홍채인식 2차 과제

생체인증보안

사이버보안전공

1871085

주유연

목차

01

Data

- Read Data
- Data Augmentation

02

Split Data

- train_test_split

03

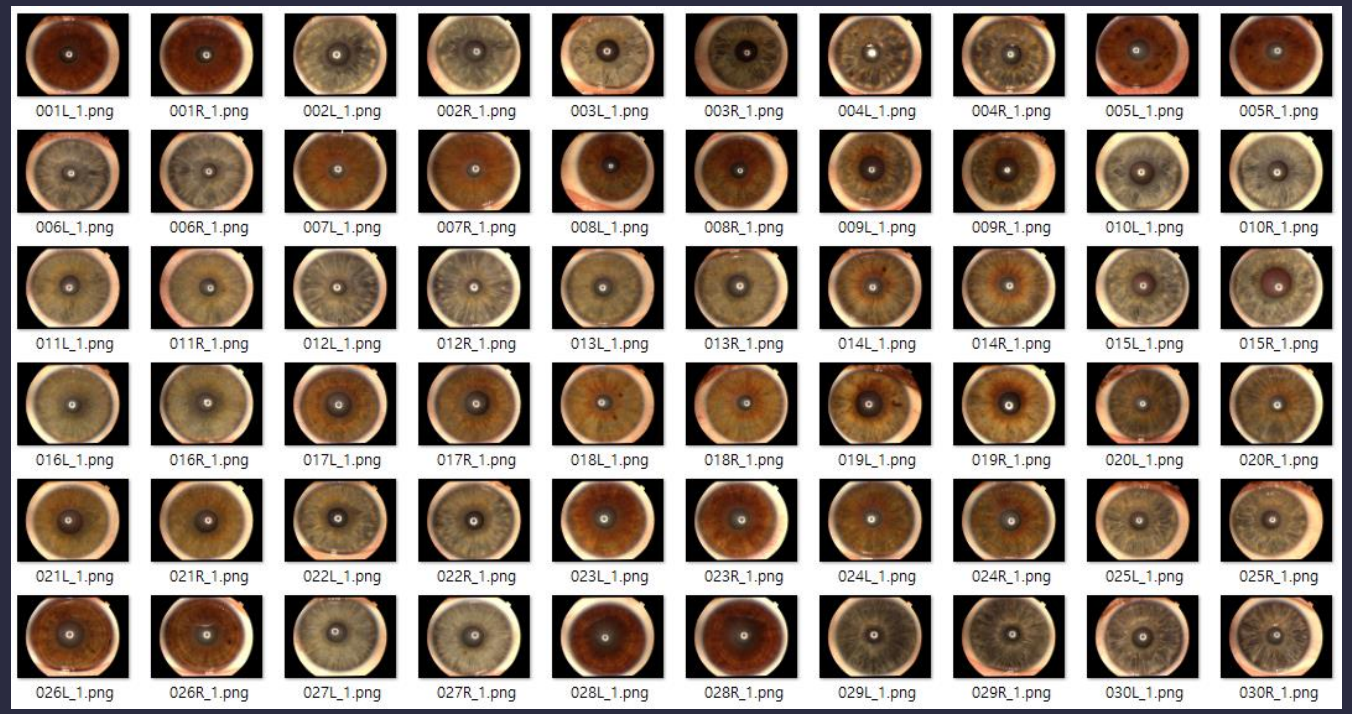
Model

- CNN Model
- Hyperparameter tuning
- Training Result

04

Evaluate Model

- Confusion Matrix



Given Data

- 64명의 홍채 데이터가 1명당 2개씩 총 128개의 홍채 이미지 데이터가 주어짐.
- 각 파일명의 앞쪽 숫자는 label을 의미함.

```
# 이미지 목록
images = glob.glob('./03_iris_training/*.png')
len(images)
```

```
128
```

```
r = re.compile('\d+')

img = [] # 이미지
label = [] # 라벨

for fname in images:
    l = r.findall(fname)[1]
    label.append(l)
    im = pilimg.open(fname)
    im = im.resize((int(im.width*0.3), int(im.height*0.3))) # 이미지 줄이기
    pix = np.array(im)
    img.append(pix)
```

```
X = np.array(img)
X.shape # img shape
```

```
(128, 172, 230, 3)
```

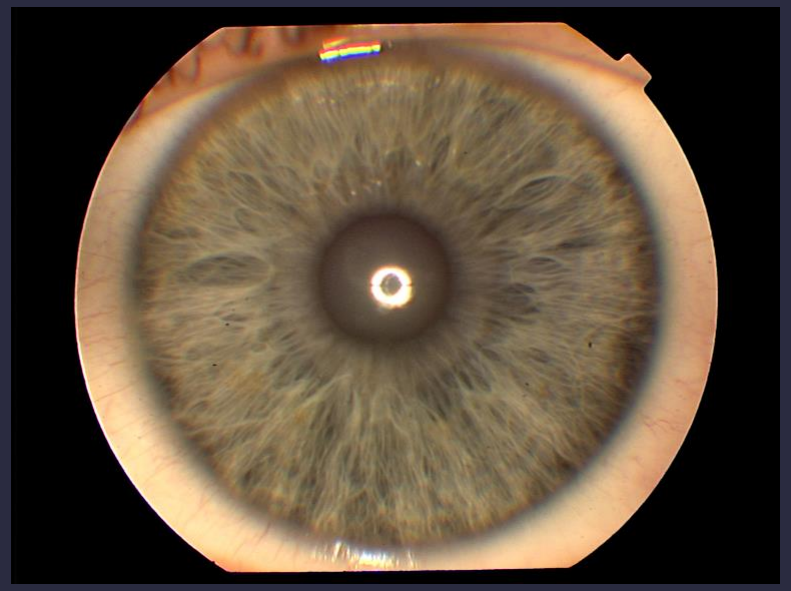
```
y = np.array(label, dtype='int32')
y # label
```

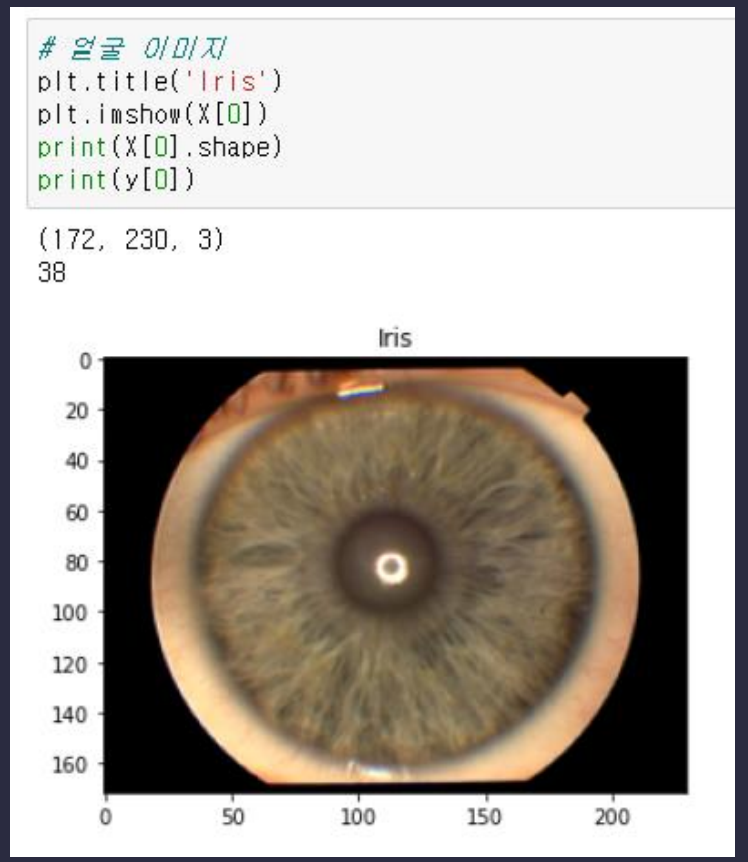
```
array([38, 12, 54, 56, 19, 51, 20, 48, 10, 45, 16, 13, 28, 29, 32, 34, 40,
       30, 59,  2, 59, 55, 51, 45, 61,  1, 15, 48, 35, 49, 64, 26, 38, 42,
       33,  5, 40, 25, 12, 56,  7, 27, 47, 33, 20, 57, 26, 43,  6, 23, 37,
       44, 17, 52, 21, 64, 27, 22, 31,  4, 10, 39, 19, 55, 41,  4, 58, 24,
        1, 18, 58, 49, 24, 53, 32,  9, 14, 14, 35, 36,  5,  2, 57, 54, 53,
       28, 60, 46, 37, 29, 46, 15, 36, 47, 43, 21, 30,  6, 44, 18, 50, 31,
       11,  3,  8, 22, 42, 50, 62, 41, 61, 63,  9,  3, 16,  7, 39, 60, 34,
       52, 17, 63, 23, 25, 11, 13,  8, 62], dtype=int32)
```

데이터 불러오기

- glob 함수를 이용해 파일명을 불러오고, 파일명에서 label을 읽어 별도의 리스트에 저장함.
- Out of memory 문제를 해결하기 위한 방법으로 이미지의 크기를 줄임 (30%)
- 불러온 이미지를 numpy array로 변환함.

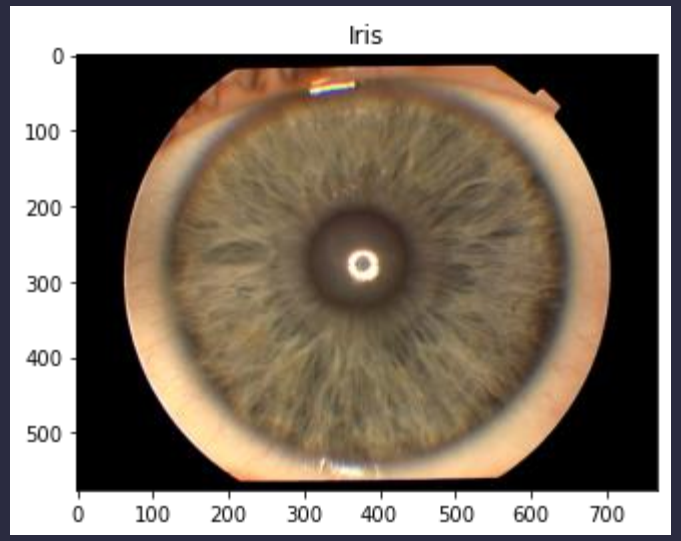
홍채 이미지 확인



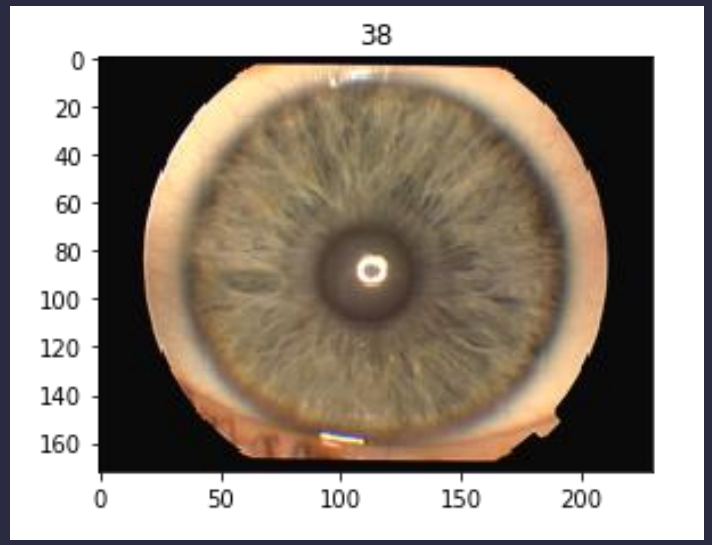

# 얼굴 이미지
plt.title('Iris')
plt.imshow(X[0])
print(X[0].shape)
print(y[0])
(172, 230, 3)
38
Iris
0
20
40
60
80
100
120
140
160
0
50
100
150
200

Data Augmentation

- 1명당 2개의 데이터는 얼굴을 구별하기에 부족하다고 판단.
- 이미지의 밝기 조절, 수평반전, 수직반전 등을 랜덤으로 적용한 image augmentation 사용.
- 각 이미지당 19개의 추가 augmented image를 생성하여 총 2560개의 데이터를 확보함.



- Add(-20,20)
- Fliplr(0.5)
- Flipud(0.5)



최종 Data

```
# 생성한 이미지
x_d = np.array(x_d)
y_d = np.array(y_d)
print(x_d.shape)
print(y_d.shape)
```

```
(2432, 172, 230, 3)
(2432,)
```

```
# 기존 이미지
print(X.shape)
print(y.shape)
```

```
(128, 172, 230, 3)
(128,)
```

```
# 기존 이미지, 생성 이미지 합치기
X_data = np.concatenate([X, x_d], axis=0)/255.
y_data = np.concatenate([y, y_d], axis=0)
print(X_data.shape)
print(y_data.shape)
```

```
(2560, 172, 230, 3)
(2560,)
```

- 1차 때와는 달리 image augmentation 후에 255.로 나누어 normalize 함.

Split Data

- sklearn.model_selection의 train_test_split 함수 이용.
- Train : Test의 비율은 8:2로 하고, stratify 옵션을 주어 label이 균등하게 나뉘도록 함.
- X_test, y_test는 모델 훈련 후 evaluate으로 이용함.

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X_data,y_data, test_size=0.2, shuffle=True, stratify=y_data, random_state=101)
```

```
X_train=X_train.astype('float32')
y_train=y_train.astype('int32')
X_test=X_test.astype('float32')
y_test=y_test.astype('int32')
```

```
print(X_train.shape, y_train.shape, X_test.shape, y_test.shape)
```

```
(2048, 172, 230, 3) (2048,) (512, 172, 230, 3) (512,)
```

```
# input shape
X_train[0].shape
```

```
(172, 230, 3)
```

모델 학습

- Stratified K-Fold Cross Validation으로 모델을 평가함.
- 데이터셋을 split하기 위해 label의 one-hot encoding은 학습 직전에 함수를 적용함

```
def score_model(cv=None):
    if cv is None:
        cv = StratifiedKFold(n_splits=4, random_state=42, shuffle=True)

    i=0
    for train_fold_index, val_fold_index in cv.split(X_train, y_train):
        i=i+1
        print('Fold #',i)
        # Get the training data
        X_train_fold, y_train_fold = X_train[train_fold_index], y_train[train_fold_index]
        # Get the validation data
        X_val_fold, y_val_fold = X_train[val_fold_index], y_train[val_fold_index]

        x,y,z,w = X_train_fold.shape
        print(y_train_fold)
        print(X_val_fold.shape)
        print(y_val_fold.shape)

        # to_categorical
        y_train_fold = to_categorical(y_train_fold, num_classes=65)
        y_val_fold = to_categorical(y_val_fold, num_classes=65)
```

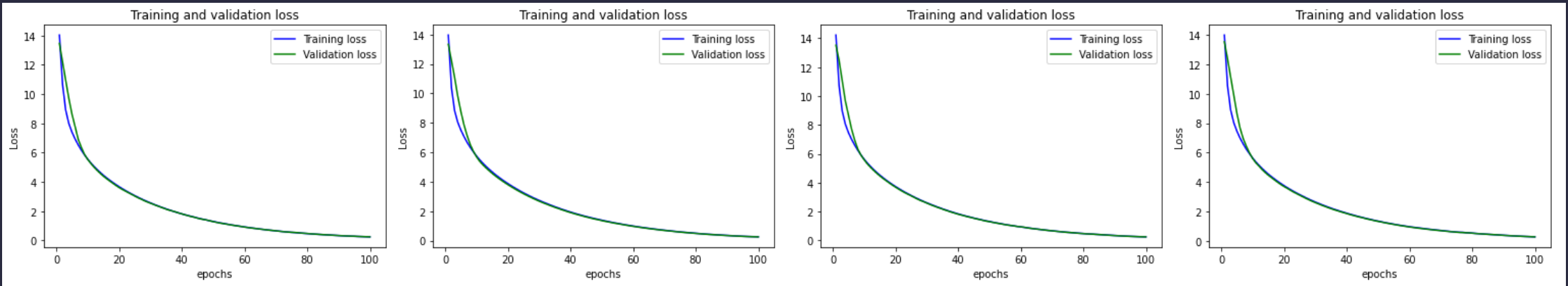
모델

- Pre-trained 모델인 ResNet50의 일부를 상위 레이어에 추가하여 특징을 추출함.
- Dense 레이어에는 l2 kernel_regularizer를 적용하여 오버피팅을 방지함.
- 두 Dense 레이어 사이에 Dropout(0.5)를 추가하여 오버피팅을 방지함.
- Loss func: categorical_crossentropy optimizer=Adam

```
def build_model():
    learning_rate = 0.0001
    METRICS = [
      tf.keras.metrics.CategoricalAccuracy(name='accuracy')
    ]
    input = Input(X_train[0].shape)
    base_model = ResNet50V2(include_top=False, input_tensor=input)

    for layer in base_model.layers[:44]:
        layer.trainable=False
    model = base_model.output
    model = MaxPooling2D()(model)

    model = Flatten()(model)
    model = Dense(512, activation='relu', kernel_regularizer='l2')(model)
    model = Dropout(0.5)(model)
    model = Dense(128, activation='relu')(model)

    output = Dense(65, activation='softmax')(model)

    model = tf.keras.Model(input,output)
    model.compile(loss='categorical_crossentropy', optimizer=tf.keras.optimizers.Adam(
    return model

build_model().summary()
```

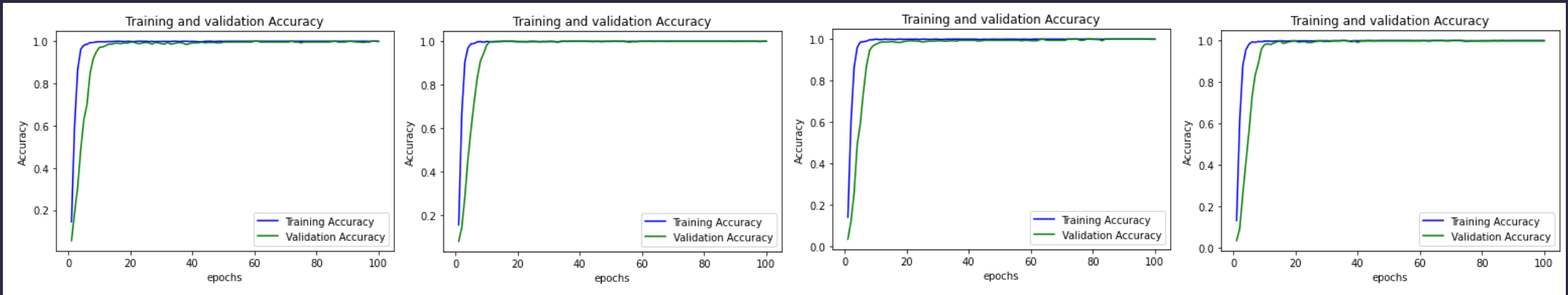
Training and Validation Loss

- StratifiedKFold (k=4) Validation
- Fold 별 Train Loss와 Validation Loss
- 매 Fold의 양상이 비슷함을 확인할 수 있음.



Training and Validation Accuracy

- StratifiedKFold (k=4) Validation
- Fold 별 Train Accuracy 와 Validation Accuracy
- 매 Fold의 양상이 비슷함을 확인할 수 있음.

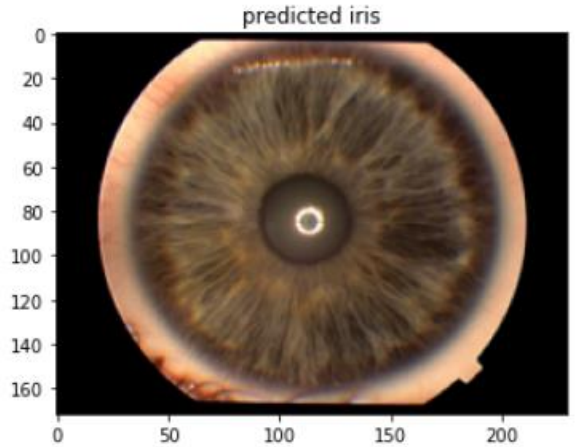


예측 결과 확인

- Test set으로 분리한 데이터의 일부의 실제 label과 예측 label을 확인함.

```
# 예측결과
plt.title('predicted iris')
this_img = X_test[0]
plt.imshow(this_img)
print(this_img.shape)
print('예측: ', preds[0])
print('실제: ', y_test[0])
```

```
(172, 230, 3)
예측:  39
실제:  39
```



Accuracy, Precision, Recall, F1 score

- Multi-class model임을 고려하여 평가지표들을 계산함.
- 각 label마다의 confusion matrix TP, FN, FP, TN의 평균인 Macro Average를 계산함.

```
from sklearn.metrics import classification_report
print(classification_report(y_test,preds, zero_division=1))
```

	precision	recall	f1-score	support
1	1.00	1.00	1.00	8
2	1.00	1.00	1.00	8
3	1.00	1.00	1.00	8
4	1.00	1.00	1.00	8
5	1.00	1.00	1.00	8
6	1.00	1.00	1.00	8
7	1.00	1.00	1.00	8

	Precision	Recall	F1-score	
accuracy			1.00	512
macro avg	1.00	1.00	1.00	512
weighted avg	1.00	1.00	1.00	512

Test 예측 결과

	Image	Answer
92	1	46
83	2	59
245	3	26
212	4	46
217	5	51
...	...	...
141	252	21
229	253	14
177	254	26
25	255	45
208	256	16

256 rows × 2 columns

정리

- 이미지의 크기를 줄여 Pre-trained model인 ResNet50을 본 모델에 적용시킬 수 있었다.
- 기존의 image augmentation을 그대로 사용하지 않고, 이미지의 밝기를 변화시키고, 좌우반전 및 상하반전을 이용해 이전 제출 때보다 더 많은 이미지를 생성해 모델을 학습시켰는데 결과가 좀 더 좋게 나온 것 같다.
- 이전까지 Kfold Validation을 제대로 수행하지 못했는데, 이를 수정하여 validation을 할 수 있었다.

감사합니다