FastCampus
For Business

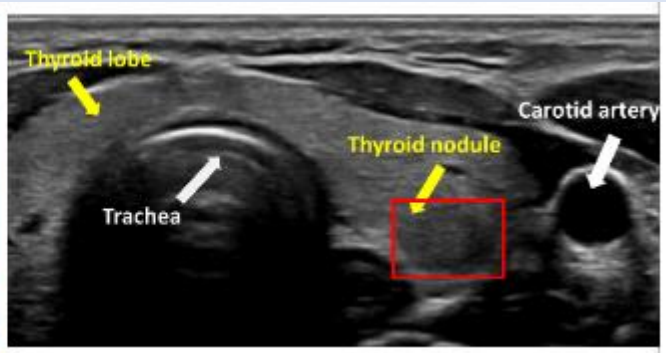
Thyroid Cancer Classification

INDEX

1. Overview
2. DataSet
3. Models
4. Result
5. End

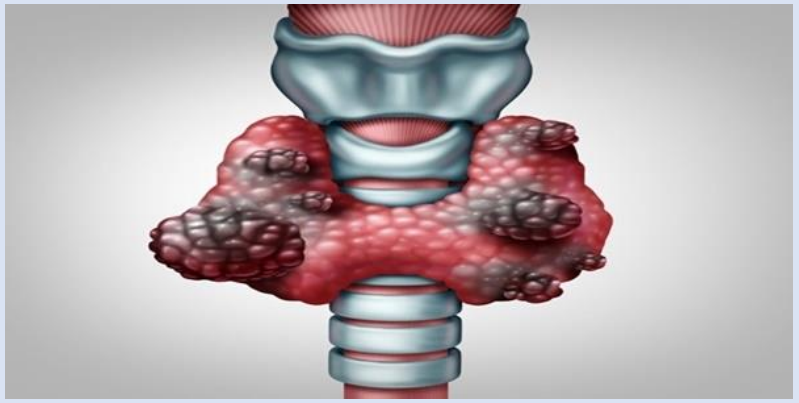
1. Overview

- 갑상선(Thyroid)
  - 갑상선 호르몬을 조절하는 내분비샘
  - 온몸의 대사를 조절
- 초음파(Ultrasound)
  - 이동성이 좋다.
  - 경제적이다.
  - 방사선 노출이 없다.



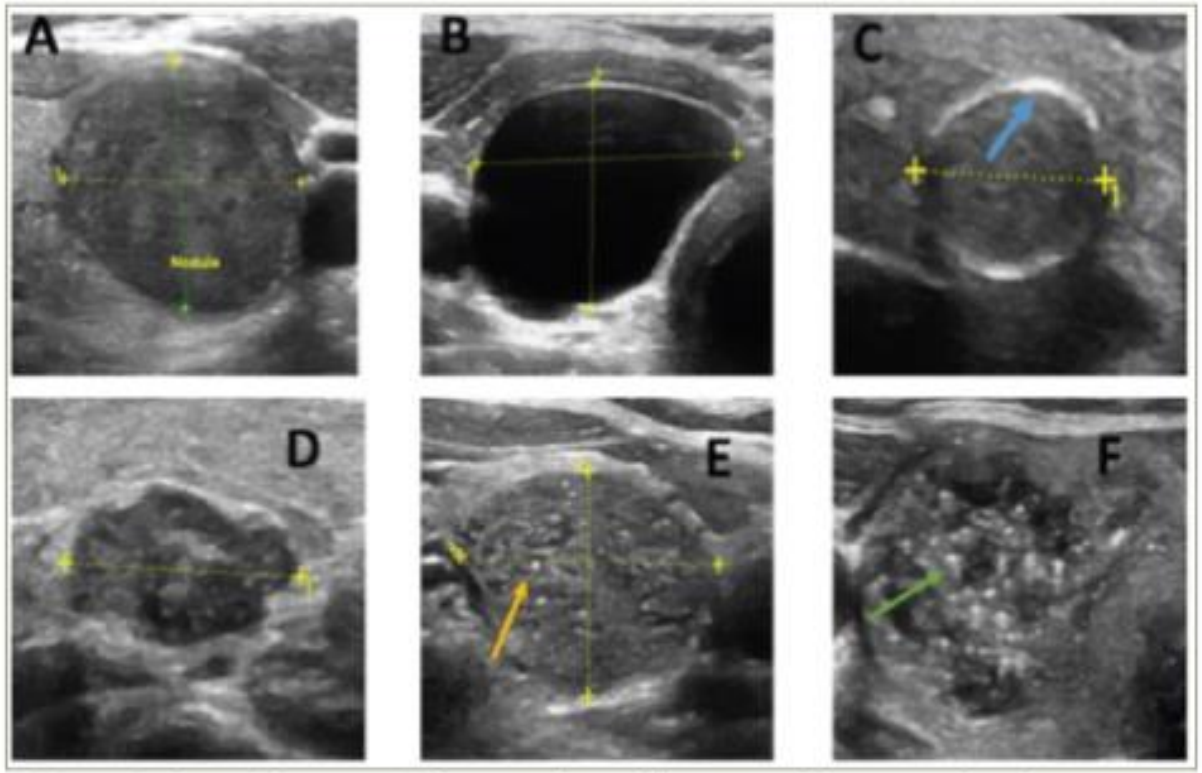
1. Overview

- 갑상선(Thyroid)
  - Benign
    - 종양이지만 전이 되지 않고 치료가 가능
  - Cancer
    - 악성 종양으로 주변 세포 및 조직으로 전이되어 기관 전체를 망가뜨림.



1. Overview

Figure 1: Some of the ultrasonic features used to describe thyroid nodules



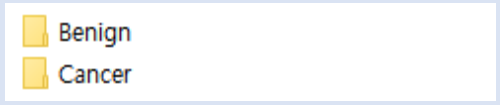
A. hypoechoic nodule; B. anechoic or cystic nodule; C. a nodule possessing rim calcification (blue arrow); D. nodule with jagged or irregular borders; E. microcalcifications (orange arrow); F. coarse calcifications with acoustic shadowing, also know as comet tails (green arrow).

1. Overview

- 딥러닝은 과연 classification을 잘 할 수 있을 것인가?
  - EfficientNet
  - VGG16
  - VGG19
  - ResNet50
  - ResNer101

2. Dataset

- Before Cropped Image

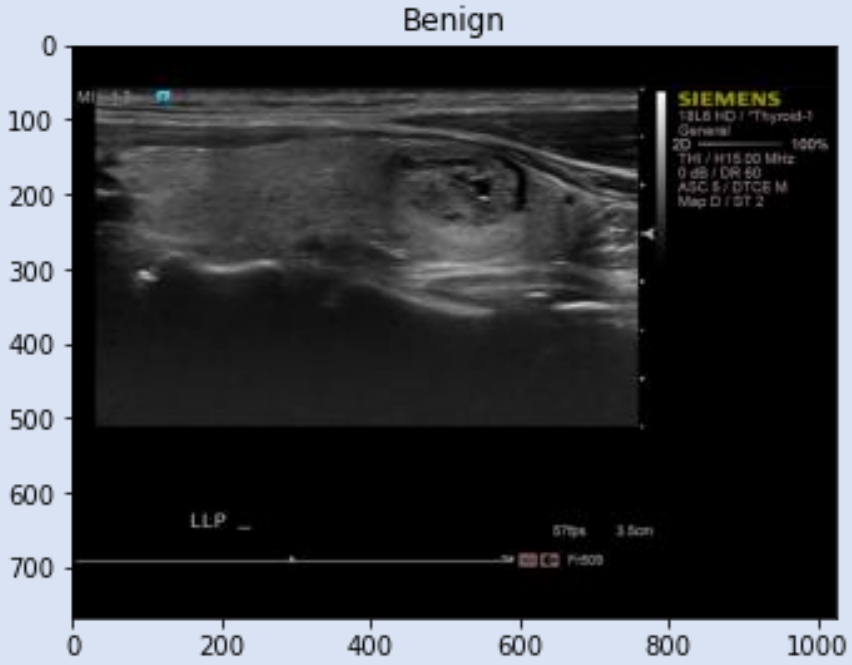


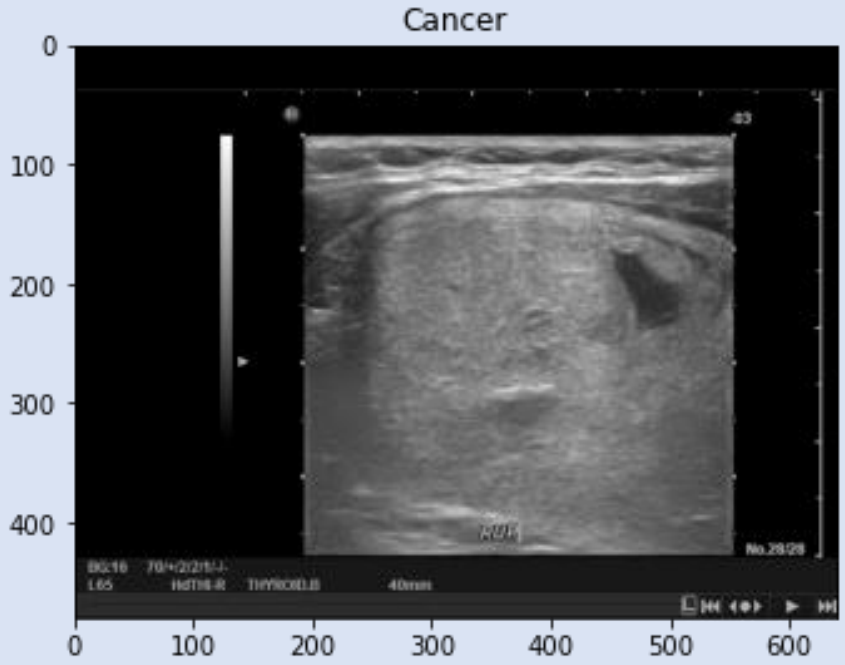
2. Dataset

- Before Cropped Image
  - TRAIN
    - Benign – 9004개 이미지
    - Cancer – 2389개 이미지
  - TEST
    - Benign – 899개 이미지
    - Cancer – 267개 이미지
  - 파일명 - B_001-0002-1.0.jpg
    - 모두 jpg 형식
    - B,M 으로 labelling

2. Dataset

- Before Cropped Image
  - 초음파 촬영 시 촬영 정보가 그대로 포함



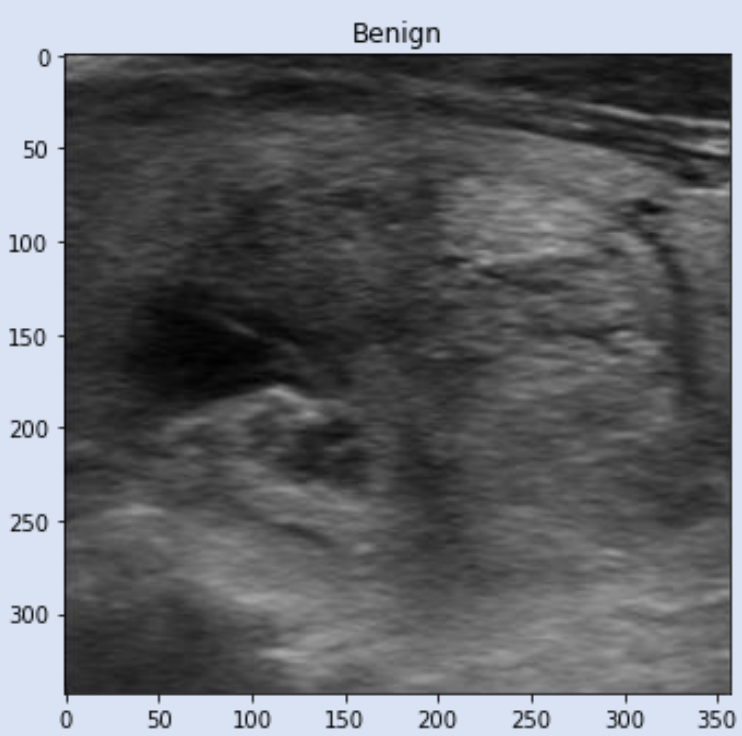


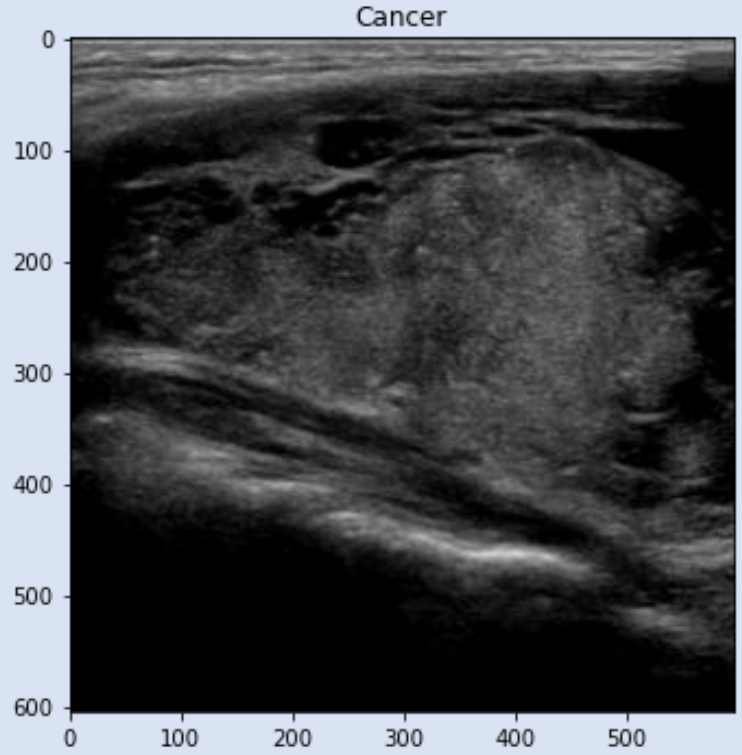
2. Dataset

- After Cropped Image
  - Train2
    - Benign – 7324개 이미지
    - Cancer – 1624개 이미지
  - Test2
    - Benign – 1831개 이미지
    - Cancer – 406개 이미지
  - 파일명 - B_001-0002-1.0.jpg
    - 모두 jpg 형식
    - B,M 으로 labelling

2. Dataset

- After Cropped Image
  - 종양이 발생한 부분을 잘라낸 이미지





3. Models

- 모델 선택
  - EfficientNetB7
    - 최신 모델이고 여러 방면에서 성능이 좋다고 알려짐
    - EfficientNetB7Model = tf.keras.applications.EfficientNetB7(include_top=False, weights='imagenet', input_tensor=None, input_shape=(img_width,img_height,3), pooling=None)
    - x = GlobalAveragePooling2D()(EfficientNetB7Model.output)
    - predictions = Dense(2, activation='softmax')(x)

3. Models

- 모델 선택
  - VGG16, VGG19
    - 수업 중 가장 많이 사용한 모델
    - VGG16Model = tf.keras.applications.VGG16(include_top=False, weights='imagenet', input_tensor=None, input_shape=(img_width,img_height,3), pooling=None)
    - x = GlobalAveragePooling2D()(VGG16Model.output)
    - predictions = Dense(2, activation='softmax')(x)

3. Models

- 모델 선택
  - ResNet50, ResNet101
    - 멘토링 이후 추가한 모델
    - ResNet50Model = tf.keras.applications.ResNet50(include_top=False, weights='imagenet', input_tensor=None, input_shape=(img_width,img_height,3), pooling=None)
    - x = GlobalAveragePooling2D()(ResNet50Model.output)
    - predictions = Dense(2, activation='softmax')(x)

3. Models

- Generator 설정

```
# Online-augmentation 적용 Generator
# 1. 이미지를 전부다 불러서 램 (메모리)에 올릴 수 없기 때문
# 2. 이미지는 Augmentation을 해주는게 좋아서

DATAGEN_TRAIN = ImageDataGenerator(
    rescale=1./255,
    rotation_range=20,
    width_shift_range=0.2,
    height_shift_range=0.2,
    shear_range=0.2,
    zoom_range=0.2,
    horizontal_flip=True,
    vertical_flip=True,
    featurewise_center=True,
    featurewise_std_normalization=True,
    data_format="channels_last",
    validation_split=0.10) # Train / Validation

# Online-augmentation 비적용 Generator (Test용)
DATAGEN_TEST = ImageDataGenerator(
    rescale=1./255,
    featurewise_center=True,
    featurewise_std_normalization=True,
    data_format="channels_last")
```

3. Models

- Generator 설정

```
# Generator의 instance 생성 (Train)
TRAIN_GENERATOR = DATAGEN_TRAIN.flow_from_directory(
    train_directory,
    target_size = (img_width, img_height),
    batch_size = batch_size,
    class_mode= "categorical",
    subset = "training")

VALID_GENERATOR = DATAGEN_TRAIN.flow_from_directory(
    train_directory,
    target_size = (img_width, img_height),
    batch_size = batch_size,
    class_mode= "categorical",
    subset = "validation")

# Generator의 instance 생성 (Test)
TEST_GENERATOR = DATAGEN_TEST.flow_from_directory(
    test_directory,
    target_size = (img_width, img_height),
    batch_size = batch_size,
    shuffle = False,
    class_mode= "categorical")
```

```
Found 10255 images belonging to 2 classes.
Found 1138 images belonging to 2 classes.
Found 1166 images belonging to 2 classes.
```

3. Models

- Model 학습
  - DeepLearning_EfficientNetB7.fit(
  - TRAIN_GENERATOR,
  - epochs=15,
  - callbacks=CALLBACK_EfficientNetB7,
  - shuffle=True,
  - validation_data=VALID_GENERATOR)

```
Epoch 12/15
 66/321 [=====>........................] - ETA: 1:32:06 -
```

3. Models

- Model 학습
  - https://keras.io/api/applications/

Model	Size (MB)	Top-1 Accuracy	Top-5 Accuracy	Parameters	Depth	Time (ms) per inference step (CPU)	Time (ms) per inference step (GPU)
Xception	88	79.0%	94.5%	22.9M	81	109.4	8.1
VGG16	528	71.3%	90.1%	138.4M	16	69.5	4.2
VGG19	549	71.3%	90.0%	143.7M	19	84.8	4.4
ResNet50	98	74.9%	92.1%	25.6M	107	58.2	4.6
ResNet50V2	98	76.0%	93.0%	25.6M	103	45.6	4.4
ResNet101	171	76.4%	92.8%	44.7M	209	89.6	5.2
ResNet101V2	171	77.2%	93.8%	44.7M	205	72.7	5.4
ResNet152	232	76.6%	93.1%	60.4M	311	127.4	6.5
ResNet152V2	232	78.0%	94.2%	60.4M	307	107.5	6.6
EfficientNetB7	256	84.3%	97.0%	66.7M	438	1578.9	61.6

3. Models

- Model 학습

```
DeepLearning_VGG16.fit(
    TRAIN_GENERATOR,
    epochs=15,
    callbacks=CALLBACK_VGG16,
    shuffle=True,
    validation_data=VALID_GENERATOR)

DeepLearning_ResNet50.fit(
    TRAIN_GENERATOR,
    epochs=15,
    callbacks=CALLBACK_ResNet50,
    shuffle=True,
    validation_data=VALID_GENERATOR)
```

4. Result(모델 예측)

- TEST_Prediction_EfficientNetB7 = DeepLearning_EfficientNetB7.predict_generator(TEST_GENERATOR, verbose=1)
- TEST_Prediction_VGG16 = DeepLearning_VGG16.predict_generator(TEST_GENERATOR, verbose=1)
- TEST_Prediction_VGG19 = DeepLearning_VGG19.predict_generator(TEST_GENERATOR, verbose=1)
- TEST_Prediction_ResNet50 = DeepLearning_ResNet50.predict_generator(TEST_GENERATOR, verbose=1)
- TEST_Prediction_ResNet101 = DeepLearning_ResNet101.predict_generator(TEST_GENERATOR, verbose=1)

call-back

- EfficientNetB7-001-0.5327-0.7909.hdf5
- VGG16-012-0.4714-0.8155.hdf5
- VGG19-010-0.4609-0.8172.hdf5
- ResNet50-005-0.5210-0.8181.hdf5
- ResNet101-005-0.5002-0.8251.hdf5

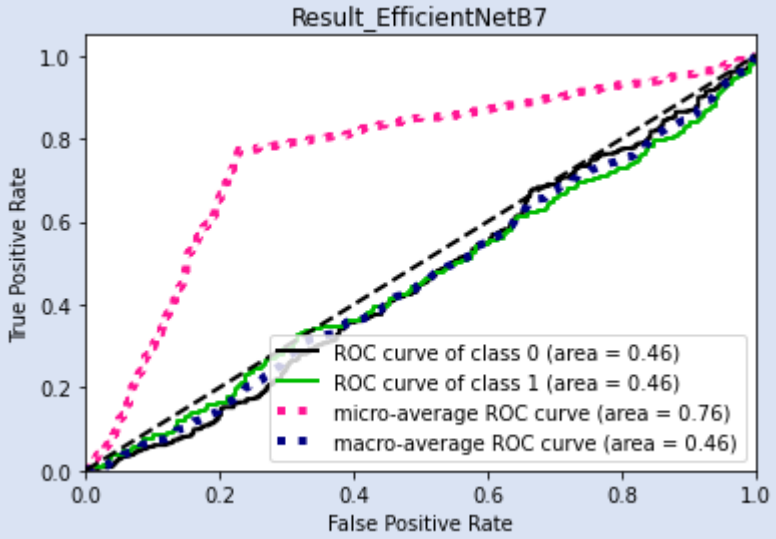
4. Result(DataFrame전환)

- Result_EfficientNetB7 = pd.DataFrame(TEST_Prediction_EfficientNetB7)
- Result_VGG16 = pd.DataFrame(TEST_Prediction_VGG16)
- Result_VGG19 = pd.DataFrame(TEST_Prediction_VGG19)
- Result_ResNet50 = pd.DataFrame(TEST_Prediction_ResNet50)
- Result_ResNet101 = pd.DataFrame(TEST_Prediction_ResNet101)

4. Result(ROC Curve)

- skplt.metrics.plot_roc(TEST_GENERATOR.classes.tolist(), Result_EfficientNetB7, title='Result_EfficientNetB7')
- skplt.metrics.plot_roc(TEST_GENERATOR.classes.tolist(), Result_VGG16, title='Result_VGG16')
- skplt.metrics.plot_roc(TEST_GENERATOR.classes.tolist(), Result_VGG19, title='Result_VGG19')
- skplt.metrics.plot_roc(TEST_GENERATOR.classes.tolist(), Result_ResNet50, title='Result_ResNet50')
- skplt.metrics.plot_roc(TEST_GENERATOR.classes.tolist(), Result_ResNet101, title='Result_ResNet101')

4. Result(ROC Curve)

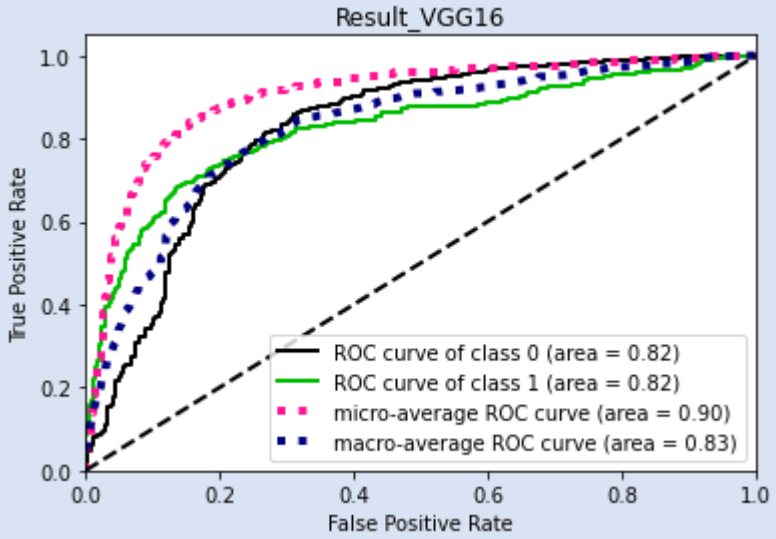


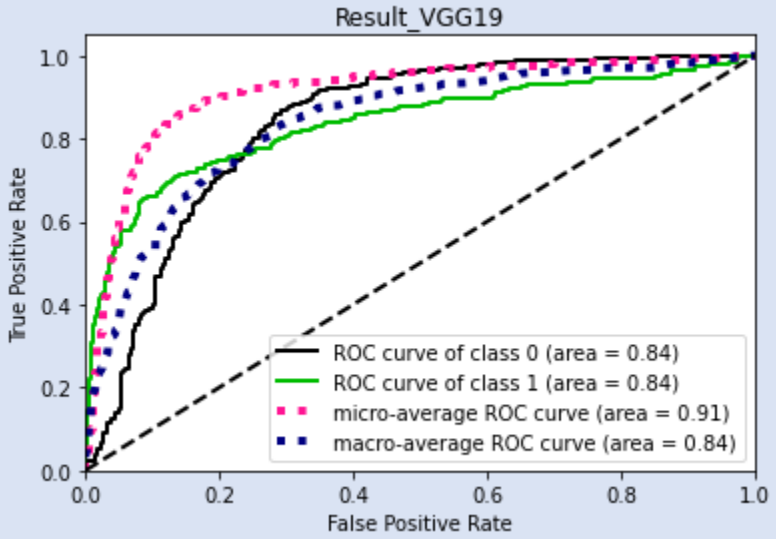
- EfficientNetB7
  - Validation accuracy는 0.79가 나왔지만, test data로 예측을 해보니 0.46이 나옴.

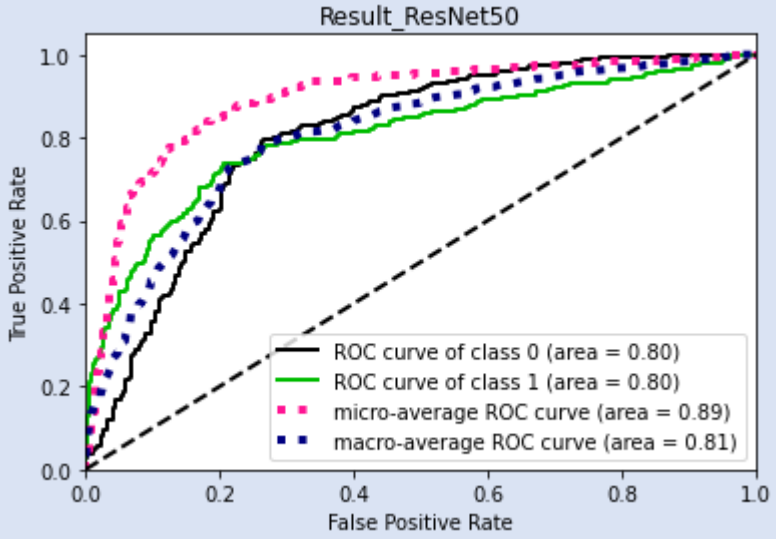
EfficientNetB7	256	84.3%	97.0%	66.7M	438	1578.9	61.6

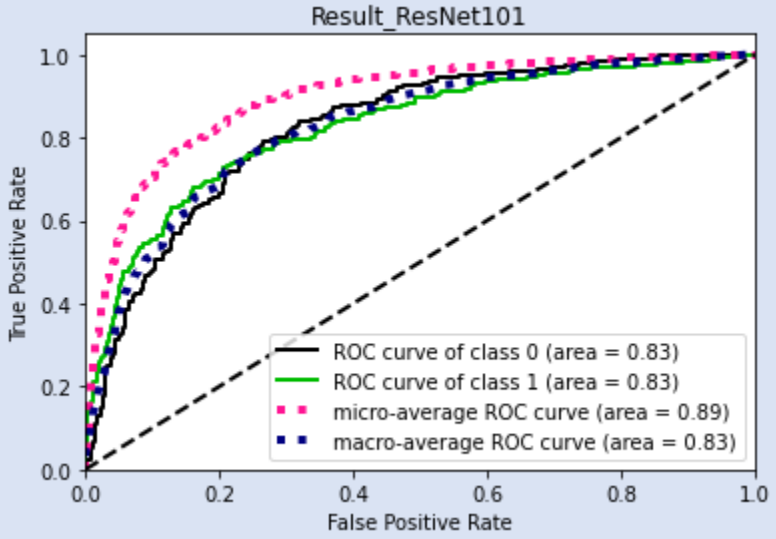
  - Data에 맞지 않는 너무 큰 모델을 사용했기 때문이고 생각됨.(overfitting)
  - 최신 모델이라고 항상 좋은 모델인 것은 아님.

4. Result(ROC Curve)





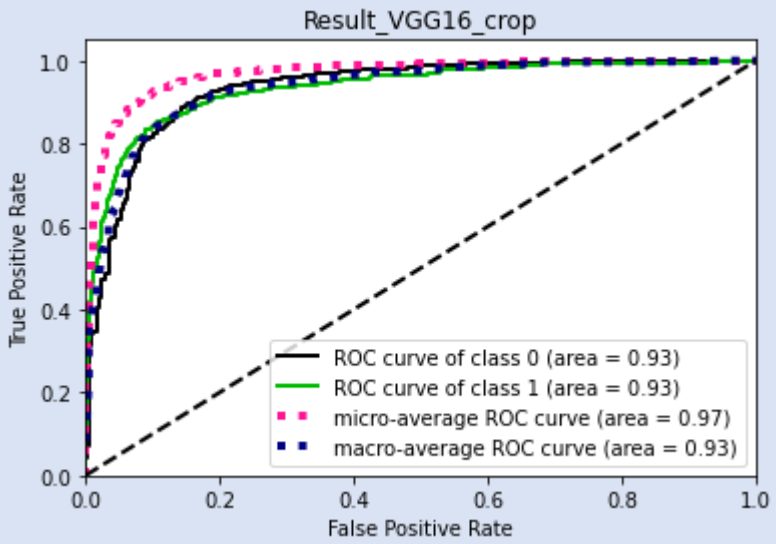


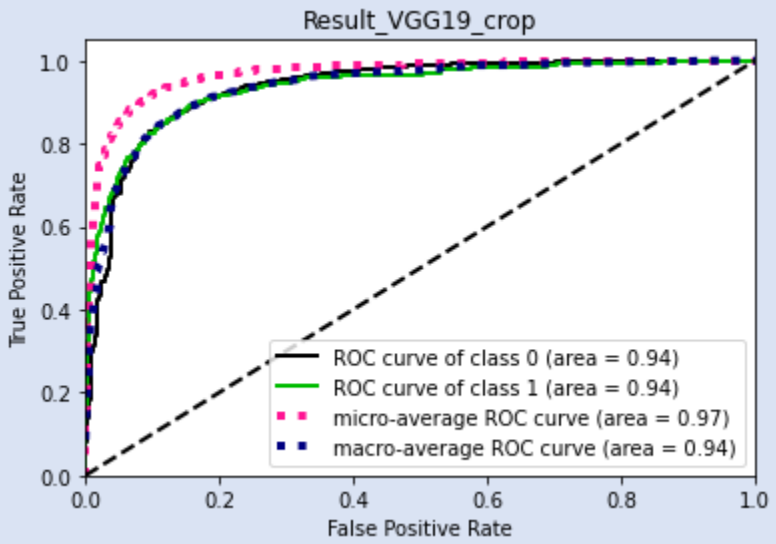


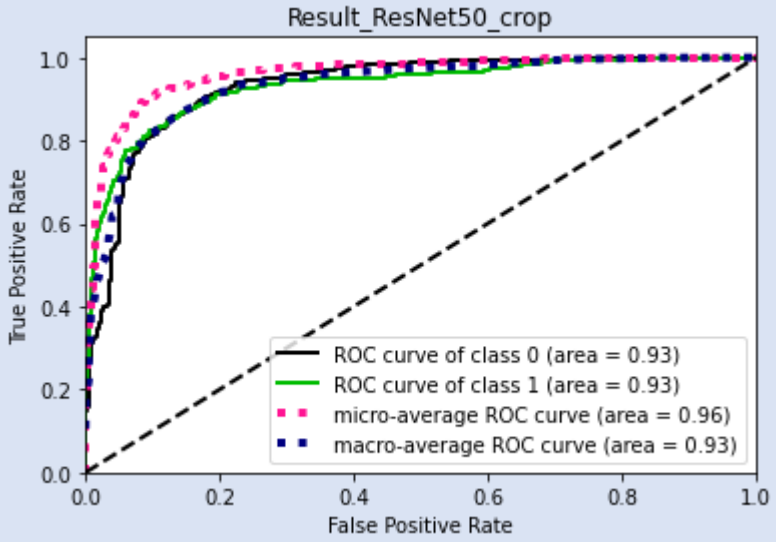
4. Result(학습 결과 종합)

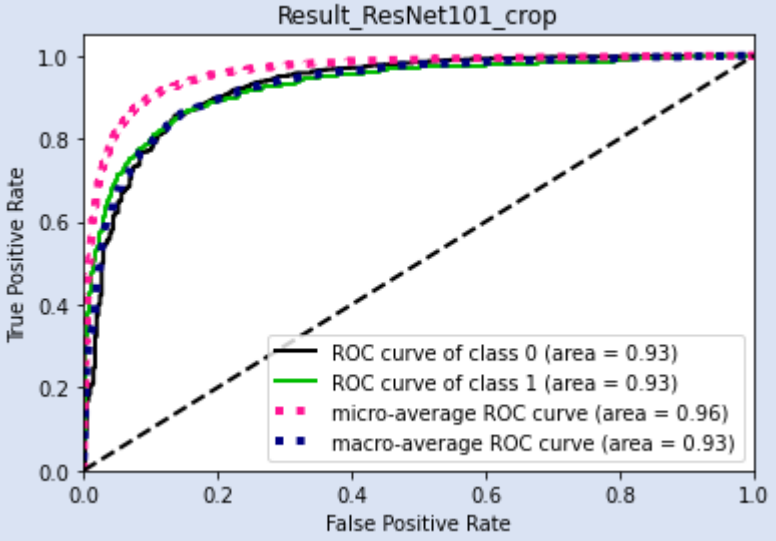
- EfficientNetB7을 제외한 다른 모델은 80% 이상의 확률로 classification을 함.
- 처음 모델 학습을 할 때에는 EfficientNetB7이 가장 좋은 모델이라고 하여 결과 또한 좋게 나올 것이라 예상.
- 의외로 제대로 학습이 되지 않았고, 이전 모델인 VGG, ResNet이 좋은 결과를 보여줌.
- VGG19 > ResNet101 > VGG16 > ResNet50 순서로 좋은 결과.
  - Keras document에 나와있는 값과는 다른 결과를 보인다.
- 어떤 data를 사용하는지, 어떤 모델을 선택하는지가 예측에 큰 영향을 준다는 것을 알 수 있었다.

4. Result(Cropped train data – Cropped test data)





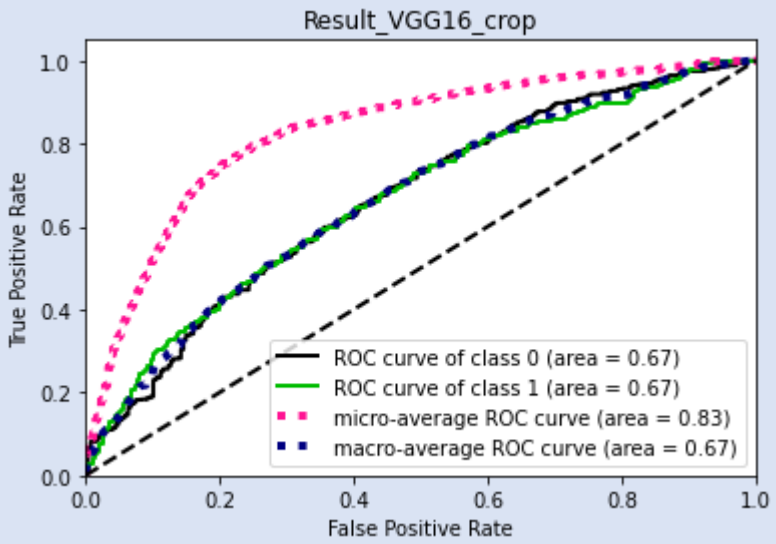


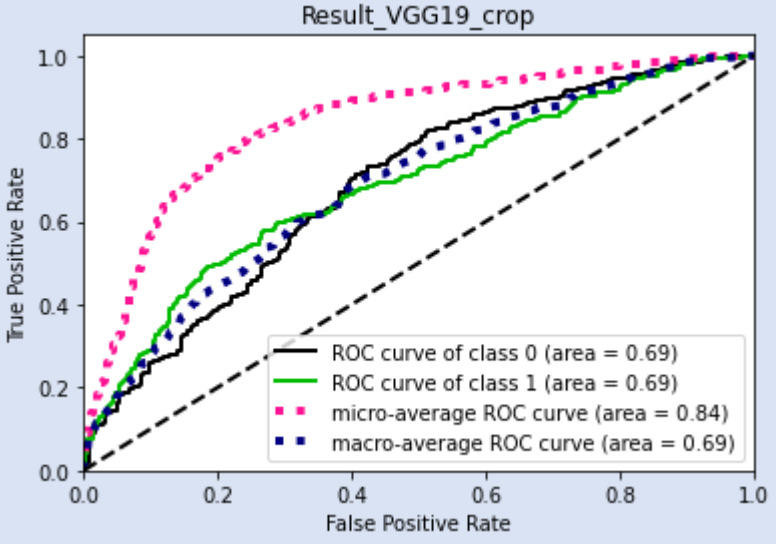


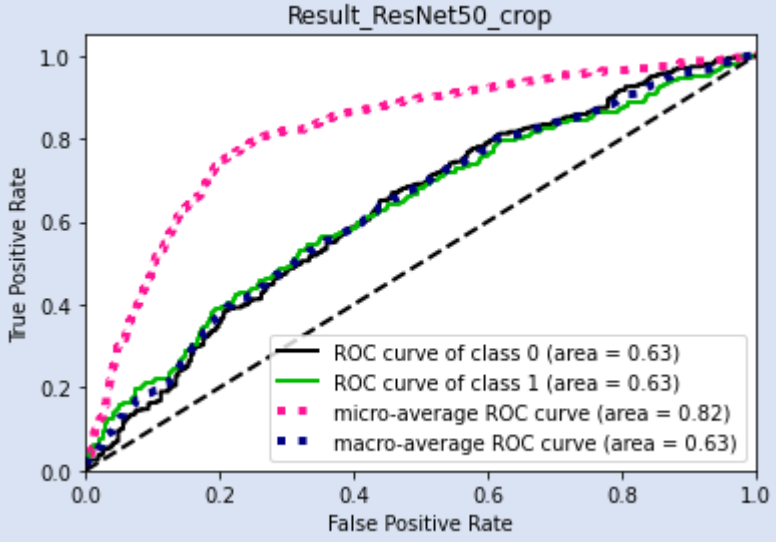
4. Result(Cropped train data – Cropped test data)

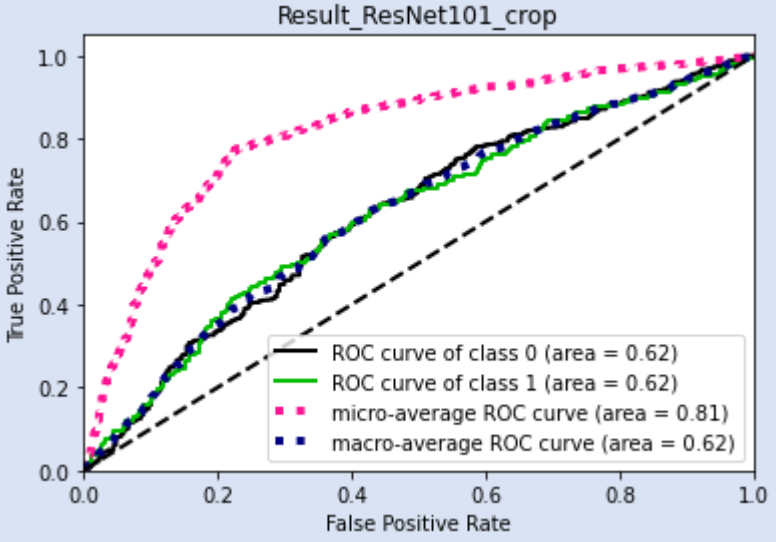
- Cropped된 data로 학습 후 Cropped된 data로 예측
  - 4개의 모델 모두 93% 이상의 확률로 Benign과 Cancer를 구분
  - ROC Curve도 매우 이상적인 모습
- 이미지를 전처리하는 것만으로도 모델의 성능이 비약적으로 상승함을 알 수 있는 결과

4. Result(Cropped train data – Normal test data)









4. Result(Cropped train data – Normal test data)

- Cropped된 data로 학습 후 Non-cropped된(Normal) data로 예측
  - 4개의 모델 모두 예측 성능이 매우 떨어졌다.
  - 상대적으로 VGG 모델이 ResNet 모델보다 좋게 나왔지만 모두 70% 미만의 확률
- 멘토링 당시 Cropped된 이미지로 학습 시 성능 하락을 예견
  - 이유 : Normal data(test data)는 초음파 사진에 대한 정보가 들어있기 때문
- 만약 이 모델을 실제로 사용할 경우, Cropped되지 않은 초음파 이미지를 그대로 사용할 것이기 때문에 예측 성능이 매우 떨어질 것이다.
- 따라서, 실제로 예측하고자 하는 이미지(test data)에 맞는 train data를 사용하는 것이 중요!

5. End(project를 마치며...)

- 이것저것 해보지 못함.
  - 모델을 학습하는데 시간이 너무 많이 소요
  - Parameter에 다양한 변화를 주며 성능을 비교하고 싶었으나...
  - 모델만 변경하여 결과를 예측
- 임상 데이터를 사용하지 못함.
  - 원래는 임상 데이터를 사용하여 multiclassification을 진행하고자 함.
  - Multiclassification에 대한 학습 부족 및 임상 데이터에 대한 정보 부족
  - 임상 데이터 EDA를 통해 예측 결과와 유의미한 연관성을 찾지 못함.
- 웹 사이트 개설 못함.
  - Benign과 Cancer를 구분할 수 있는 웹 사이트를 개설하고자 했으나...
  - Flask에 대한 학습 부족으로 이미지를 upload하는 Flask coding에 실패
  - 나중에 Flask 공부 후 웹 사이트를 개설할 예정