

**싸이버로지텍&성모병원**  
**위암/대장암 데이터 구축 및 진단 모델**

산출물 코드 설명 및 메뉴

# DCC(object Detection Crop and Classification)

Summary .....	3
1. Development Environment .....	4
<u><a href="#">A. development language .....</a></u>	<u><a href="#">4</a></u>
<u><a href="#">B. package list .....</a></u>	<u><a href="#">4</a></u>
2. Data Prepare .....	5
<u><a href="#">A. make annotation .....</a></u>	<u><a href="#">5</a></u>
<u><a href="#">B. split dataset.....</a></u>	<u><a href="#">6</a></u>
<u><a href="#">C. crop image .....</a></u>	<u><a href="#">8</a></u>
3. Detect Model.....	11
<u><a href="#">A. yoloV5 .....</a></u>	<u><a href="#">11</a></u>
4. Classification Model. ....	13
<u><a href="#">A. train classification model .....</a></u>	<u><a href="#">13</a></u>
<u><a href="#">B. inference classification model .....</a></u>	<u><a href="#">15</a></u>
5. 검증 및 평가.....	13
<u><a href="#">A. 검증 및 평가 .....</a></u>	<u><a href="#">17</a></u>

# Summary

2020년 NIA AI 데이터 구축 프로젝트(성모병원, 위암/대장암 데이터 구축 및 진단 모델)의 진단 모델 산출물 코드 설명 및 메뉴얼 문서입니다. 개발 환경은 1. Development Environment 에 버전 정보와 함께 기술 되어있습니다. 진단 모델의 실행 및 학습 과정은 크게 전처리 -> Object Detection 학습 -> Classification 학습 3 단계로 이루어져 있습니다.

- 전처리 단계에서는 Object Detection 과 Classification 학습을 위한 데이터 정제 단계입니다. 제공되는 데이터에서 AI 모델에 맞게 데이터셋을 구축하는 단계입니다.
- Detect 단계에서의 모델은 'yoloV5'를 사용합니다. 암 병변의 위치 정보를 학습하여 암 병변에 Bounding Box 를 생성합니다.
- Classification 의 모델은 'ResNet50', 'Inception', 'EfficientNet' 3 가지 모델을 사용하였습니다. 전처리 단계에서 제공되는 박스의 정보를 통해 이미지를 잘라낸 뒤 학습합니다.

# 1. Development Environment

## A. development language

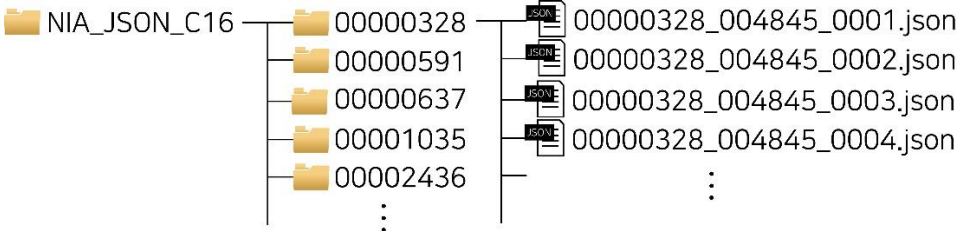
개발 언어	Version
Python	3.7.0

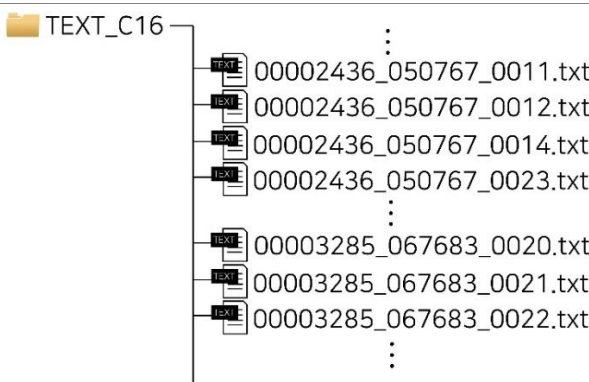
## B. package list

conda_env/final_env.txt	
package Name	Version
opencv-python	4.1.0.25
tensorboard-plugin-wit	1.7.0
tensorflow	2.3.0
h5py	2.10.0
keras	2.4.3
pandas	1.1.3
torch	1.7.0
torchvision	0.8.1

## 2. Data Prepare

### A. make annotation

make_txt_c16.py, make_txt_c18.py	
목적	json형식의 annotation 파일을 yolo 모델에 필요한 txt 형식의 annotation 파일로 변환하는 것을 목적으로 한다
입력 정보	<p>입력 정보로서 json 파일들이 모여있는 폴더의 경로와 txt 파일이 저장될 경로를 필요로 한다. Json 파일이 저장된 디렉토리의 구조가 아래의 그림 2-1처럼 되어있다면 최상단 폴더인 'NIA_JSON_C16' 폴더의 경로를 입력해야 한다.</p>  <p style="text-align: center;">그림 2-1 annotation dataset 구조</p>
실행 방법	<p>annotation 정보가 들어있는 json 파일들과 json 파일을 txt로 변환한 후 저장할 경로가 필요하다.</p> <p>각각 위암, 대장암의 이미지들을 분류하는 코드가 다음과 같다. make_txt_c16.py는 위암의 annotation 파일을 변환해주는 스크립트이고, make_txt_c18.py는 대장암의 annotation 파일을 변환해주는 스크립트이다. 코드 내의 변수 json_dir와 save_path를 수정 후 실행하면 된다.</p> <pre style="background-color: #f0f0f0; padding: 10px;"> &gt;&gt;python3 make_txt_c16.py &gt;&gt;python3 make_txt_c18.py </pre>
출력 정보	<p>입력에서 받은 폴더 아래의 모든 json파일들을 대상으로 txt로 변환하여 하나의 폴더에 저장한다.(그림 2-2 참고) txt 파일의 이름은 json 파일과 동일하며, annotation 이 들어있는 json에 대해서만 파일을 생성하기 때문에 json의 개수와 생성된 txt의 개수는 다르다.</p>

	 <p>그림 2-2 txt annotation dataset 구조</p>
--	--

## B. split dataset

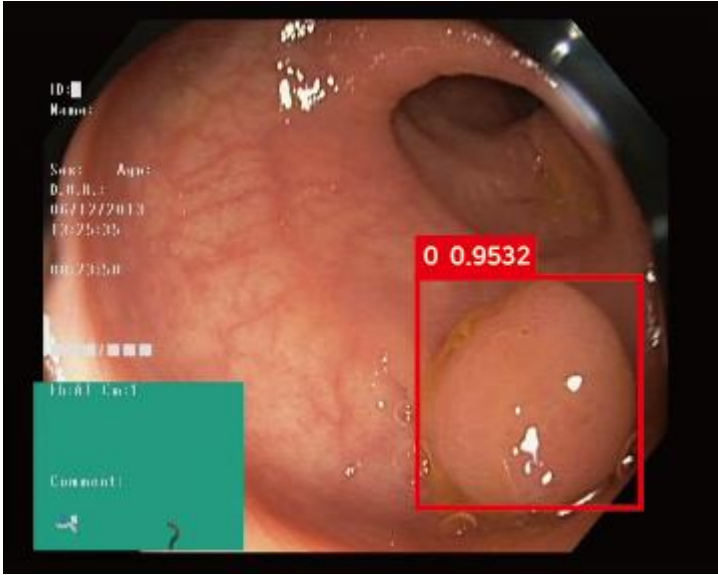
### split\_c16.py, split\_c18.py

목적	원본 이미지를 읽어서, train, valid, test의 3가지 부류로 이미지들을 분할하는 것을 목적으로 한다.
입력 정보	입력 정보로서 test용을 사용된 RID가 저장된 csv파일을 입력을 받는다.
실행 방법	<p>test용으로 사용할 이미지들의 환자 아이디 RID리스트를 모아놓은 리스트가 있어야 한다. 그 test로 사용된 해당 환자의 RID 리스트를 제외한 이미지 파일들을 train과 valid로 분류하게 된다.</p> <p>각각 위암, 대장암의 이미지들을 분류하는 코드가 다음과 같다. split_c16.py는 위암 이미지를 분류하는 파일이고, split_c18.py는 대장암 이미지를 분류하는 파일이다.</p> <pre> 33 34 df = pd.read_csv("./a_hyuk_test_C16.csv", dtype={"RID": str}) 35 df_test = df[df["GUBUN"] == "TEST"] 36 ALL_TEST = list(df_test["RID"].unique()) 37 ALL_TEST.sort() 38 print(ALL_TEST) 39 print() 40 41 df2 = pd.read_csv("./a2_hyuk_test_C16.csv", dtype={"RID": str}) 42 df_valid = df2[df2["GUBUN"] == "VALID"] 43 df_train = df2[df2["GUBUN"] == "TRAIN"] 44 ALL_VALID = list(df_valid["RID"].unique()) 45 ALL_VALID.sort() 46 print(ALL_VALID) 47 print() 48 49 ALL_TRAIN = list(df_train["RID"].unique()) 50 ALL_TRAIN.sort() 51 print(ALL_TRAIN) 52 print() 53 </pre>

	<p>위 그림에서, a_hyuk_test_C16.csv에는 테스트 할 RID리스트와 학습 및 검증에 사용할 RID 리스트, 두 가지로 분류하는 과정에서 필요한 정보이고, a2_hyuk_test_C16.csv는 학습 및 검증에 사용할 RID 리스트에서, 학습과 검증 두 가지로 나누기 위해서 필요한 csv 파일이다.</p> <p>결과적으로 위 코드에 따라, test, train, valid로 정할 내시경 이미지 파일들이 결정이 된다.</p> <pre> 54 55  img_list = glob.glob("/home/joyhyuk/final_hyuk/a_c16/*.txt") 56  img_list.sort() 57  print(len(img_list)) 58 </pre> <p>위에서 나오는 경로 "/home/joyhyuk/final_hyuk/a_c16/*.txt"는 실제로 내시경 파일 이미지를 모아 놓은 폴더 이름이다. 혹은 여기서는 이름은 같고, 확장자만 다른 율로 형식의 annotation파일 이름으로 대신 기입을 해서 수행을 한 것이다.</p> <pre> 59  train_img_list, val_img_list, test_list = split_dataset(img_list, ALL_TEST, ALL_VALID, ALL_TRAIN) 60 61  print(len(train_img_list), len(val_img_list), len(test_list)) 62 63  with open('./anewtrainByRid.txt', 'w') as f: 64      f.write('\n'.join(train_img_list) + '\n') 65 66  with open('./anewvalByRid.txt', 'w') as f: 67      f.write('\n'.join(val_img_list) + '\n') 68 69  with open('./anewtestByRid.txt', 'w') as f: 70      f.write('\n'.join(test_list) + '\n') </pre> <p>최종적으로 위 코드에 따라, train에 속하는 파일 이름 리스트, valid에서 사용할 파일 이름 리스트, test에서 사용할 파일 이름 리스트들이 절대 경로로 만들어진다.</p> <p>위에서 언급한 예시는 c16에 해당되는 이름이고, 동일한 과정으로 c18에 해당되는 이름으로 수정하면 된다.</p> <p>그래서, 최종적으로 위의 정보들을 결정해서, 다음의 명령어를 실행하면 된다.</p> <pre> &gt;&gt;python3 split_c16.py &gt;&gt;python3 split_c18.py </pre>
출력 정보	<p>실행 결과로서, train에 사용될 이미지 파일 리스트, valid에 사용될 이미지 리스트, test용으로 사용될 이미지 리스트가 저장된 3개의 파일들이 생성이 된다.</p>

## A. crop image

### crop\_make\_folder.py

목적	yolo 형식의 annotation 정보 파일을 읽어서, 해당 원본 이미지에서 특정 병변의 영역을 잘라서, (299,299)의 형식의 검은색 바탕의 이미지의 중앙에 놓은 crop image 를 만드는 것이 목적이다.
입력 정보	<p>각각의 train, valid, test의 3종류로 학습 데이터를 만들기 때문에, train에 속하는 파일 리스트, valid에 속하는 파일 리스트, test에 속하는 파일 리스트 파일을 각각 입력을 받아서 실행한다.</p>  <p>그림 5-1 원본 이미지</p>
실행 방법	<p>이미지를 train, valid, test의 3가지 유형으로 나누어야 하고, 그 각각의 파일이 위치할 폴더를 정해야 한다. 원본 이미지 파일과 그 각각의 이미지 파일에 대한 annotation파일이 같은 폴더에 있고, 이름은 같고, 확장자만 다르게 놓여 있다고 가정한다.</p> <pre> 13 14 dst_folder = "/home/aigpuserver/img_cls/c_data3/test" #실제 학습 데이터가 놓일 폴더 15 dst_folder2 = "/home/aigpuserver/img_cls/c_data3/test/" #실제 학습 데이터가 놓일 폴더 16 </pre> <p>실제로 학습 데이터로서 crop 이미지가 저장된 폴더를 지정한다. train, valid, test로 3가지로 지정해서 실행한다.</p>



```
num_of_class = 4 #클래스 종류의 개수
```

클래스 종류가 4가지로 정한다. (1기, 2기, 3기, 4기)

```
1 """
2
3 실제 이미지 경로들을 저장한 파일을 읽어들이는 코드
4 """
5 lines = open("./newtestByRid.txt").readlines()
6
```

newtestByRid.txt에 실제로 내시경 사진이 있는 절대 경로 파일 이름이 있다고 가정한다.

```
/home/joyhyuk/final_hyuk/a_c16/00002855_059601_0007.txt
/home/joyhyuk/final_hyuk/a_c16/00002855_059601_0008.txt
/home/joyhyuk/final_hyuk/a_c16/00002855_059601_0009.txt
/home/joyhyuk/final_hyuk/a_c16/00002855_059601_0010.txt
/home/joyhyuk/final_hyuk/a_c16/00002855_059601_0015.txt
/home/joyhyuk/final_hyuk/a_c16/00002855_059601_0016.txt
/home/joyhyuk/final_hyuk/a_c16/00002855_059601_0017.txt
/home/joyhyuk/final_hyuk/a_c16/00002855_059601_0018.txt
/home/joyhyuk/final_hyuk/a_c16/00002855_059601_0020.txt
/home/joyhyuk/final_hyuk/a_c16/00002855_059601_0033.txt
/home/joyhyuk/final_hyuk/a_c16/00002855_059601_0034.txt
/home/joyhyuk/final_hyuk/a_c16/00002855_059601_0035.txt
/home/joyhyuk/final_hyuk/a_c16/00002855_059601_0036.txt
/home/joyhyuk/final_hyuk/a_c16/00002855_059601_0037.txt
/home/joyhyuk/final_hyuk/a_c16/00002855_059601_0038.txt
```

위에서 언급된 것처럼 실제 내시경 이미지 파일 절대 경로 리스트가 위 사진처럼 기록되어 있다고 가정한다.

```
image_path3 = "/home/joyhyuk/final_hyuk/c16/"+ ff[:4]+".jpg"
label_path3 = "/home/joyhyuk/final_hyuk/a_c16/"+ff[:4]+".txt"
```

위 정보에서 이미지가 저장된 폴더와 라벨 정보가 저장된 폴더를 정의한다.

따라서, train, valid, test의 3가지 종류의 text 파일을 달리 하면서, 해당 명령어를 실행해야 한다.

```
>>python3 crop_make_folder.py
```

출

각각의 train, valid, test의 하위 폴더에 이미지 라벨별 폴더 이름으로, 그 라벨에 속하는 crop image들이 위치하게 된다. 원본 이미지에서 annotation영역을 crop하고 검은색 바탕

력  
정  
보

이미지 중앙에 그 crop 이미지를 위치 시켜서 하나의 이미지를 만든다.

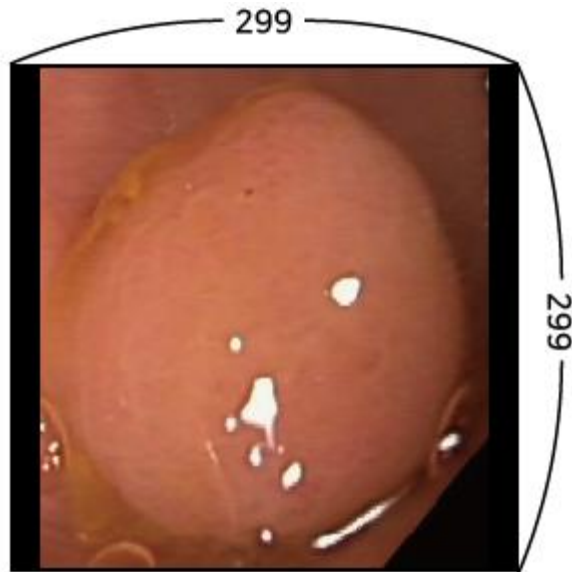
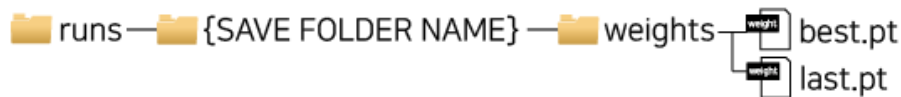


그림 5-2 crop 이미지

### 3. Detect Model

#### A. yolov5

yoloV5	
목적	객체를 탐지하여 해당 객체의 분류 값, confidence score 및 bounding box의 좌표 값을 추론하는 것이 그 목적이다.
준비	<ul style="list-style-type: none"> <li>■ YOLO v5 package</li> <li>■ Custom Dataset(Image 와 Annotation 파일) (Annotation 파일은 txt 파일로, center x, y 값과 width, height 의 상대 좌표 값이다.) (학습 이미지 1 장 당 Annotation 텍스트 파일 1 개가 존재하며, 이미지와 Annotation 파일은 모두 동일 폴더에 존재해야 한다.)</li> <li>■ training 과 validation 에 사용되는 이미지 목록 텍스트 파일 (train.txt, val.txt)</li> <li>■ Class 이름과 Class 개수, train.txt 와 val.txt 경로 정보가 들어있는 .yaml 파일 (그림 3-1 참고)</li> </ul> <pre>names: - Cancer nc: 1 train: /home/aigpuserver/data/yolo_dataset/c18/train.txt val: /home/aigpuserver/data/yolo_dataset/c18/val.txt</pre> <p>그림 3-1 yoloV5 yaml 파일 구조</p>
학습 방법	<ol style="list-style-type: none"> <li>1. Install_dc.sh 를 실행하여 YOLO v5 환경을 갖춘다. (pip install -r requirements.txt)</li> <li>2. Weights 폴더 내의 download_weights.sh 를 실행하여 pre-trained 된 weight 파일을 다운로드한다.</li> <li>3. write_train_val.py 를 실행하여 train.txt 와 val.txt 를 생성한다.</li> <li>4. 3 번에서 생성한 train.txt, val.txt 의 절대 경로 및 분류 라벨 개수, 분류 라벨명이 적힌 yaml 파일을 작성한다.</li> <li>5. training_dc.sh 를 실행하여 학습을 진행한다.</li> </ol>

	<p>(Image size, batch size, epoch number, model type 등의 설정이 가능하다.)</p> <pre>&gt;&gt;python3 train.py --img 416 --batch 512 --epochs 300 --data {YOUR .yaml file}.yaml --cfg ./models/yolov5x.yaml --weights ./weights/yolov5x.pt --name {SAVE FOLDER NAME}</pre>
학습 결과	<p>runs 폴더 내에 학습 결과 폴더가 생성되고, 해당 폴더 내에 weights 폴더가 생성된다. weights 폴더 내부에 best.pt 와 last.pt 가 생성된다.(그림 3-2 참고) best.pt 는 학습 중에 진행된 validation 결과가 가장 우수한 epoch 의 weight 파일이다. last.pt 는 가장 마지막에 진행된 epoch 의 weight 파일이다.</p>  <pre> runs — {SAVE FOLDER NAME} — weights — best.pt  last.pt </pre> <ul style="list-style-type: none"> <li>* best.pt : 학습 중에 진행된 validation 결과가 가장 우수한 epoch의 weight 파일</li> <li>* last.pt : 가장 마지막에 진행된 epoch의 weight 파일</li> </ul> <p>그림 3-2 weight file 저장 경로</p>
추론	<p>inference_dc.sh 를 실행하여 실제 데이터에서 객체를 탐지할 수 있다. confidence score 의 저장 여부, 출력 결과 타입, 학습한 weight 파일 등이 설정 가능하다.</p> <pre>&gt;&gt;./inference_dc.sh</pre> <p>Inference 폴더 내의 output 폴더에 추론 결과가 생성된다. 추론에 사용된 이미지 파일에 bounding box 가 그려진 이미지 파일과, 이에 따른 bounding box 의 좌표, confidence score 정보가 들어있는 텍스트 파일이 생성된다. 만약 이미지 내에서 어떠한 객체도 탐지하지 못한 경우에는 텍스트 파일이 생성되지 않는다.</p>

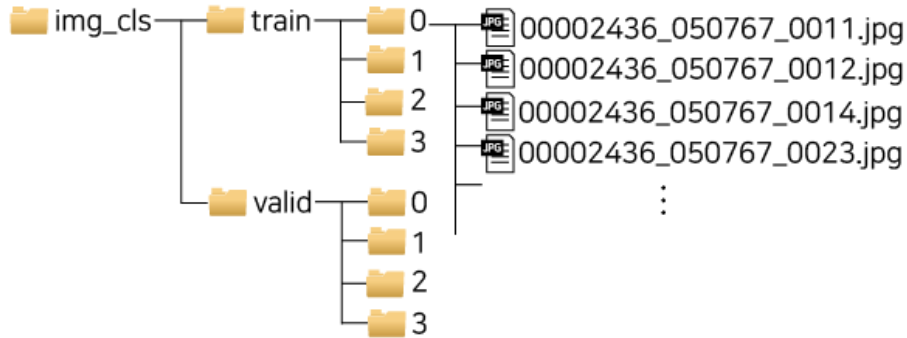
## 4. Classification Model

### A. train classification model

**Inception V(img\_cls16.py, img\_cls18.py)**

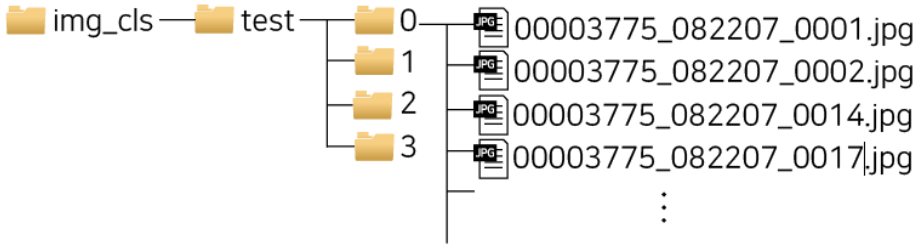
**ResNet(img\_cls16\_resnet.py, img\_cls18\_resnet.py)**

**EfficientNet(eff\_train\_c16\_dc.py, eff\_train\_c18\_dc.py)**

목적	crop image 를 분류하는데 사용될 학습 모델을 훈련시키는 것이 그 목적이다.
입력 정보	<p>train에 속하는 이미지 폴더 구조, valid에 속하는 폴더 구조를 읽어서 처리한다. train과 valid 폴더의 구조는 아래의 그림 4-1 예시를 따르면 된다.</p>  <p style="text-align: center;">그림 4-1 classification dataset 구조</p>
실행 방법	<p>분류될 라벨의 종류의 개수, 이미지 크기 정보, train 폴더 이름, valid 폴더 이름의 정보들을 결정을 한다. 그리고, 기타 표준 모델을 학습하기 위한 정해진 학습 파라미터를 설정을 한다.</p> <pre> 21 # /home/joynhyuk/finat_hyuk/a_data_18/train 22 img_list_0 = glob.glob("./a_data_18/train/0/*.jpg") 23 img_list_1 = glob.glob("./a_data_18/train/1/*.jpg") 24 img_list_2 = glob.glob("./a_data_18/train/2/*.jpg") 25 img_list_3 = glob.glob("./a_data_18/train/3/*.jpg") 26 </pre> <p>각각의 학습 폴더에 있는 클래스 별 개수를 구하기 위한 코드</p> <pre> vimg_list_0 = glob.glob("./a_data_18/valid/0/*.jpg") vimg_list_1 = glob.glob("./a_data_18/valid/1/*.jpg") vimg_list_2 = glob.glob("./a_data_18/valid/2/*.jpg") vimg_list_3 = glob.glob("./a_data_18/valid/3/*.jpg") </pre> <p>각각의 검증 폴더에 있는 클래스 별 개수를 구하기 위한 코드,</p>

	<pre>cnt_0 = len(img_list_0) cnt_1 = len(img_list_1) cnt_2 = len(img_list_2) cnt_3 = len(img_list_3)</pre> <pre>vcnt_0 = len(vimg_list_0) vcnt_1 = len(vimg_list_1) vcnt_2 = len(vimg_list_2) vcnt_3 = len(vimg_list_3)</pre> <pre>total_train = cnt_0+cnt_1+cnt_2+cnt_3 total_valid = vcnt_0+vcnt_1+vcnt_2+vcnt_3</pre> <p>위 코드를 바탕으로 최종 학습에 사용된 총 파일 개수, 검증에 사용되는 총 파일 개수를 구하게 된다.</p> <pre>n_classes = 4 img_width, img_height = 299, 299 train_data_dir = 'a_data_18/train' validation_data_dir = 'a_data_18/valid' nb_train_samples = total_train # 4730 # 2250 #75750 nb_validation_samples = total_valid #526 # 750 #25250 batch_size = 32</pre> <p>최종적으로, 위에서 보이는 값, 분류할 클래스의 총 개수, 이미지의 width, height의 값, 학습 데이터 폴더 이름, 검증 데이터 폴더 이름, 학습에 사용할 총 파일 개수, 검증에 사용할 총 파일 개수, 그리고 배치 파일을 결정해서, 아래와 같은 명령어를 수행하면 된다.</p> <pre></pre> <p>나머지 resnet, efficient net도 해당 정보를 수정해서 해당 명령어를 수행하면 된다.</p>
출력 정보	<p>학습 결과로서 학습 모델 weights정보가 생성이 된다.</p> <p>ex)</p> <p>'resnet_16_best_model_4class.hdf5'의 이름으로 학습 결과 weights가 저장된다. (resnet 모델의 경우에)</p>

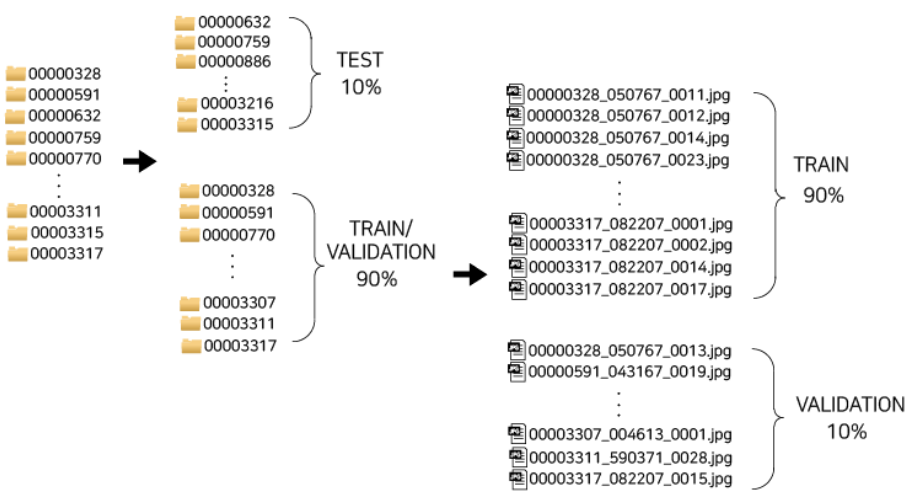
## B. inference classification model

<b>Inception V(in_16_pred.py, in_18_pred.py)</b> <b>ResNet(res_16_pred.py, res_18_pred.py)</b> <b>EfficientNet(eff_16_pred.py, eff_18_pred.py)</b>	
목적	각각의 학습 모델 weight 파일을 읽은 다음에, 해당 폴더의 test 에 있는 이미지들을 읽어서, 추론 테스트를 하는 코드이다.
입력 정보	<p>분류 모델 weight 파일, test용으로 사용될 이미지 파일들을 입력정보로 받는다. 이미지 파일의 구조는 아래의 그림 4-2 예시를 따르면 된다.</p>  <p style="text-align: center;">그림 4-2 classification test dataset 구조</p>
실행 방법	<p>test용으로 사용될 이미지들이 준비되어 있어야 한다.</p> <pre>K.clear_session() model_best = load_model('resnet_16_best_model_4class.hdf5', compile = False)</pre> <p>위에서 학습 결과로 나온 학습 모델 weight 이름을 지정한다.</p> <pre>num_of_class = 4</pre> <p>클래스 개수를 설정한다.</p> <pre>folder_name = "./c_data3/test/"</pre> <p>테스트 할 폴더 이름을 지정한다.</p> <pre>csv_df.to_csv("res_16_final.csv", index=None)</pre> <p>테스트 결과를 저장할 csv파일 이름을 지정한다.</p> <p>그런 다음에 아래와 같이 명령어를 실행한다.</p> <pre>&gt;&gt;python3 res_16_pred.py</pre>
출력 정보	해당 이미지를 분류해서, 정답을 맞춘 개수와 틀린 개수를 계산해서, 정확도를 출력한다.

	<p>ex)</p> <p>inception (c16) ~&gt; 위암인 경우 예시 모델 Inception</p> <p>- T0 ~ T3(class 4)</p> <p>개별 평가</p> <p>true : 346</p> <p>false : 227</p> <p>total : 573</p> <p>정확도(accuracy) : 60.38%</p> <p>가장 많이 받은 라벨로 평가</p> <p>true : 29</p> <p>false : 13</p> <p>정확도(accuracy) : 69.04%</p>
--	---



## 5. 검증 및 평가 방법

검증 및 평가 방법	
dataset	 <p>그림 6-1 평가 dataset 구조</p> <ul style="list-style-type: none"> <li>- 전체 RID중에 테스트용 RID를 랜덤으로 선택, 그 외는 학습용으로 선택</li> <li>- 학습용 RID의 이미지를 랜덤하게 train/validation으로 구분하여 학습 수행( 9 : 1 )</li> </ul>
추론	<ul style="list-style-type: none"> <li>- 학습 완료된 모델에 테스트용 RID들의 이미지들을 입력하여 각 이미지들의 추론 결과를 얻음 (Detect -&gt; Crop -&gt; Classify)</li> <li>- 한 RID(환자)내의 이미지들의 추론 결과를 종합하여 다수결로 그 환자의 암 병기를 결정함</li> </ul> <p>예) 추론 결과 1기 이미지가 2장, 2기 이미지가 3장이라면 해당 RID 환자는 최종 2기로 판정함.</p>