2021 미소 인공지능 모델 개발 챌린지

근골격 데이터

데이터 및 코드 소개

1.Overview

- 대회 주제 : 근골격 퇴행성 질환의 필수측정값을 정확하게 측정하는 AI 모델 개발
  1) 요추 일반촬영 데이터를 이용한 추간판간격 및 분절각도 측정 AI 모델 개발
  2) 무릎 일반탈영 데이터를 이용한 관절간격 측정 AI 모델 개발

- 목적 : 퇴행성질환에서 질환의 진행 정도를 평가하고 추후 치료방침을 결정하기 위해서는 몇가지 중요한 수치들의 측정이 필요하다. 하지만 이를 매번 사람의 손으로 측정하기 때문에, 측정자나 측정도구에서의 오차를 피하기 어렵다. 본 대회에서 제공되는 데이터는 한국인에서 가장흔한 퇴행성 질환인 요추 추간판 질환과 무릎의 골관절염 환자의 일반촬영 데이터로, 필수 측정값을 자동으로 측정하는 모델을 개발하여 실제 진료현장에 적용할 수 있는 가능성을 확인해보고자 한다.

- 대회방식

1. 참가팀은 제공된 두개의 데이터 셋을 모두 활용하여 **AI 모델을 개발**합니다. 하나의 공통된 모델을 개발하거나 두개의 모델을 별도로 개발하여도 상관없지만 모델 성능은 각각의 데이터에 대해 평가합니다.
2. 대회 종료 시 **개발된 AI 모델**과 규정된 양식의 **결과 요약지**를 이용하여 모델 설명 및 자체 성능 평가 결과를 제출합니다. (결과 요약지는 대회 Github 에서 다운로드 가능 https://github.com/DatathonInfo/MISOChallenge-musculoskeletal)
3. 제출한 AI 모델을 이용하여 **주최측에서 테스트셋으로 성능 평가를 실시**하고, **AI 모델의 성능 및 우수성을 평가**하여 **대상(1팀), 최우수상(1팀), 우수상(1팀)을 선정**하여 시상이 진행됩니다. (평가 결과는 공개되지 않습니다)

- 지원사항

원활한 학습을 위해 각 참가팀별 네이버클라우드 GPU서버가 제공됩니다.
(Nvidia Tesla P40 (2GPU), GPU 메모리 48GB, vCPU 8개, 메모리 60GB, 디스크 100GB SSD)

1.Overview

결과 요약지

대회명: 근골격 데이터

참가팀명		팀원수	
선택 주제			
모델 설명	개발 모델에 대한 간략한 설명을 적어주세요.		
성능 평가 결과	모델의 성능 Evaluation 결과 (성능 평가는 주최측에서 제시한 평가 기준을 사용하여 측정하시고, 평가 결과에 대한 설명이나 스크린샷 첨부 필수)		
기타 사항	(추가 의견이나 설명하고 싶은 내용이 있을 시 자유롭게 기술해주세요)		

※ 양식 제한 및 장수 제한 없음

2.Data Information

- 데이터 구조

<table>
<tr><th colspan="2">DIRECTORY</th><th>FILES</th><th>COUNT</th><th>CONTENTS</th></tr>
<tr><td rowspan="4">/SPINE</td><td rowspan="2">/train</td><td>/*.dcm</td><td>240</td><td rowspan="8">• Test 데이터는 참가자들에게 제공되지 않음.</td></tr>
<tr><td>/*.json</td><td>240</td></tr>
<tr><td rowspan="2">/test</td><td>/*.dcm</td><td>50</td></tr>
<tr><td>/*.json</td><td>50</td></tr>
<tr><td rowspan="4">/PAIN</td><td rowspan="2">/train</td><td>/*.dcm</td><td>240</td></tr>
<tr><td>/*.json</td><td>240</td></tr>
<tr><td rowspan="2">/test</td><td>/*.dcm</td><td>56</td></tr>
<tr><td>/*.json</td><td>56</td></tr>
</table>

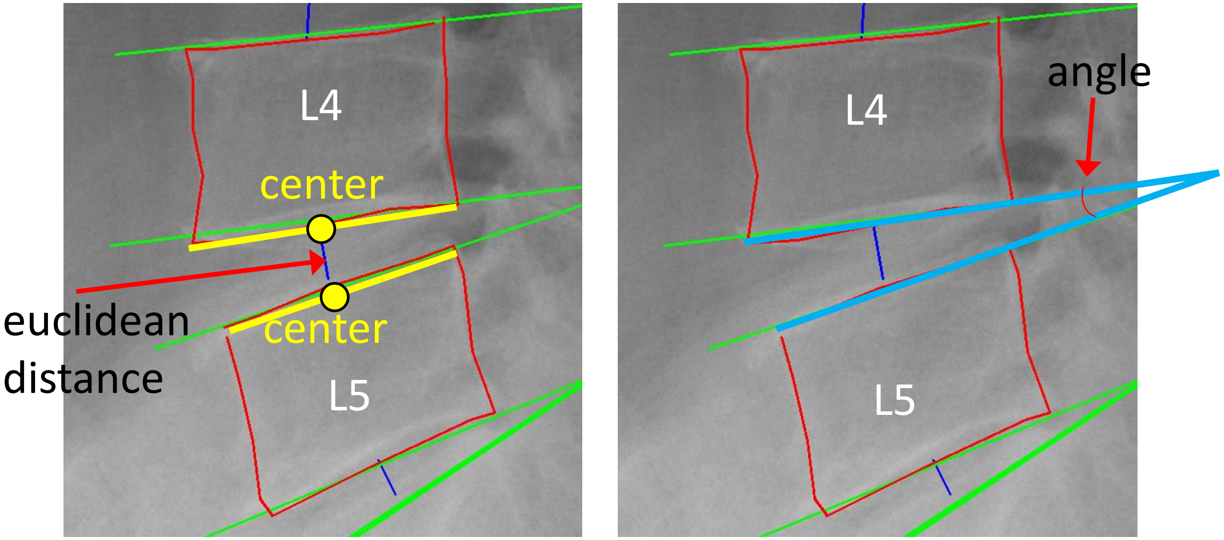
2.Data Information

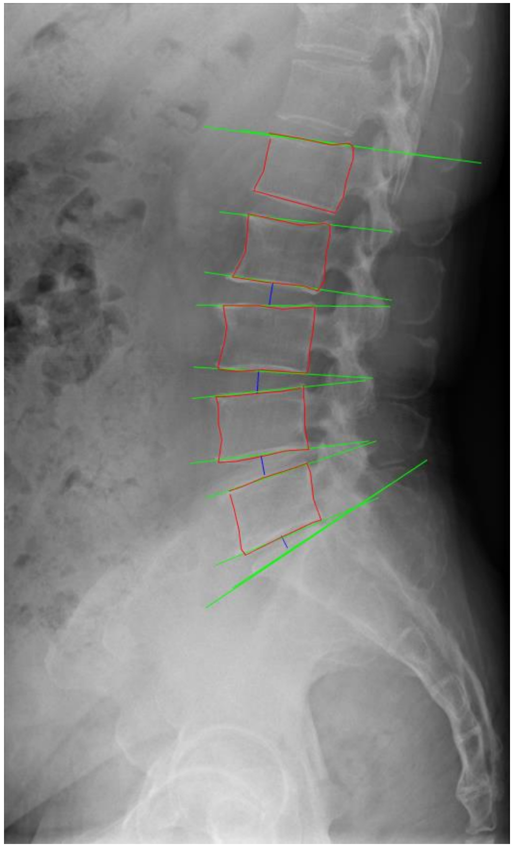
데이터 : 척추 X-ray 영상

목표 : 딥러닝을 이용하여 척추체를 분할 후, 영상처리 알고리즘을 이용하여 아래 두개의 정량적 지표를 측정하여 제시함.

- ✓ **척추 L4번과 L5번 사이의 간격 (디스크 높이)**
- ✓ **척추 L4번과 L5번 척추체의 분절각도**

디스크 높이 **각도**





2.Data Information

데이터 : 척추 X-ray 영상

- **제공 : 익명화된 DICOM 데이터 + JSON 라벨링 파일**
- **총 제공 데이터 : 척추 퇴행성 질환 290례 (Train : 240례, Test : 50례)**


```
{"annotation":
    {"DATA_CATEGORY":[],
    "diseases_category":"DLD",
    "ANNOTATION_DATA":[
        {"vs":{"x":910,"y":1750},"ve":{"x":929,"y":1788},"distMm":42.485291572496,"label":"L5-S1H","id":6,"type":"line"},
        {"vs":{"x":843,"y":1486},"ve":{"x":854,"y":1548},"distMm":62.96824596572466,"label":"L4-5H","id":7,"type":"line"},
        {"vs":{"x":835,"y":1204},"ve":{"x":830,"y":1282},"distMm":78.16009211867653,"label":"L3-4H","id":8,"type":"line"},
        {"vs":{"x":881,"y":921},"ve":{"x":870,"y":991},"distMm":70.8590149522275,"label":"L2-3H","id":9,"type":"line"},

        {"points":[{"x":776,"y":417},{"x":1430,"y":509},{"x":657,"y":883},{"x":1271,"y":975}],"angle":0.514225665444721,"label":"L1-2A","id":11,"type":"cobbAngle"},
        {"points":[{"x":708,"y":686},{"x":1274,"y":751},{"x":622,"y":1195},{"x":1209,"y":1231}],"angle":3.041719239870207,"label":"L2-3A","id":12,"type":"cobbAngle"},
        {"points":[{"x":630,"y":991},{"x":1266,"y":996},{"x":611,"y":1511},{"x":1196,"y":1438}],"angle":7.56339093208983,"label":"L3-4A","id":13,"type":"cobbAngle"},
        {"points":[{"x":617,"y":1298},{"x":1185,"y":1239},{"x":695,"y":1845},{"x":1228,"y":1624}],"angle":16.59033472847627,"label":"L4-5A","id":14,"type":"cobbAngle"},
        {"points":[{"x":660,"y":1626},{"x":1220,"y":1438},{"x":751,"y":1920},{"x":1269,"y":1578}],"angle":14.87641824614882,"label":"L5-S1A","id":15,"type":"cobbAngle"},
        {"points":[{"x":657,"y":406},{"x":1565,"y":525},{"x":662,"y":1985},{"x":1387,"y":1500}],"angle":41.247632074052845,"label":"L1-S1A","id":16,"type":"cobbAngle"},

        {"m_isClosed":true,"m_points":[{"x":741,"y":1613},{"x":754,"y":1659},{"x":770,"y":1729},{"x":778,"y":1793},{"x":792,"y":1812},{"x":900,"y":1758},{"x":1013,"y":1705},{"x":1040,"y":1697},{"x":101
            "m_area":54476,"label":"척추체B/L","id":1,"type":"poly"},
        {"m_isClosed":true,"m_points":[{"x":983,"y":1256.7142857142858},{"x":983,"y":1292},{"x":986,"y":1330},{"x":986,"y":1389},{"x":994,"y":1435},{"x":999,"y":1465},{"x":924,"y":1470},{"x":854,"y'
            "m_area":58569.07142857136,"label":"척추체B/L","id":2,"type":"poly"},
        {"m_isClosed":true,"m_points":[{"x":1020.28,"y":1000.56},{"x":1013,"y":1077},{"x":1002,"y":1139},{"x":1002,"y":1185},{"x":997,"y":1201},{"x":994,"y":1214},{"x":945,"y":1212},{"x":846,"y":1204
            "m_area":61102.24000000005,"label":"척추체B/L","id":3,"type":"poly"},
        {"m_isClosed":true,"m_points":[{"x":802.9370629370629,"y":692.7342657342657},{"x":797,"y":762},{"x":792,"y":818},{"x":768,"y":864},{"x":749,"y":899},{"x":854,"y":915},{"x":983,"y":929},{"x
            "m_area":56512.60139860143,"label":"척추체B/L","id":4,"type":"poly"},
        {"m_isClosed":true,"m_points":[{"x":874.9475890985325,"y":446.8008385744235},{"x":865,"y":482},{"x":851,"y":552},{"x":824,"y":603},{"x":819,"y":616},{"x":916,"y":643},{"x":1045,"y":678},{"
            "m_area":58165.912997903564,"label":"척추체B/L","id":5,"type":"poly"}
        ],
```

2.Data Information

데이터 : 척추 X-ray 영상

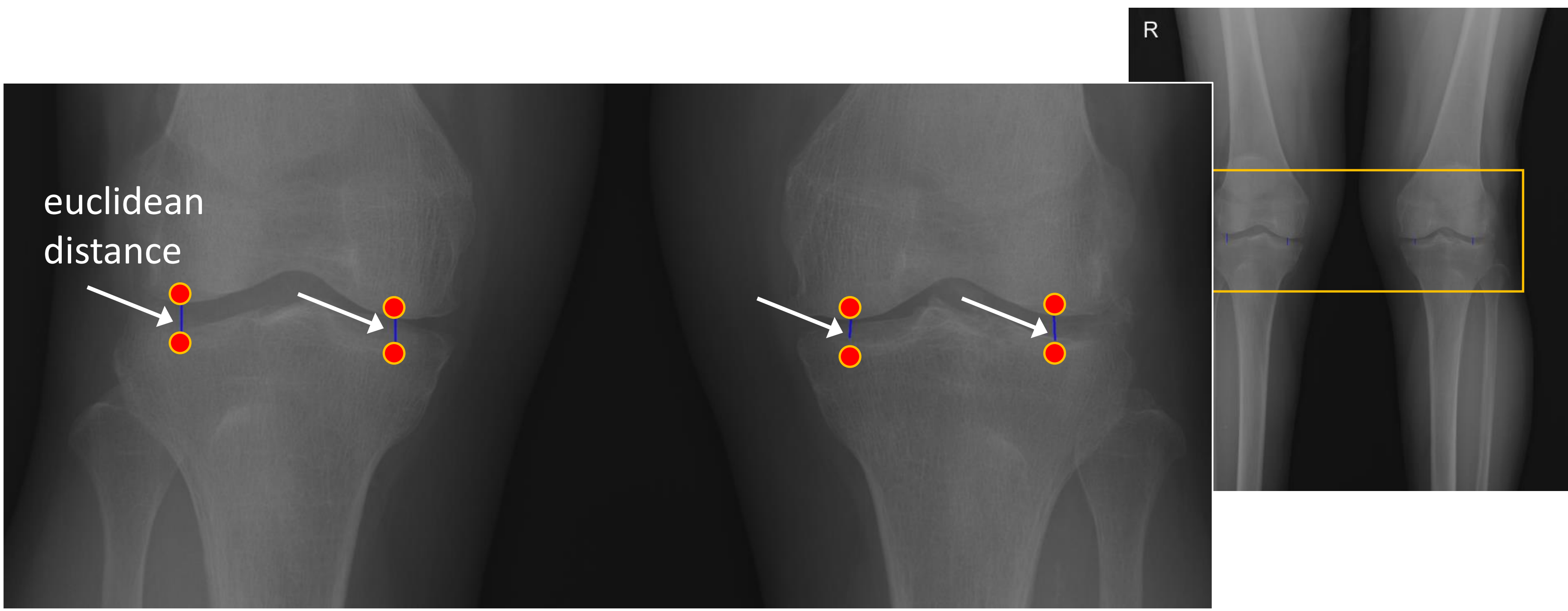
"clinic":
{"sex":"F",
"name":"DLD",
"id":"00000024",
"age":68}
},
"Dataset":
{"identifier":"SPINE_01",
"name":"SPINE_01_DLD_C001_00000024_6_00001",
"diseases":3,"src_path":"/SPINE/01/DLD/00000024/C001/SPINE_01_DLD_C001_00000024_6_00001.dcm",
"label_path":"/SPINE/01/DLD/00000024/C001/SPINE_01_DLD_C001_00000024_6_00001.json",
"category":1,
"type":0
},
"Images":
{"identifier":"SPINE_01_DLD_C001_00000024_6_00001.dcm",
"width":1660,
"type":"dcm",
"dataCaptured":"20200101",
"height":2761}}

2.Data Information

데이터 : 무릎 X-ray 영상

목표 : 딥러닝을 이용하여 8개의 landmark의 위치를 검출하고, 영상처리 알고리즘을 이용하여 아래의 정량적 지표를 측정하여 제시함.

- ✓ **좌우 각각에서의 내측 관절 간격, 외측 관절 간격**



2.Data Information

데이터 : 무릎 X-ray 영상

- **제공 : 익명화된 DICOM 데이터 + JSON 라벨링 파일**
- **총 제공 데이터 : 무릎 퇴행성 질환 300례**


```
{"annotation":
        {"DATA_CATEGORY":[
                {"lipping":1},
                {"carbonehard":2},
                {"meniscuslt":3},
                {"meniscusrup":3},
                {"meniscusana":3},
                {"bonemedema":3},
                {"pleuralefluid":3}
                ],
        "diseases_category":"DJDK",
        "ANNOTATION_DATA":[
                {"vs":{"x":608,"y":1396},"ve":{"x":608,"y":1453},"distMm":57,"label":"관절간격수치-외측","id":1,"type":"line"},
                {"vs":{"x":981,"y":1424},"ve":{"x":981,"y":1468},"distMm":44,"label":"관절간격수치-내측","id":2,"type":"line"},
                {"vs":{"x":1775,"y":1429},"ve":{"x":1773,"y":1460},"distMm":31.064449134018133,"label":"관절간격수치-내측","id":3,"type":"line"},
                {"vs":{"x":2129,"y":1422},"ve":{"x":2131,"y":1465},"distMm":43.04648650006177,"label":"관절간격수치-외측","id":4,"type":"line"}
                ],
        "clinic":{
                "sex":"F",
                "name":"DJDK",
                "id":"00000011",
                "age":70}
                },
        "Dataset":{
                "identifier":"PAIN_01",
                "name":"PAIN_01_DJDK_C001_00000011_4_00001",
                "diseases":0,"src_path":"/PAIN/01/DJDK/00000011/C001/PAIN_01_DJDK_C001_00000011_4_00001.dcm",
                "label_path":"/PAIN/01/DJDK/00000011/C001/PAIN_01_DJDK_C001_00000011_4_00001.json",
                "category":1,
                "type":0
                },
        "Images":{"identifier":"PAIN_01_DJDK_C001_00000011_4_00001.dcm",
                "width":2802,
                "type":"dcm",
                "dataCaptured":"20190101",
                "height":2991}}
```

3.Baseline Code - Spine

0.Main

```
def main():
    parser = argparse.ArgumentParser(description='Parser test input uid')
    parser.add_argument('--train_path', type = str, default ='spine/train/', help='학습데이터 위치')
    parser.add_argument('--test_path', type = str, default ='spine/test/', help='테스트데이터 위치')
    parser.add_argument("--image_size",type= int , default= 512, help='학습에 사용될 이미지의 크기')
    parser.add_argument("--epochs",type= int , default= 10, help='에폭')
    parser.add_argument("--batch_size",type= int , default= 4, help='배치사이즈')
    args = parser.parse_args()

    imgs_train, imgs_mask_train, imgs_name = train_data_loading(args.train_path, image_size = args.image_size)
    model = deep(imgs_train,imgs_mask_train,args.train_path,batch_size = args.batch_size,epochs = args.epochs,image_size=args.image_size)
    test_name = predict_val(model,args.test_path,image_size = args.image_size)
    angle_list,dist_list = get_results(test_name)
    get_score(angle_list,dist_list,test_name)
```

①

②

① 코드 실행시 commandline argument를 이용해 수치를 입력받는 부분.

- Argparse.ArgumentParser() : parser를 생성하는 함수
- Add_argument() : 입력받고자 하는 argument의 형식과 조건을 설정
- Parse_args() : 입력받은 argument를 변수에 저장하여 이후에 사용할 수 있도록 함.

② 본격적인 processing을 실행하는 함수들

- Train_data_loading : 학습에 사용할 데이터들을 불러오고 전처리 하는 함수
- Deep : 딥러닝 학습 모델의 설정 및 학습을 진행하는 함수
- Predict_val : 검증용 데이터를 이용하여 학습된 모델의 성능을 검증하는 함수
- Get_results : 학습의 결과물을 이용하여 4-5번 척추의 거리와 각도를 측정하는 함수
- Get_score : 실측된 결과와 모델의 결과로 검증한 결과의 오차를 비교하는 함수

3.Baseline Code - Spine

1. Train Data Load

```
def train_data_loading(path,image_size = 512):
    mkfolder(path)
    trainlist = glob.glob(path+'/*.dcm')
    data_pre(trainlist)
    train_img_path = path+'/*.jpg'
    train_mask_path = path+'/*.png'
    aug_path = augmentation(train_img_path,train_mask_path)
    imgs_train, imgs_mask_train, imgs_name = create_train_data(aug_path, image_size, image_size, 'train', 'jpg')
    return imgs_train, imgs_mask_train, imgs_name
```

data_pre : 입력된 데이터들의 전처리 함수

augmentation : 적은 학습 데이터의 한계를 극복하기 위해 임의로 학습용 데이터를 변형시켜 증식시키는 함수

create_train_data : 전처리가 완료된 이미지들을 stack 하여 하나의 데이터 array로 묶어주는 함수.

3.Baseline Code - Spine

1-1). data_prepocess

```python
def data_pre(dcmlist):
    for i in range(len(dcmlist)):
        try:
            dicompath = dcmlist[i]
            dicom = pydicom.read_file(dicompath)
            img = dicom.pixel_array
            img = cv2.normalize(img, None, 0, 255, cv2.NORM_MINMAX)
        except:
            print(dicompath)
            continue
        pad_img = zero_padding(img)
        image = Image.fromarray(pad_img)
        image = image.convert('L')
        image.save(dicompath.replace('.dcm','.jpg'))
```

```python
for dcm in dcmlist:
    jsonfile = dcm[:-4]+'.json'
    reader = sitk.ReadImage(dcm)
    image_array = sitk.GetArrayFromImage(reader)
    height = reader.GetMetaData('0028|0010')
    width = reader.GetMetaData('0028|0011')
    data = []
    for line in open(jsonfile,'r'):
        data.append(json.loads(line))
    for json_data in data:
        mask = np.zeros((int(height), int(width)))
        if json_data['annotation']['ANNOTATION_DATA'] is not None:
            for m in json_data['annotation']['ANNOTATION_DATA']:
                if 'm_points' in m:
                    a = []
                    for i in m['m_points']:
                        b = (i['x'], i['y'])
                        a.append(b)
                    r = LinearRing(a)
                    s = Polygon(r)
                    x, y = s.exterior.coords.xy
                    maskd = poly2mask(y, x, (int(height), int(width)))
                    mask = mask + maskd
            mask = mask*255
            mask = zero_padding(mask)
            mask = np.expand_dims(mask, axis=0)
            img = sitk.GetImageFromArray(mask.astype('uint8'))
            num = 0
            maskpath = dcm[:-4]+'.png'
            sitk.WriteImage(img, maskpath)
        else:
            print('haha')
```

① Dicom을 불러와 pixel array로 만든 후 픽셀값의 범위를 0~255로 normalize 해주기

② 이미지의 넓이와 높이를 동일하게 맞추기 위해서 둘중 더 좁은 방향에 0값을 채워서 padding 후 .jpg 확장자로 저장함.

③ Json데이터의 내용을 읽은 후 m_points라는 이름으로 지정된 좌표 값들을 탐색(해당 값들이 척추의 roi 좌표이다.)

④ 탐색한 좌표들을 연결 후 (ⓐ) 연결된 값의 안쪽을 1, 바깥쪽을 0으로 하는 mask 생성

⑤ 생성된 마스크에 Dicom image와 동일하게 Zero padding 실행 후, .png 확장자로 저장.

3.Baseline Code - Spine

1-2). Augmentation

```python
def Augment_crop(img, mask):
    p_x, p_y=find_top_point(img, mask)
    rotate_img, rotate_mask = randomRoate(img, mask, (p_x, p_y), 20)
    random_size = np.random.randint(15,35)*20 # 300-600

    h, w= img.shape
    x1 = p_x-random_size if p_x-random_size>0 else 0
    x2 = p_x+random_size if p_x+random_size<w else w
    y1 = p_y-random_size if p_y-random_size>0 else 0
    y2 = p_y+random_size if p_y+random_size<h else h

    crop_img = rotate_img[y1:y2,x1:x2]
    crop_mask = rotate_mask[y1:y2,x1:x2]

    return crop_img, crop_mask
```

```python
def augmentation(img_path,mask_path):
    img_li = sorted(glob.glob(img_path))
    mask_li = sorted(glob.glob(mask_path))
    print(len(img_li), len(mask_li))
    i=0

    for img, mask in zip(img_li, mask_li):
        if i%100==0:
            print('{}/{}'.format(i, len(img_li)))
        savepath = 'spine/train/aug/'
        mkfolder(savepath)
        ori_img = cv2.imread(img, 0)
        mask_img = cv2.imread(mask, 0)
        img_name = img[img.rindex('/')+1:-4]
        mask_name = img[mask.rindex('/')+1:-4]
        print(img_name)
        print(mask_name)
        cv2.imwrite(savepath+'/{}.jpg'.format(img_name), cv2.resize(ori_img, (512,512)))
        cv2.imwrite(savepath+'/{}.png'.format(img_name), cv2.resize(mask_img, (512,512)))
        for j in range(9):
            aug_img, aug_mask = Augment_crop(ori_img, mask_img)
            aug_img = cv2.resize(aug_img, (512,512))
            aug_mask = cv2.resize(aug_mask, (512,512))
            cv2.imwrite(savepath+'/{}_{}.jpg'.format(img_name, j), aug_img)
            cv2.imwrite(savepath+'/{}_{}.png'.format(img_name, j), aug_mask)
        i+=1
    return savepath
```

① 이미지와 마스크를 변형하고(Augment_crop), 그 변형된 결과물의 사이즈를 512로 맞추어 저장하는것을 9회 반복한다. -> 총 10배의 데이터 증식 효과.

② 입력된 데이터를 ±20 사이의 각도로 회전시키고, 300-600 사이의 픽셀 사이즈로 임의로 데이터 변형.

③ 변형된 데이터에서 임의의 위치를 지정하여 데이터를 Crop

3.Baseline Code - Spine

1-3). create_train_data

```python
def create_test_data(test_path, out_rows, out_cols, name, img_type):

    print('-'*30)
    print('Creating test images...')
    print('-'*30)

    i = 0
    imgs = glob.glob(test_path + "*." + img_type)
    imgdatas = np.ndarray((len(imgs),out_rows,out_cols,1), dtype=np.uint8)
    imglabels = np.ndarray((len(imgs),out_rows,out_cols,1), dtype=np.uint8)

    imgnames=[]
    for j, imgname in enumerate(imgs):
        if j%100==0:
            print('{}/{}'.format(j, len(imgs)))
        midname = imgname[imgname.rindex("/")+1:-4]
        img = load_img(imgname, color_mode = "grayscale")
        label = load_img(imgname.replace('jpg', 'png'), color_mode = "grayscale")
        img=img.resize((out_rows,out_cols))
        label=label.resize((out_rows,out_cols))

        img = img_to_array(img)
        label = img_to_array(label)
        imgdatas[j] = img
        imglabels[j] = label
        imgnames.append(midname)
```

```python
    imgdatas = imgdatas.astype('uint8')
    imglabels = imglabels.astype('uint8')

    print('img : ', imgdatas.max())
    print('mask : ',imglabels.max())

    print('-'*30)
    print('normalization start...')
    print('-'*30)
    imgdatas = imgdatas/255.0

    imglabels[imglabels <= 127] = 0
    imglabels[imglabels > 127] = 1

    print('img : ',imgdatas.max())
    print('mask : ',imglabels.max())
    print('mask : ',imglabels.min())
    print('loading done')
    return(imgdatas,imglabels,imgnames)
```

① Img_type(jpg)에 맞는 파일들을 모두 불러오고, 그 파일의 수 만큼의 수용공간을 가지는 Array를 선언.
- glob은 지정된 경로에서 조건에 맞는 파일들을 모두 탐색해 경로를 넘겨주는 함수.

② 선언한 Array에 image파일(.jpg)과 mask파일(.png)를 불러와 집어넣고, 지정된 사이즈로 변환한다.

③ 이미지가 모두 불러와 지면 0~255의 값을 가지는 각 이미지를 0~1로 normalize 한다.

3.Baseline Code - Spine

2. Deep Learning Model

```
def deep(imgs_train,imgs_mask_train,path,batch_size = 4,epochs = 10,image_size=512):
    model = get_unet(image_size, image_size)
    model.summary()
    model = multi_gpu_model(model,gpus=4)
    model.compile(optimizer=Adam(lr=0.0001), loss=dice_coef_loss,
                    metrics=['accuracy', sens, dice_coef_loss])

    check_model_path = path+'/check/'
    predict_path = path+'/pred/'

    model_checkpoint = ModelCheckpoint(check_model_path+'ap_aug_exp1_{epoch:d}_{loss:f}.hdf5',
                                    monitor='val_dice_coef_loss',verbose=1,
                                    save_best_only=True)
    reduce_lr = ReduceLROnPlateau(monitor='val_dice_coef_loss', factor=0.8, min_delta = 0.01,
                                patience=5, min_lr=1e-6, verbose=1)
    earlystopping = EarlyStopping(monitor='val_dice_coef_loss', patience=10)
```

- get_unet : image_size(512,512)를 input으로 하는 U-net 모델 호출
- Adam optimizer : 이전 optimizer의 장점들을 취해 만들어진 optimizer
- Dice_coef_loss : 두 데이터의 유사성을 측정하는데 사용되는 수치.

3.Baseline Code - Spine

2. Deep Learning Model

```
print('Fitting model...')
model.fit(imgs_train, imgs_mask_train, batch_size=batch_size, epochs=epochs, verbose=1,
          validation_split=0.2, shuffle=True, callbacks=[model_checkpoint, earlystopping])

print('save model')
model.save(predict_path+'ap_aug_exp1.h5')
return model
```

```
Epoch 00001: val_dice_coef_loss improved from inf to 0.53634, saving model to spine/train//check/ap_aug_exp1_1_0.500262.hdf5
Epoch 2/10
1920/1920 [==============================] - 74s 38ms/step - loss: 0.3459 - accuracy: 0.9464 - sens: 0.8889 - dice_coef_loss: 0.3459 - val_loss: 0.6526 - val_accuracy: 0.9010 - val_sens: 0.2955 - val_dice_coef_loss: 0.6526

Epoch 00002: val_dice_coef_loss did not improve from 0.53634
Epoch 3/10
1920/1920 [==============================] - 74s 38ms/step - loss: 0.2543 - accuracy: 0.9609 - sens: 0.8939 - dice_coef_loss: 0.2543 - val_loss: 0.5769 - val_accuracy: 0.9077 - val_sens: 0.3940 - val_dice_coef_loss: 0.5769

Epoch 00003: val_dice_coef_loss did not improve from 0.53634
Epoch 4/10
  12/1920 [..............................] - ETA: 1:03 - loss: 0.1907 - accuracy: 0.9731 - sens: 0.9260 - dice_coef_loss: 0.1907
```

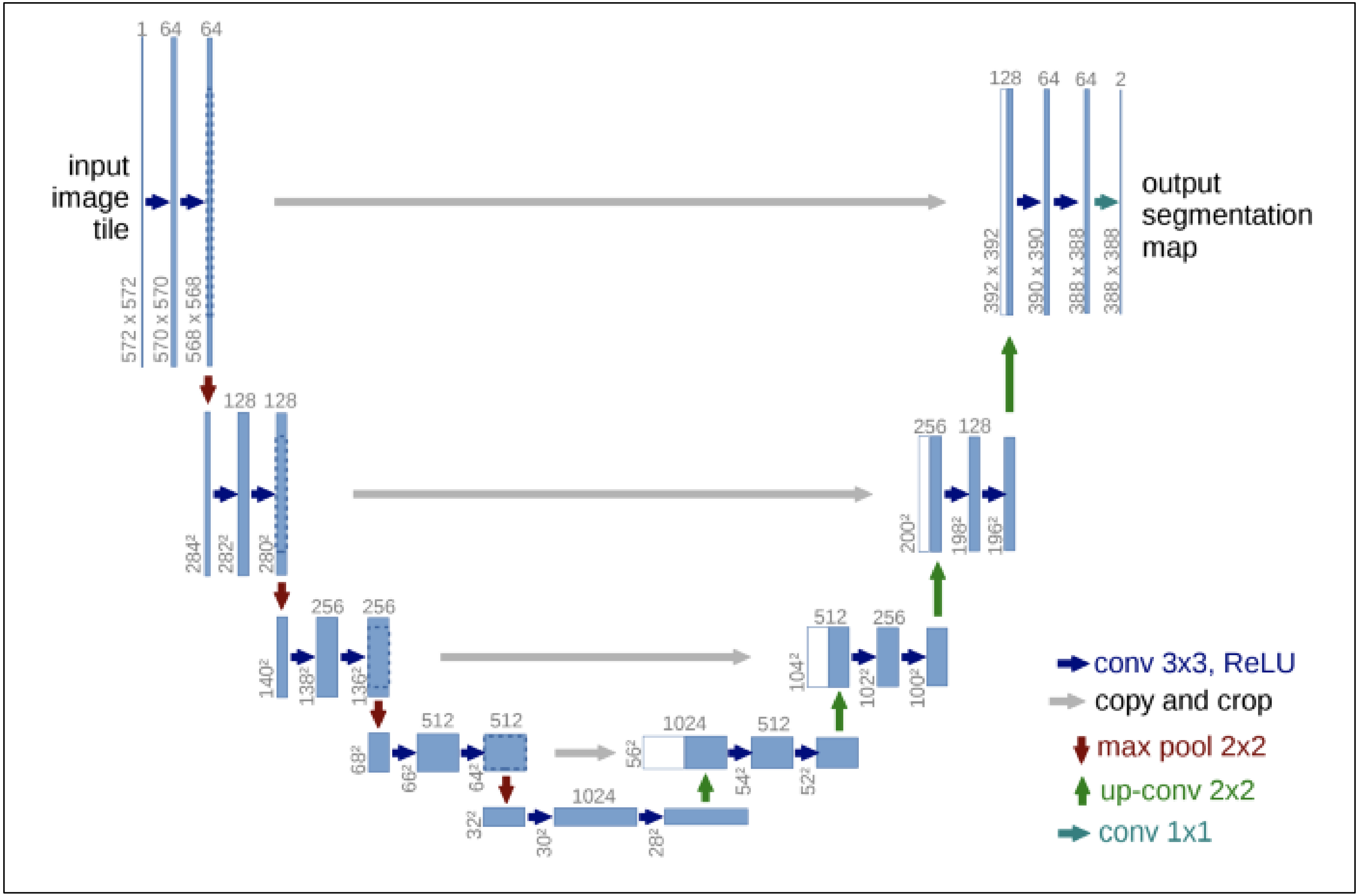
- Model.fit : 앞서 구축한 모델에 데이터를 input시켜 학습하는 명령어
  - Batch_size : 해당 모델에서 동시에 학습하는 데이터의 수
  - Epochs : 전체 training data를 반복해서 학습하는 횟수
  - Validation_split : 전체 training data중 일부분을 분리해 각 epoch 학습결과를 검증함.

- Model_checkpoint : 한 epoch이 끝나고, 그 학습된 중간결과를 저장하는 함수

- Earlystopping : 학습과정을 반복함에도 validation score에 향상이 없을 시, 과적합을 막기 위해서 학습과정을 중단시키는 함수

3.Baseline Code - Spine

2-1).U-net

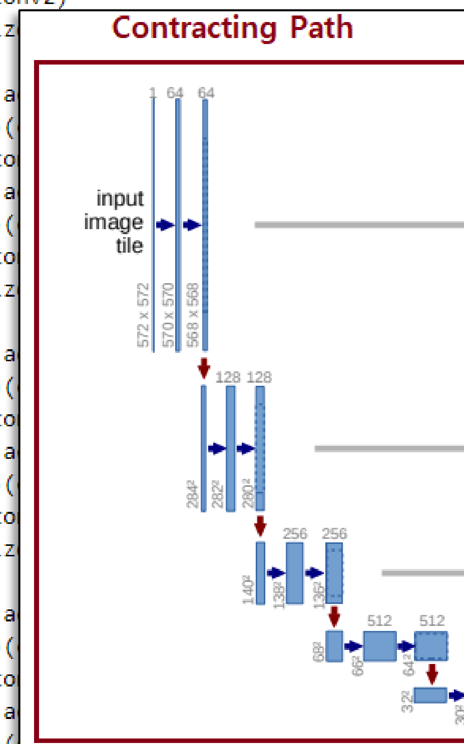


3.Baseline Code - Spine

2-1).U-net

- Conv2D : 3x3의 Kernel_size를 통해 이미지를 sliding window 방식으로 합성곱 계산
- BatchNormalization : 각 feature 별로 평균과 표준편차를 구해준 다음 normalize 해주고, scale factor와 shift factor를 이용하여 새로운 값을 만들어주는 명령어
- ReLU : 출력값이 0 이하이면 모두 0으로 만드는 활성화 함수.
- Maxpooling2D : 지정된 사이즈(2X2)의 필터에서 가장 큰 숫자로 대체하는 subsampling 방식

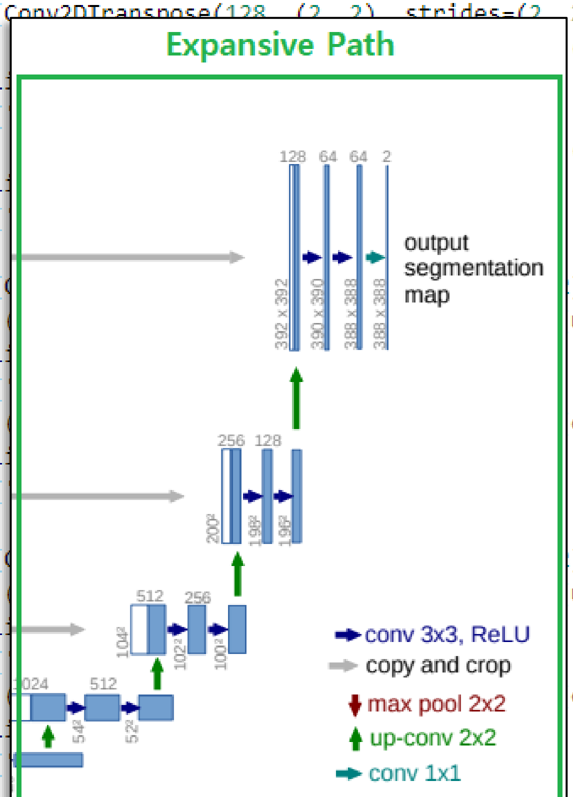
```
def get_unet(img_rows, img_cols):
    inputs = Input((img_rows, img_cols,1))
    conv1 = Conv2D(32, (3, 3), activation=None, padding='same')(inputs)
    conv1 = BatchNormalization()(conv1)
    conv1 = Activation('relu')(conv1)
    conv1 = Conv2D(32, (3, 3), activation=None, padding='same')(conv1)
    conv1 = BatchNormalization()(conv1)
    conv1 = Activation('relu')(conv1)
    pool1 = MaxPooling2D(pool_size=(2, 2))(conv1)

    conv2 = Conv2D(64, (3, 3), activation=None, padding='same')(pool1)
    conv2 = BatchNormalization()(conv2)
    conv2 = Activation('relu')(conv2)
    conv2 = Conv2D(64, (3, 3), activation=None, padding='same')(conv2)
    conv2 = BatchNormalization()(conv2)
    conv2 = Activation('relu')(conv2)
    pool2 = MaxPooling2D(pool_siz

    conv3 = Conv2D(128, (3, 3), a                                        l2)
    conv3 = BatchNormalization()(
    conv3 = Activation('relu')(co
    conv3 = Conv2D(128, (3, 3), a                                        v3)
    conv3 = BatchNormalization()(
    conv3 = Activation('relu')(co
    pool3 = MaxPooling2D(pool_siz

    conv4 = Conv2D(256, (3, 3), a                                        l3)
    conv4 = BatchNormalization()(
    conv4 = Activation('relu')(co
    conv4 = Conv2D(256, (3, 3), a                                        v4)
    conv4 = BatchNormalization()(
    conv4 = Activation('relu')(co
    pool4 = MaxPooling2D(pool_siz

    conv5 = Conv2D(512, (3, 3), a                                        l4)
    conv5 = BatchNormalization()(
    conv5 = Activation('relu')(co
    conv5 = Conv2D(512, (3, 3), a                                        v5)
    conv5 = BatchNormalization()(
    conv5 = Activation('relu')(conv5)
```



3.Baseline Code - Spine

2-1).U-net

```
up6 = concatenate([Conv2DTranspose(256, (2, 2), strides=(2, 2), padding='same')(conv5), conv4], axis=3
conv6 = Conv2D(256, (3, 3), activation=None, padding='same')(up6)
conv6 = BatchNormalization()(conv6)
conv6 = Activation('relu')(conv6)
conv6 = Conv2D(256, (3, 3), activation=None, padding='same')(conv6)
conv6 = BatchNormalization()(conv6)
conv6 = Activation('relu')(conv6)

up7 = concatenate([Conv2DTranspose(128, (2, 2), strides=(2, 2), padding='same')(conv6), conv3], axis=3
conv7 = Conv2D(128,                          (up7)
conv7 = BatchNormali
conv7 = Activation(
conv7 = Conv2D(128,                          (conv
conv7 = BatchNormali
conv7 = Activation(

up8 = concatenate([(                         ), pa
conv8 = Conv2D(64,                           up8)
conv8 = BatchNormali
conv8 = Activation(
conv8 = Conv2D(64,                           conv8)
conv8 = BatchNormali
conv8 = Activation(

up9 = concatenate([(                         ), padding='same')(conv8), conv1], axis=3)
conv9 = Conv2D(32,                           up9)
conv9 = BatchNormali
conv9 = Activation(
conv9 = Conv2D(32,                           conv9)
conv9 = BatchNormali
conv9 = Activation(

conv10 = Conv2D(1, (1, 1), activation='sigmoid')(conv9)

model = Model(inputs=inputs, outputs=conv10)

return model
```



- Concatenate : 입력된 레이어들을 하나의 layer로 합친 layer.
- Conv2DTranspose : Convolution layer에 전치행렬을 곱하여 upsampling 해주는 명령어

3.Baseline Code - Spine

3.Predict Validation

```
for i in range(len(true_list)):
    yt=true_list[i].flatten()
    yp=pred_list[i].flatten()
    mat=confusion_matrix(yt,yp)
    if len(mat) == 2:
        ac=(mat[1,1]+mat[0,0])/(mat[1,0]+mat[1,1]+mat[0,1]+mat[0,0])
        st=mat[1,1]/(mat[1,0]+mat[1,1])
        sp=mat[0,0]/(mat[0,1]+mat[0,0])
        if mat[1,0]+mat[1,1] == 0:
            specificity.append(sp)
            acc.append(ac)
        else:
            sensitivity.append(st)
            specificity.append(sp)
            acc.append(ac)
    else:
        specificity.append(1)
        acc.append(1)

    yt=true_list[i]
    yp=pred_list[i]
    if np.sum(yt) != 0 and np.sum(yp) != 0:
        dice = np.sum(yp[yt==1])*2.0 / (np.sum(yt) + np.sum(yp))
        dsc.append(dice)
    df=  df.append({'name':name_list[i], 'acc':ac, 'sen':st, 'spe':sp, 'dsc':dice}, ignore_index=True)
```

- Flatten : 데이터를 1차원으로 바꿔주는 레이어
- Confusion_matrix : 각 픽셀의 값을 비교해 TP/TN/FP/FN으로 나타내주는 행렬
    - ac = accuracy
    - st = sensitivity
    - sp = specificity
- Dice : 라벨링된 영역과 예측한 영역의 일치도를 보는 score

3.Baseline Code - Spine

5. Get score

- Segmentation된 마스크를 통해 뽑아낸 angle, distance 와 실제 측정된 값 사이의 오차를 비교

- Json에 저장된 실제 측정 값을 뽑아내는 코드
    - cobbsAngle 타입 데이터들 중 4-5번 척추 사이의 각도 데이터를 가져옴
    - 척추 사이의 거리를 젠 line 타입 데이터중에서도 4-5번 척추 사이의 거리 측정 자료를 가져옴

- 실측값과 결과값 사이의 비교는 오차값을 이용한 R2 Score를 사용

```python
def get_score(angle_list,dist_list,test_name):
    angle_test = []
    angle_ai = []
    dist_test = []
    dist_ai = []
    get_no45 = []
    for i in len(test_name):
        name = test_name[i]
        data = []

        jsonfile = test_path+'/{}.json'.format(name)
        for line in open(jsonfile,'r'):
            data.append(json.loads(line))
        for json_data in data:
            check = 0
            if json_data['annotation']['ANNOTATION_DATA'] is not None:
                for m in json_data['annotation']['ANNOTATION_DATA']:
                    if m['type']=='cobbAngle':
                        if m['label'] == 'L4-5A':
                            angle_test.append(m['angle'])
                            angle_ai.append(angle_list[i])
                            check = 1
                    elif m['type']=='line':
                        if m['label'] == 'L4-5H':
                            dist_test.append(m['distMm'])
                            dist_ai.append(dist_list[i])
                            check = 1
                if check==0:
                    get_no45.append(name)
    print(get_no45)
    print(r2_score(angle_ai, angle_test))
    print(r2_score(dist_ai, dist_test))
```

3.Baseline Code - pain

1.Load_data.ipynb

```
dcm_list = sorted(glob.glob('../data/DJDK300/*.dcm'))
json_list = sorted(glob.glob('../data/DJDK300/*.json'))

label = np.zeros((300,8,2))
reduce_ratio = np.zeros((300,2))
pixel_spacing = np.zeros((300,2))
file_name = []

for i in tqdm_notebook(range(len(json_list))):

    try:

        file_name.append(dcm_list[i].split('/')[-1].split('.')[0])

        image_dummy = sitk.ReadImage(dcm_list[i])

        x_spacing = image_dummy.GetSpacing()[0]     ①
        y_spacing = image_dummy.GetSpacing()[1]

        pixel_spacing[i,0] = x_spacing
        pixel_spacing[i,1] = y_spacing

        image_dummy = sitk.GetArrayFromImage(image_dummy).transpose(2,1,0)


        x_coor_reduce_ratio = 512 / image_dummy.shape[0]

        y_coor_reduce_ratio = 512 / image_dummy.shape[1]

        reduce_ratio[i,0] = x_coor_reduce_ratio
        reduce_ratio[i,1] = y_coor_reduce_ratio        ②
```

① Validation에 사용될 Distance Error 계산에 이용되는 Pixel spacing 값을 저장
- Pixel spacing : pixel과 pixel 사이의 실제 거리

② 이미지의 크기를 512x512로 변환 하였을 시, 각 좌표들의 위치 변화 비율을 저장.
- 라벨의 좌표 위치를 512x512 사이즈에 맞춰서 이동시키기 위함.

3.Baseline Code - pain

1.Load_data.ipynb

```
with open(json_list[i], "r") as josn_dummy:

    josn_dummy = json.load(josn_dummy)

point_1 = [josn_dummy['annotation']['ANNOTATION_DATA'][0]['vs']['x'], josn
point_2 = [josn_dummy['annotation']['ANNOTATION_DATA'][0]['ve']['x'], josn

point_3 = [josn_dummy['annotation']['ANNOTATION_DATA'][1]['vs']['x'], josn
point_4 = [josn_dummy['annotation']['ANNOTATION_DATA'][1]['ve']['x'], josn

point_5 = [josn_dummy['annotation']['ANNOTATION_DATA'][2]['vs']['x'], josn
point_6 = [josn_dummy['annotation']['ANNOTATION_DATA'][2]['ve']['x'], josn

point_7 = [josn_dummy['annotation']['ANNOTATION_DATA'][3]['vs']['x'], josn
point_8 = [josn_dummy['annotation']['ANNOTATION_DATA'][3]['ve']['x'], josn

label[i, 0, 0] = point_1[0] * x_coor_reduce_ratio
label[i, 0, 1] = point_1[1] * y_coor_reduce_ratio

label[i, 1, 0] = point_2[0] * x_coor_reduce_ratio
label[i, 1, 1] = point_2[1] * y_coor_reduce_ratio

label[i, 2, 0] = point_3[0] * x_coor_reduce_ratio
label[i, 2, 1] = point_3[1] * y_coor_reduce_ratio

label[i, 3, 0] = point_4[0] * x_coor_reduce_ratio
label[i, 3, 1] = point_4[1] * y_coor_reduce_ratio

label[i, 4, 0] = point_5[0] * x_coor_reduce_ratio
label[i, 4, 1] = point_5[1] * y_coor_reduce_ratio

label[i, 5, 0] = point_6[0] * x_coor_reduce_ratio
label[i, 5, 1] = point_6[1] * y_coor_reduce_ratio

label[i, 6, 0] = point_7[0] * x_coor_reduce_ratio
label[i, 6, 1] = point_7[1] * y_coor_reduce_ratio

label[i, 7, 0] = point_8[0] * x_coor_reduce_ratio
label[i, 7, 1] = point_8[1] * y_coor_reduce_ratio
```

```
# save numpy array
np.save('../data/data_set/label.npy', label)
np.save('../data/data_set/reduce_ratio.npy', reduce_ratio)
np.save('../data/data_set/pixel_spacing.npy', pixel_spacing)
np.save('../data/data_set/file_name.npy', file_name)
```

① Json 파일로부터 무릎 사이 거리를 측정할 때 기준이 되는 8개의 좌표점을 불러옴

② 512x512 크기로 변환되는 이미지에 맞춰서 불러온 기준점들의 좌표에 x,y 변화 비율을 곱해서 위치를 조정함.

③ 조정된 기준점들과 pixel spacing 값들을 npy 형식으로 저장함.

3.Baseline Code - pain

1.Load_data.ipynb

- Dicom 이미지를 불러와서 이미지의 크기를 조정하고 하나의 image array로 저장해주는 구문
  - tqdm : 반복문의 진행도를 출력해주는 함수
  - cv2.resize : 이미지의 사이즈를 선택된 크기로 변환
  - np.expand_dim : 이미지의 shape에 한 차원을 추가해주는 함수
    - Ex) (512,512) -> (512,512,1)

```
from tqdm import tqdm_notebook

dcm_list = sorted(glob.glob('../data/DJDK300/*.dcm'))
image_array = np.zeros((300,512,512,1), dtype=np.uint8)

for i in tqdm_notebook(range(len(dcm_list))):
    image_dummy = sitk.ReadImage(dcm_list[i])
    image_dummy = sitk.GetArrayFromImage(image_dummy).transpose(2,1,0)
    image_dummy = cv2.resize(image_dummy, dsize=(512,512))
    image_dummy = cv2.resize(image_dummy, dsize=(512,512))
    image_dummy = np.expand_dims(image_dummy, axis=2)
    image_array[i] = image_dummy
```

3.Baseline Code - pain

2.Train.ipynb

1) Load data

```
# Load Data
image_array = np.load('../data/data_set/image.npy')
label_array = np.load('../data/data_set/label.npy').reshape(-1, 16)


# Normalization
X_train = image_array[:180] / 255
y_train = label_array[:180] / 512

X_valid = image_array[180:240] / 255
y_valid = label_array[180:240] / 512

X_test = image_array[240:] / 255
y_test = label_array[240:] / 512
```

- 학습에 사용할 data 로드
- 총 300케이스의 데이터에서 Train:Validation:Test = 180:60:60 = 6:2:2 비율로 분리
- 각 데이터에서 image는 8bit(0~255)값을 0~1 값으로 변환
- Label은 좌표값이므로 (0~512,0~512)의 값을 가지고, 이를 512로 나누어 0~1 값으로 변환.

3.Baseline Code - pain

2.Train.ipynb

2-0)model

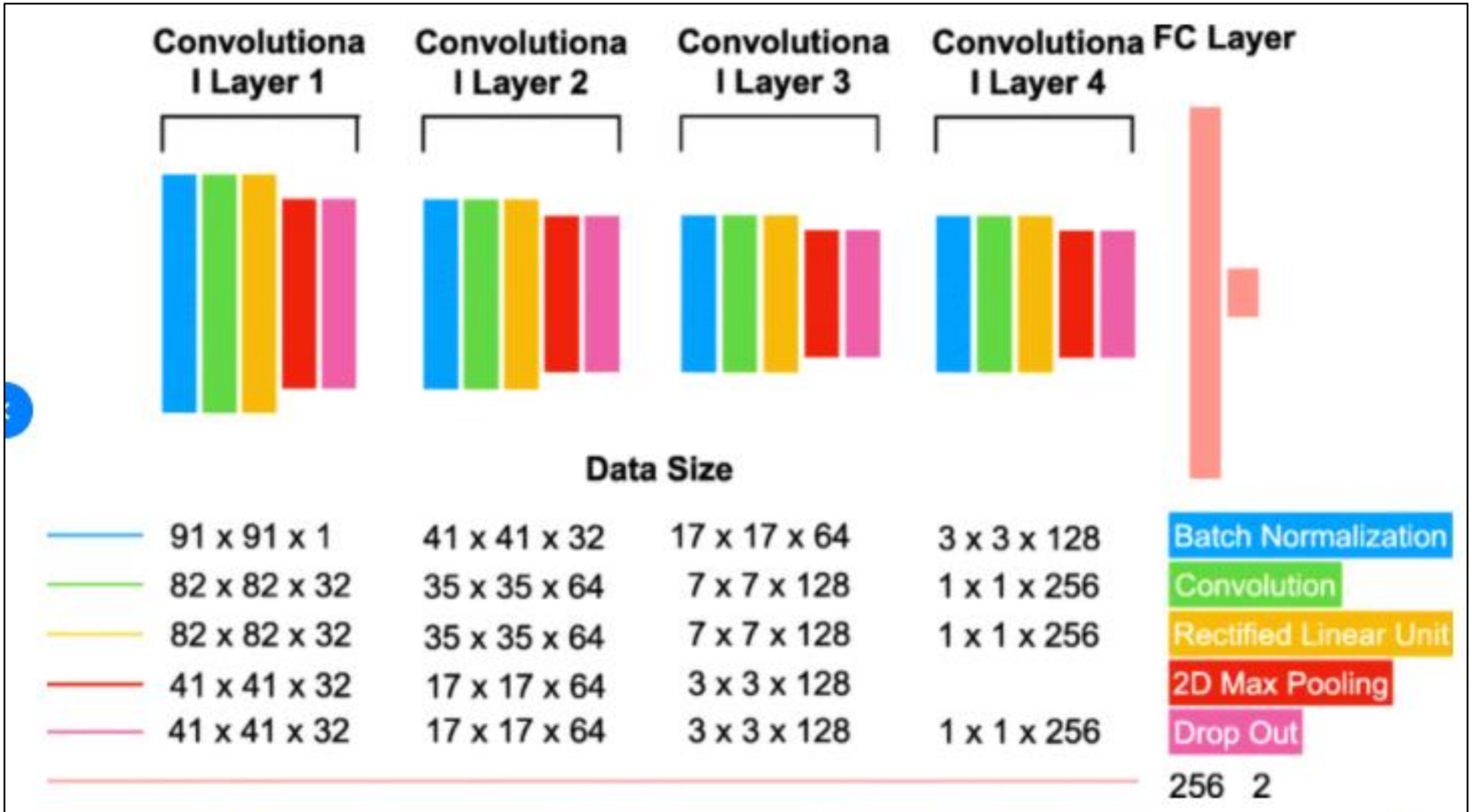
```
# Model
model = landmark_cnn()
model.compile( loss=tf.keras.losses.mean_squared_error , optimizer=tf.ke ras.optimizers.Adam( lr=0.0001 ) , metrics=[ 'mse' ] )


# Call Back
monitor = 'val_loss'

reduce_lr = ReduceLROnPlateau(monitor=monitor, factor=0.1, patience=10, min_lr=0.0000001,verbose=1)
earlystopper = EarlyStopping(monitor=monitor, patience=50, verbose=1)
model_checkpoint = ModelCheckpoint(filepath = '../result/model_save/landmark_model_1.h5', verbose=1, save_best_only=True)

callbacks_list = [reduce_lr, model_checkpoint, earlystopper]


# Train
history = model.fit(X_train, y_train, batch_size=20, epochs=1000, shuffle=True, verbose=1,
                    validation_data=(X_test, y_test), callbacks=callbacks_list)
```

- Model : Landmark cnn
- Loss : mean_squared_error(평균 제곱 편차). 예측값의 오차 제곱값의 평균
- Optimizer : Adam optimizer
- Callback list
  - Reduce_lr : 학습 가중치를 수정할때 loss값이 발산하는것을 막기 위하여 학습률(learning rate)를 일정 조건에 따라 점차적으로 감소시키는 함수
  - Earlystopper : 학습의 과적합을 막기 위하여 일정 횟수 이상 validation loss의 값이 변화가 없으면 학습을 중단시키는 함수
  - Checkpoint : 한 epoch이 끝날때마다 모델 학습의 중간 결과를 저장하는 함수
- Model attribute
  - Batch size : 20, epochs = 1000

3.Baseline Code - pain

2.Train.ipynb

2-1) Landmark Detection Architecture



The model architecture of our landmark detecting framework. The architecture has 4 Convolutional Cluster (CC) and 2 Fully Connected (FC) layers. Each CC contains the Batch Normalization layer, Convolution layer, Non-linearity, 2D max-pooling, and dropout in the mentioned order

3.Baseline Code - pain

2.Train.ipynb

2-2) Landmark Detection model

- 이미지에서 중요한 지점의 좌표를 찾아내는 landmark Detection 알고리즘
- 각 Step마다 컨볼루션 블록을 이용
  1. (512,512,1) 이미지에 3x3 커널 사이즈의 컨볼루션 레이어 적용
  2. 각 feature들의 평균과 표준편차를 이용하여 feature 값들을 normalize 해준다
  3. Output 값에 ReLU 활성함수를 적용한다.
- Step 사이에 2x2 Max pooling을 적용하여 Feature map의 크기를 반으로 축소
- 최종적으로 20%의 비율로 drop out을 진행하여 오버피팅을 방지해주고, 만들어진 feature map을 1차원으로 변환한 후 output size에 맞추어 결과값을 출력

```python
# Settings
INPUT_SHAPE = (512, 512, 1)
OUTPUT_SIZE = 16
```

```python
def landmark_cnn(input_shape=INPUT_SHA

    img_input = Input(shape=input_shap

    x = Conv2D(16, (3,3), strides=(1,1), name='Conv1')(img_input)
    x = BatchNormalization()(x)
    x = Activation('relu', name='Relu_conv1')(x)

    x = MaxPooling2D(pool_size=(2,2), strides=(2,2), name='Pool1')(x)


    x = Conv2D(32, (3,3), strides=(1,1), name='Conv2')(x)
    x = BatchNormalization()(x)
    x = Activation('relu', name='Relu_conv2')(x)
    x = Conv2D(32, (3,3), strides=(1,1), name='Conv3')(x)
    x = BatchNormalization()(x)
    x = Activation('relu', name='Relu_conv3')(x)

    x = MaxPooling2D(pool_size=(2,2), strides=(2,2), name='Pool2')(x)


    x = Conv2D(32, (3,3), strides=(1,1), name='Conv4')(x)
    x = BatchNormalization()(x)
    x = Activation('relu', name='Relu_conv4')(x)
    x = Conv2D(32, (3,3), strides=(1,1), name='Conv5')(x)
    x = BatchNormalization()(x)
    x = Activation('relu', name='Relu_conv5')(x)

    x = MaxPooling2D(pool_size=(2,2), strides=(2,2), name='Pool3')(x)


    x = Conv2D(64, (3,3), strides=(1,1), name='Conv6')(x)
    x = BatchNormalization()(x)
    x = Activation('relu', name='Relu_conv6')(x)

    x = Dropout(0.2)(x)

    x = Flatten(name='Flatten')(x)
    x = Dense(128, activation='relu', name='FC1')(x)
    x = Dense(output_size, activation=None, name='Predictions')(x)


    model = Model([img_input], x, name='Landmark_model')

    return model
```

3.Baseline Code - pain

3.Evaluation.ipynb

1) Load data

```
original_size_label = np.zeros((60, 8, 2))


predict = np.load('../result/predict/predict.npy')
label = np.load('../data/data_set/label.npy')[240:]

file_name = np.load('../data/data_set/file_name.npy')[240:]
reduce_ratio = np.load('../data/data_set/reduce_ratio.npy')[240:]
pixel_spacing = np.load('../data/data_set/pixel_spacing.npy')[240:]


original_size_predict = predict * 512
original_size_predict = original_size_predict.reshape(60, 8, 2)


for i in tqdm_notebook(range(predict.shape[0])):

    x_reduce_ratio = reduce_ratio[i][0]
    y_reduce_ratio = reduce_ratio[i][1]

    original_size_label[i][:, 0] = label[i][:, 0] / x_reduce_ratio
    original_size_label[i][:, 1] = label[i][:, 1] / y_reduce_ratio

    original_size_predict[i][:,0] = original_size_predict[i][:, 0] / x_reduce_ratio
    original_size_predict[i][:,1] = original_size_predict[i][:, 1] / y_reduce_ratio
```

①

- Original_size_label : 예측된 결과값을 512 x512 사이즈에 맞춰서 normalize 해주고, 원본 라벨의 shape인 (n,8,2) 형식으로 저장한 array

① 각 데이터의 label과 predict 좌표값에 data loading중에 저장한 X,Y 변화량을 곱해 원본 라벨의 x,y 좌표값을 구함.

3.Baseline Code - pain

3.Evaluation.ipynb

2) distance error

```
point_1_distance_error.append(pixel_spacing[i][0] * np.sqrt((original_size_label[i][:, 0][0] - original_size_predict[i]
                                                    [:, 0][0])**2 + (original_size_label[i][:, 1][0] - original_size_predict[
point_2_distance_error.append(pixel_spacing[i][0] * np.sqrt((original_size_label[i][:, 0][1] - original_size_predict[i]
                                                    [:, 0][1])**2 + (original_size_label[i][:, 1][1] - original_size_predict[
point_3_distance_error.append(pixel_spacing[i][0] * np.sqrt((original_size_label[i][:, 0][2] - original_size_predict[i]
                                                    [:, 0][2])**2 + (original_size_label[i][:, 1][2] - original_size_predict[
point_4_distance_error.append(pixel_spacing[i][0] * np.sqrt((original_size_label[i][:, 0][3] - original_size_predict[i]
                                                    [:, 0][3])**2 + (original_size_label[i][:, 1][3] - original_size_predict[

point_5_distance_error.append(pixel_spacing[i][0] * np.sqrt((original_size_label[i][:, 0][4] - original_size_predict[i]
                                                    [:, 0][4])**2 + (original_size_label[i][:, 1][4] - original_size_predict[
point_6_distance_error.append(pixel_spacing[i][0] * np.sqrt((original_size_label[i][:, 0][5] - original_size_predict[i]
                                                    [:, 0][5])**2 + (original_size_label[i][:, 1][5] - original_size_predict[
point_7_distance_error.append(pixel_spacing[i][0] * np.sqrt((original_size_label[i][:, 0][6] - original_size_predict[i]
                                                    [:, 0][6])**2 + (original_size_label[i][:, 1][6] - original_size_predict[
point_8_distance_error.append(pixel_spacing[i][0] * np.sqrt((original_size_label[i][:, 0][7] - original_size_predict[i]
                                                    [:, 0][7])**2 + (original_size_label[i][:, 1][7] - original_size_predict[
```

- **Distance Error** : 예측 좌표값과 실측 좌표값 사이의 거리차이를 비교하여 모델의 detection 성능을 검증

- $\sqrt{((\text{실측 } X\text{좌표} - \text{예측 } X\text{좌표})^2 + (\text{실측 } Y\text{좌표} - \text{예측 } Y\text{좌표})^2)}$ * pixel spacing 값

  = (실측 좌표와 예측 좌표 사이의 거리)*pixel spacing 값이다.

3.Baseline Code - pain

3.Evaluation.ipynb

3) mean/standard deviation

```
point_1_mean, point_1_std = np.round(
    np.mean(point_1_distance_error), 3), np.round(np.std(point_1_distance_error), 3)
point_2_mean, point_2_std = np.round(
    np.mean(point_2_distance_error), 3), np.round(np.std(point_2_distance_error), 3)
point_3_mean, point_3_std = np.round(
    np.mean(point_3_distance_error), 3), np.round(np.std(point_3_distance_error), 3)
point_4_mean, point_4_std = np.round(
    np.mean(point_4_distance_error), 3), np.round(np.std(point_4_distance_error), 3)


point_5_mean, point_5_std = np.round(
    np.mean(point_5_distance_error), 3), np.round(np.std(point_5_distance_error), 3)
point_6_mean, point_6_std = np.round(
    np.mean(point_6_distance_error), 3), np.round(np.std(point_6_distance_error), 3)
point_7_mean, point_7_std = np.round(
    np.mean(point_7_distance_error), 3), np.round(np.std(point_7_distance_error), 3)
point_8_mean, point_8_std = np.round(
    np.mean(point_8_distance_error), 3), np.round(np.std(point_8_distance_error), 3)
```

- 예측된 landmark 좌표값들의 평균과 표준편차 계산
  - np.mean : array의 평균
  - np.std : array의 표준편차

```
HBox(children=(IntProgress(value=0, max=60), HTML(value='')))


Point 1 Distance Error:  55.062  ±  61.879     (mm)
Point 2 Distance Error:  55.414  ±  61.74      (mm)
Point 3 Distance Error:  95.49  ±  66.309      (mm)
Point 4 Distance Error:  96.096  ±  66.147     (mm)
Point 5 Distance Error:  98.915  ±  39.359     (mm)
Point 6 Distance Error:  97.287  ±  42.316     (mm)
Point 7 Distance Error:  92.259  ±  54.752     (mm)
Point 8 Distance Error:  93.966  ±  52.573     (mm)
```

3.Baseline Code - pain

3.Evaluation.ipynb

4) Line length

```
label_line_1.append(np.round(pixel_spacing[i][0] * np.sqrt((original_size_label[i][0][0] - original_size_label[i][1][0])**2 +
                                                           (original_size_label[i][0][1] - original_size_label[i][1][1])**2), 3))
label_line_2.append(np.round(pixel_spacing[i][0] * np.sqrt((original_size_label[i][2][0] - original_size_label[i][3][0])**2 +
                                                           (original_size_label[i][2][1] - original_size_label[i][3][1])**2), 3))
label_line_3.append(np.round(pixel_spacing[i][0] * np.sqrt((original_size_label[i][4][0] - original_size_label[i][5][0])**2 +
                                                           (original_size_label[i][4][1] - original_size_label[i][5][1])**2), 3))
label_line_4.append(np.round(pixel_spacing[i][0] * np.sqrt((original_size_label[i][6][0] - original_size_label[i][7][0])**2 +
                                                           (original_size_label[i][6][1] - original_size_label[i][7][1])**2), 3))

pred_line_1.append(np.round(pixel_spacing[i][0] * np.sqrt((original_size_predict[i][0][0] - original_size_predict[i][1][0])**2 +
                                                          (original_size_predict[i][0][1] - original_size_predict[i][1][1])**2), 3))
pred_line_2.append(np.round(pixel_spacing[i][0] * np.sqrt((original_size_predict[i][2][0] - original_size_predict[i][3][0])**2 +
                                                          (original_size_predict[i][2][1] - original_size_predict[i][3][1])**2), 3))
pred_line_3.append(np.round(pixel_spacing[i][0] * np.sqrt((original_size_predict[i][4][0] - original_size_predict[i][5][0])**2 +
                                                          (original_size_predict[i][4][1] - original_size_predict[i][5][1])**2), 3))
pred_line_4.append(np.round(pixel_spacing[i][0] * np.sqrt((original_size_predict[i][6][0] - original_size_predict[i][7][0])**2 +
                                                          (original_size_predict[i][6][1] - original_size_predict[i][7][1])**2), 3))
```

- 실측 label인 8개의 좌표를 이용해 실제 무릎사이의 거리를 측정
- Landmark Detection으로 찾은 8개의 landmark point를 이용하여 무릎 사이 거리를 예측
- $\sqrt{((A\text{점 } X\text{좌표} - B\text{점 } X\text{좌표})^2 + (A\text{점 } Y\text{좌표} - B\text{점 } Y\text{좌표})^2)}$ * pixel spacing 값

3.Baseline Code - pain

3.Evaluation.ipynb

5) Line difference mean/standard deviation

```
print('Line 1 R2 Score: ',r2_score(label_line_1, pred_line_1))
print('Line 2 R2 Score: ',r2_score(label_line_2, pred_line_2))
print('Line 3 R2 Score: ',r2_score(label_line_3, pred_line_3))
print('Line 4 R2 Score: ',r2_score(label_line_4, pred_line_4))
```

- 실측 거리값(label_line)과 예측 거리값(pred_line) 사이의 오차를 이용해서 R2-score를 계산
  - R2-score : 실측값의 변동량 대비 모델 예측값의 변동량을 나타내는 값. 결정계수라고 한다.
  - R2-score는 실측값과 예측값 사이의 상관관계가 높을수록 1에 가까워짐.

4.성능평가지표

- **모델 개발 평가지표**

데이터 : 척추 X-ray 영상

- ✓ 척추 L4번과 L5번 사이의 간격(디스크 높이)에 대해 GT(Ground Truth)와 개발된 알고리즘으로 측정된 결과 간에 R2 score
- ✓ 척추 L4번과 L5번 사이의 각도에 대해 GT와 개발된 알고리즘으로 측정된 결과간에 R2 score

데이터 : 무릎 X-ray 영상

- ✓ 좌우 각각 내측 관절 간격과 외측 관절 간격에 대해 GT와 개발된 알고리즘으로 측정된 결과 간에 R2 score