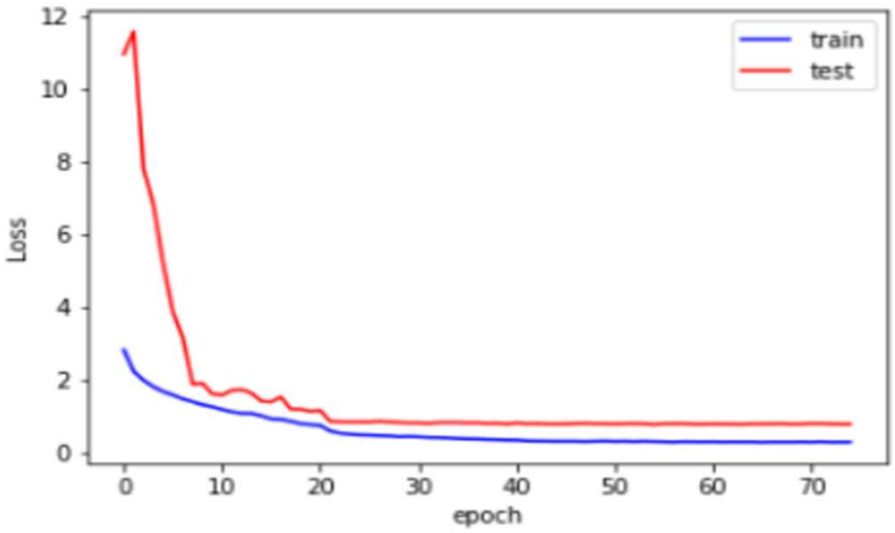


```
------------------------------------------------------------------------
Score per fold
------------------------------------------------------------------------
> Fold 1 - Loss: 0.7884774208068848 - Accuracy: 76.45000219345093%
------------------------------------------------------------------------
> Fold 2 - Loss: 0.7574902176856995 - Accuracy: 77.60000228881836%
------------------------------------------------------------------------
> Fold 3 - Loss: 0.7435040473937988 - Accuracy: 78.1000018119812%
------------------------------------------------------------------------
> Fold 4 - Loss: 0.8002474308013916 - Accuracy: 77.14999914169312%
------------------------------------------------------------------------
> Fold 5 - Loss: 0.7867732644081116 - Accuracy: 76.84999704360962%
------------------------------------------------------------------------
Average scores for all folds:
> Accuracy: 77.23000049591064 (+- 0.5749789229174928)
> Loss: 0.7752984762191772
------------------------------------------------------------------------
```

NO.	Accuracy
Fold1	0.7645
Fold2	0.7600
Fold3	0.7810
Fold4	0.7715
Fold5	0.7685
Average	0.7723

분류 결과


train
test
Accuracy
0.0
0.2
0.4
0.6
0.8
0
10
20
30
40
50
60
70
epoch


train
test
Loss
0
2
4
6
8
10
12
0
10
20
30
40
50
60
70
epoch

THANK YOU