홍채인식 1차 과제

생체인증보안

사이버보안전공

1871085

주유연

목차

01

Data

- Read Data
- Data Augmentation

02

Split Data

- train_test_split

03

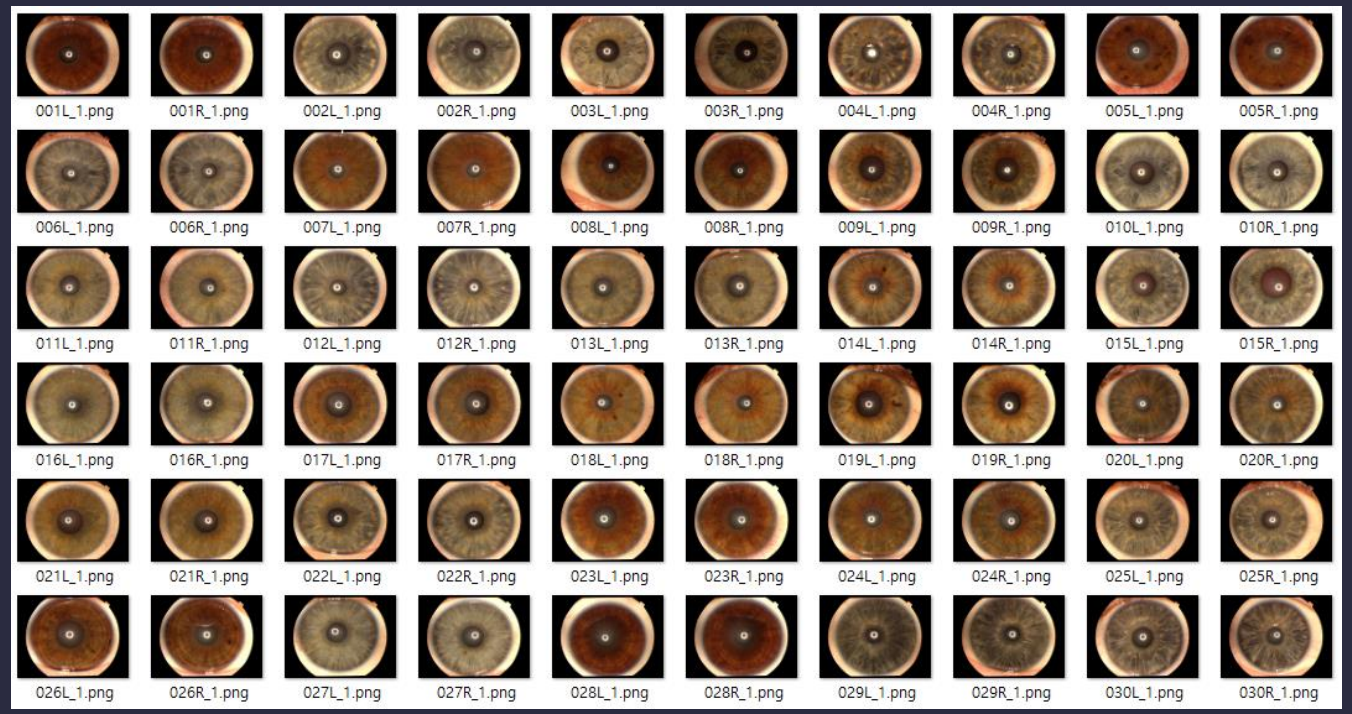
Model

- 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)
    pix = np.array(im)/255. # Normalize
    img.append(pix)
```

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

```
(128, 576, 768, 3)
```

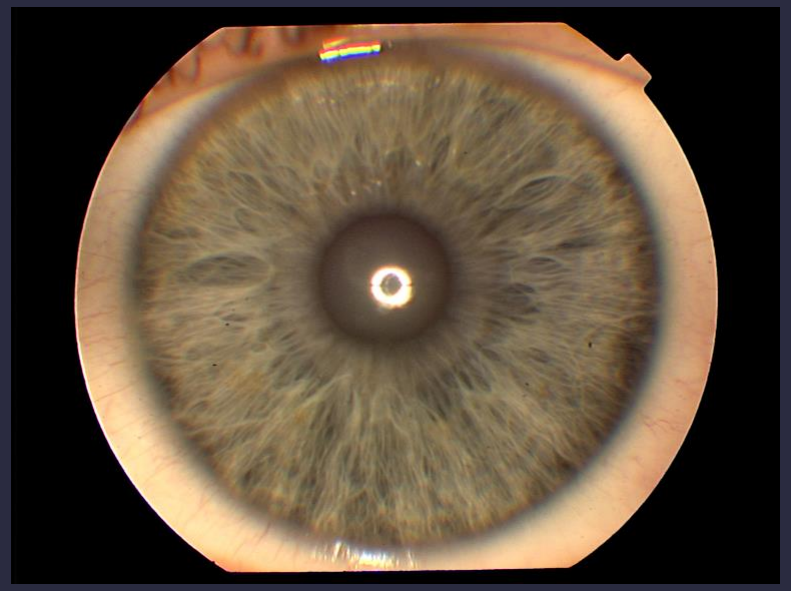
```
y = np.array(label, dtype='int32')
y # label
```

```
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.
- 불러온 이미지를 numpy array로 변환함.

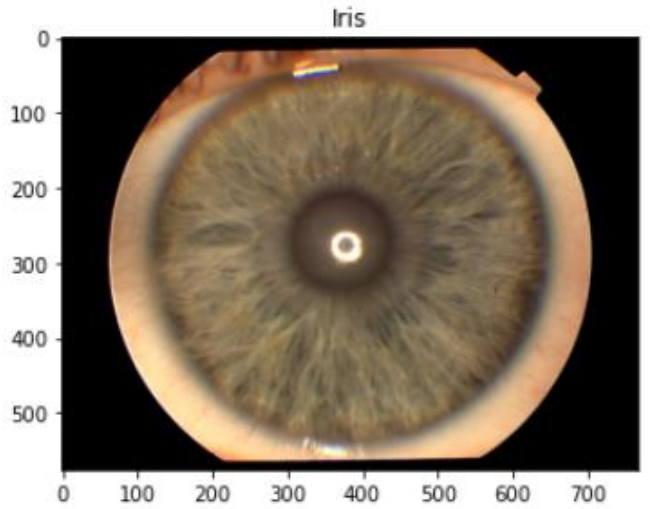
홍채 이미지 확인



- Normalization

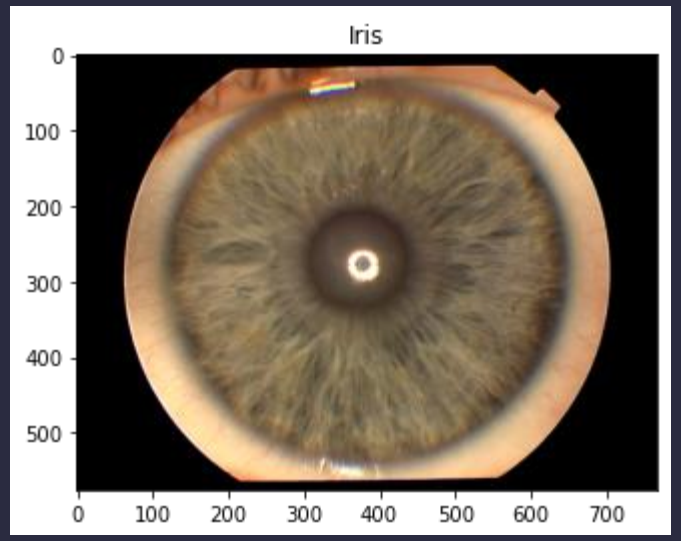
```
# 얼굴 이미지
plt.title('Iris')
plt.imshow(X[0])
print(X[0].shape)
print(y[0])
```

```
(576, 768, 3)
38
```

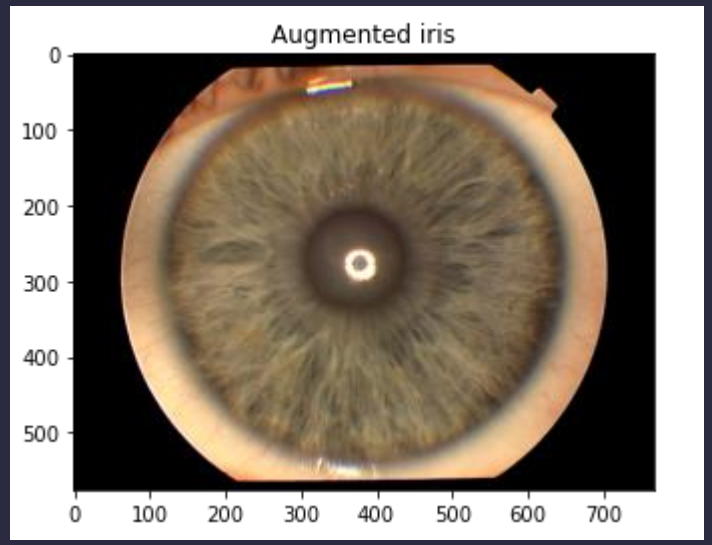


Data Augmentation

- 1명당 2개의 데이터는 얼굴을 구별하기에 부족하다고 판단.
- Affine의 translate_percent 값을 주어 거의 변형이 없는 이미지를 생성
- 각 이미지당 4개의 추가 augmented image를 생성하여 총 640개의 데이터를 확보함.



- Affine – translate



최종 Data

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

```
(512, 576, 768, 3)
(512,)
```

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

```
(128, 576, 768, 3)
(128,)
```

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

```
(640, 576, 768, 3)
(640,)
```

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)
```

```
(512, 576, 768, 3) (512,) (128, 576, 768, 3) (128,)
```

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

```
(576, 768, 3)
```

Label one-hot encoding (원-핫 인코딩)

- 모델 학습 시 loss function으로 categorical_crossentropy를 사용하기 위해 label을 원-핫 인코딩함.
- Label 값을 희소행렬로 변환해준다.

```
from keras.utils import to_categorical
y_train = to_categorical(y_train, num_classes=65)
y_test = to_categorical(y_test, num_classes=65)
```

시도해본 모델

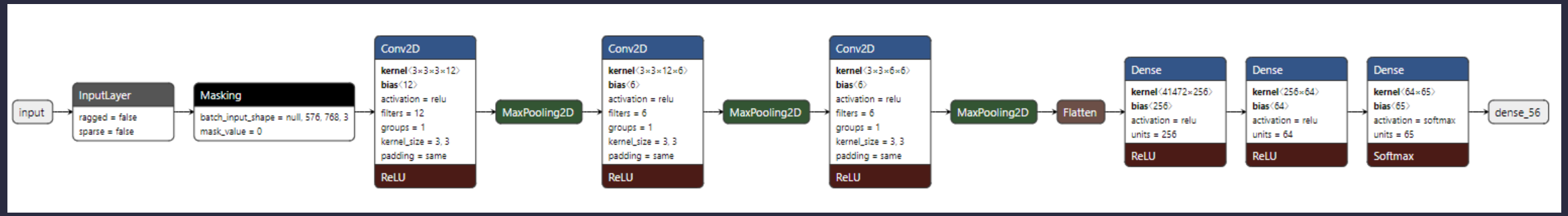
	01	02	03	04	05	06	07	08	09	10
Augmentation 수	4	4	4	3	3	2	2	4	4	4
Model 구조	Conv 12 MaxPool Conv 6 MaxPool Dense128 Dense64	Conv 12 MaxPool Conv 6 MaxPool Dense128 Dense64	Conv 12 MaxPool Conv 6 MaxPool Dense256 Dense64	Conv 12 MaxPool Conv 6 MaxPool Dense128 Dense32	Conv 6 MaxPool Conv 3 MaxPool Dense128 Dense64	Conv 12 MaxPool Dense128 Dense64	Conv 12 MaxPool Conv 6 MaxPool Dense128 Dense64	Conv 12 MaxPool Conv 6 MaxPool Conv 6 MaxPool Dense256 Dense64	Conv 6 MaxPool Conv 3 MaxPool Dense128 Dense64	Conv 6 MaxPool Conv 3 MaxPool Dense128 Dense64
Optimizer	Adam	Adam	Adam	Adam	Adam	Adam	Adam	Adam	RMSprop	Adam
Epoch	100*4	120*4	80*4	80*4	80*4	80*4	80*4	80*4	80*4	40*4
Accuracy	0.98	0.98	0.98	0.81	0.96	0.68	0.94	0.98	1.0	0.98
Precision	0.99	0.99	0.98	0.90	0.98	0.81	0.96	0.99	1.0	0.98
Recall	0.98	0.98	0.98	0.80	0.96	0.66	0.31	0.98	1.0	0.98
F1-score	0.98	0.98	0.97	0.76	0.95	0.58	0.91	0.98	1.0	0.98

제출한 모델

	01	02	03	04	05	06	07	08	09	10
Augmentation 수	4	4	4	3	3	2	2	4	4	4
Model 구조	Conv 12 MaxPool Conv 6 MaxPool Dense128 Dense64	Conv 12 MaxPool Conv 6 MaxPool Dense128 Dense64	Conv 12 MaxPool Conv 6 MaxPool Dense256 Dense64	Conv 12 MaxPool Conv 6 MaxPool Dense128 Dense32	Conv 6 MaxPool Conv 3 MaxPool Dense128 Dense64	Conv 12 MaxPool Dense128 Dense64	Conv 12 MaxPool Conv 6 MaxPool Dense128 Dense64	Conv 12 MaxPool Conv 6 MaxPool Conv 6 MaxPool Dense256 Dense64	Conv 6 MaxPool Conv 3 MaxPool Dense128 Dense64	Conv 6 MaxPool Conv 3 MaxPool Dense128 Dense64
Optimizer	Adam	Adam	Adam	Adam	Adam	Adam	Adam	Adam	RMSprop	Adam
Epoch	100*4	120*4	80*4	80*4	80*4	80*4	80*4	80*4	80*4	40*4
Accuracy	0.98	0.98	0.98	0.81	0.96	0.68	0.94	0.98	1.0	0.98
Precision	0.99	0.99	0.98	0.90	0.98	0.81	0.96	0.99	1.0	0.98
Recall	0.98	0.98	0.98	0.80	0.96	0.66	0.31	0.98	1.0	0.98
F1-score	0.98	0.98	0.97	0.76	0.95	0.58	0.91	0.98	1.0	0.98

CNN (Convolution Neural Network)

- 모델의 가장 앞에 Masking 레이어를 넣어 이미지 상에 나타나는 검은 부분을 모델 학습에 제외시킴.
- Convolution-MaxPooling2D 레이어를 3층으로 쌓아 반복적으로 특징을 추출함.
- 오버피팅 방지를 위해 Flatten 레이어 이후의 Dense 레이어에 L2 kernel_regularizer를 적용함.
- Compile 시 lr=0.00001 optimizer=adam 이용.



Model summary

```python
def build_model():
    learning_rate = 0.00001
    width = 768
    height = 576
    METRICS = [
      tf.keras.metrics.CategoricalAccuracy(name='accuracy')
    ]
    model = Sequential()
    model.add(Masking(mask_value=0., input_shape=X_train[0].shape)) # 데두리 검정 제외
    model.add(Conv2D(filters=12, kernel_size=(3,3), padding='same', activation='relu'))
    model.add(MaxPooling2D(pool_size=(2,2)))
    model.add(Conv2D(filters=6, kernel_size=(3,3), padding='same', activation='relu'))
    model.add(MaxPooling2D(pool_size=(2,2)))
    model.add(Conv2D(filters=6, kernel_size=(3,3), padding='same', activation='relu'))
    model.add(MaxPooling2D(pool_size=(2,2)))

    model.add(Flatten())
    model.add(Dense(256, activation='relu', kernel_regularizer='l2'))
    model.add(Dense(64, activation='relu'))
    model.add(Dense(65, activation='softmax'))
    model.compile(loss='categorical_crossentropy', optimizer=tf.keras.optimizers.Adam(lr=learning_rate),
    return model

model = build_model()
model.summary()
```

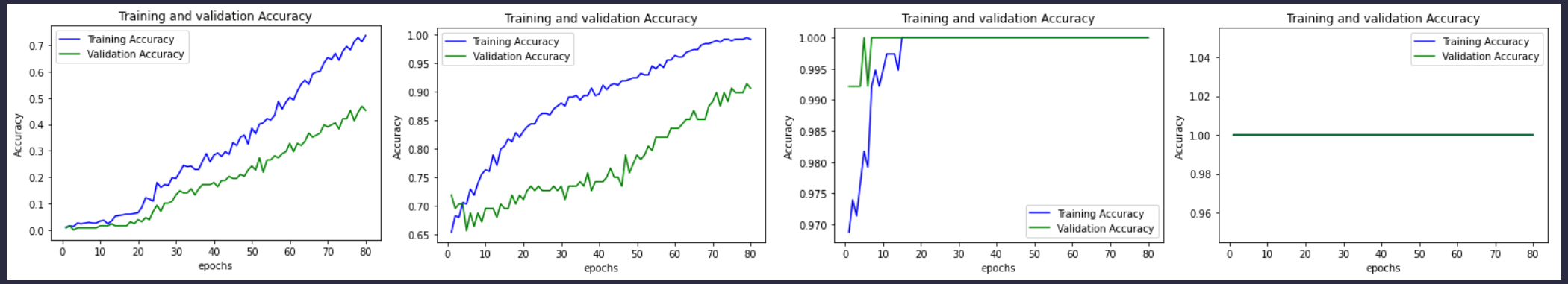
```
Model: "sequential_21"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
masking_21 (Masking)         (None, 576, 768, 3)       0
_________________________________________________________________
conv2d_89 (Conv2D)           (None, 576, 768, 12)      336
_________________________________________________________________
max_pooling2d_32 (MaxPooling (None, 288, 384, 12)      0
_________________________________________________________________
conv2d_90 (Conv2D)           (None, 288, 384, 6)       654
_________________________________________________________________
max_pooling2d_33 (MaxPooling (None, 144, 192, 6)       0
_________________________________________________________________
conv2d_91 (Conv2D)           (None, 144, 192, 6)       330
_________________________________________________________________
max_pooling2d_34 (MaxPooling (None, 72, 96, 6)         0
_________________________________________________________________
flatten_19 (Flatten)         (None, 41472)             0
_________________________________________________________________
dense_54 (Dense)             (None, 256)               10617088
_________________________________________________________________
dense_55 (Dense)             (None, 64)                16448
_________________________________________________________________
dense_56 (Dense)             (None, 65)                4225
=================================================================
Total params: 10,639,081
Trainable params: 10,639,081
Non-trainable params: 0
_________________________________________________________________
```

Training and Validation Loss

- KFold (k=4) Validation
- Fold 별 Train Loss와 Validation Loss
- Fold마다 loss가 줄어들었음을 확인할 수 있음.

Training and Validation Accuracy

- KFold (k=4) Validation
- Fold 별 Train Accuracy 와 Validation Accuracy
- Fold마다 Accuracy가 증가하며 1에 수렴함을 확인할 수 있음.

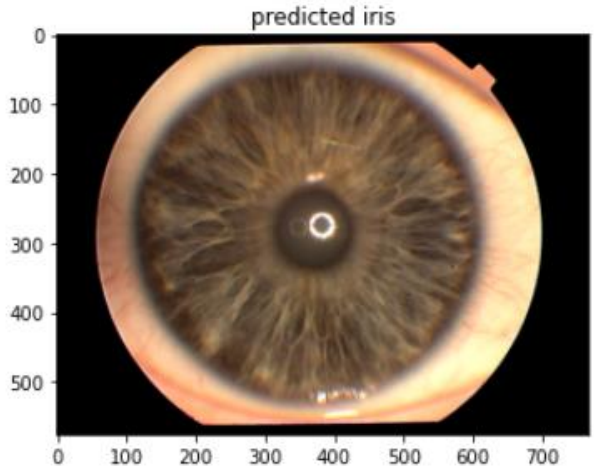


예측 결과 확인

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

```
# 예측결과
plt.title('predicted iris')
this_img = X_test[1]
plt.imshow(this_img)
print(this_img.shape)
print('예측: ', preds[1])
print('실제: ', y_test_origin[1])
```

```
(576, 768, 3)
예측:  30
실제:  30
```



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_origin,preds, zero_division=1))
```

	precision	recall	f1-score	support
1	1.00	1.00	1.00	2
2	0.67	1.00	0.80	2
3	1.00	1.00	1.00	2
4	1.00	1.00	1.00	2
5	1.00	1.00	1.00	2
6	1.00	1.00	1.00	2
7	1.00	1.00	1.00	2
8	1.00	1.00	1.00	2
9	1.00	1.00	1.00	2
10	1.00	1.00	1.00	2
11	1.00	1.00	1.00	2
12	1.00	1.00	1.00	2
13	1.00	1.00	1.00	2
14	1.00	1.00	1.00	2
15	1.00	1.00	1.00	2
16	1.00	1.00	1.00	2
17	1.00	1.00	1.00	2

	Precision	Recall	F1-score	
accuracy			0.98	128
macro avg	0.99	0.98	0.98	128
weighted avg	0.99	0.98	0.98	128

Test 예측 결과

	Image	Answer
92	1	46
83	2	45
245	3	23
212	4	46
217	5	57
...	...	...
141	252	21
229	253	14
177	254	64
25	255	37
208	256	15

256 rows × 2 columns

개선방향

- Pre-trained model인 ResNet50을 적용해보려 했으나 Out of memory 문제로 적용할 수 없음
- Cross Validation Fold 수 조절
- 모델 구조 변경
- Activation function 변경 (leaky relu, selu, swish 등)
- Optimizer 변경 (RMSprop, Nadam, NAG 등)

감사합니다