딥러닝을 활용한 캡슐내시경 판별모델

CODE STATES AI Bootcamp 3기 신정인

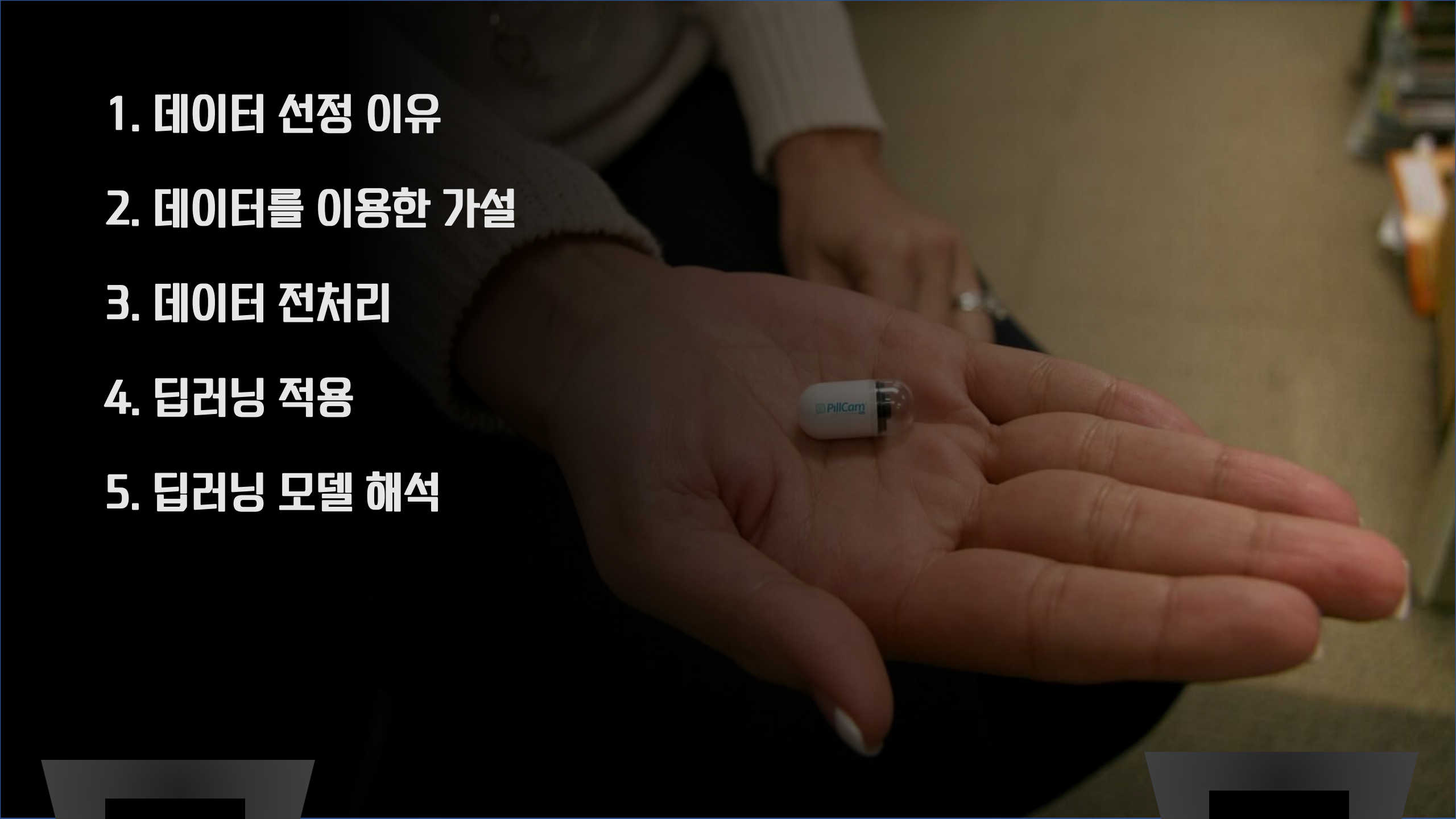
1. 데이터 선정 이유

2. 데이터를 이용한 가설

3. 데이터 전처리

4. 딥러닝 적용

5. 딥러닝 모델 해석



1
데이터 선정 이유

The Kvasir Datasets

https://datasets.simula.no/kvasir/

4000 Images 8 Classes 500 Images for each class (v1)
8000 Images 8 Classes 1000 Images for each class (v2)

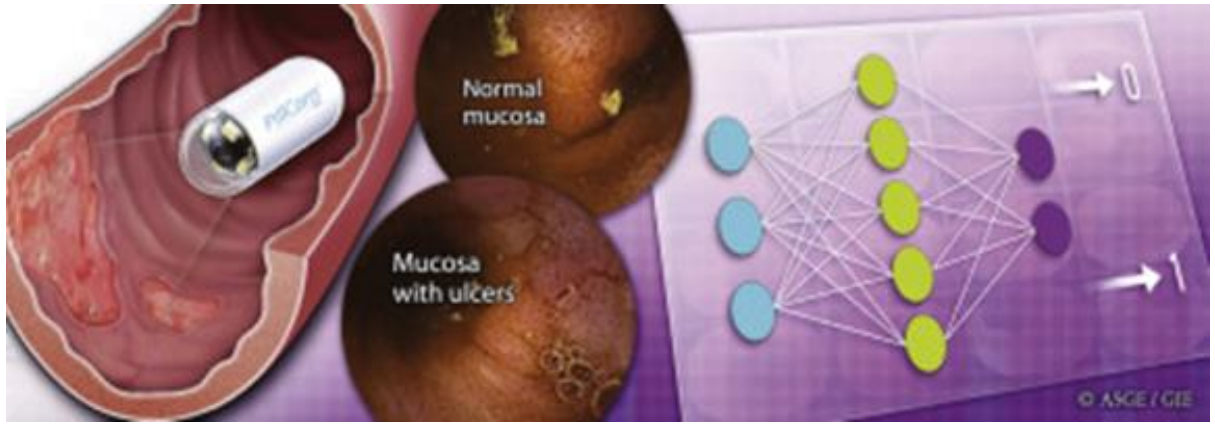
Class	해석	모델 적용
Z-line	식도 위 경계	○
Normal-Pylorus	위 소장 경계	○
Normal-Cecum	정상 맹장	○
Esophagitis	식도염	○
Polyps	폴립(용종)	○
Ulcerative Colitis	궤양성 대장염	○
Dyed and Lifted Polyps	염색 및 제거된 폴립	×
Dyed and Resection Margins	염색된 폴립 절제면	×

※ 검진의 목적이기 때문에 절제시술 후 사진은 제외 → 3,000 Images 학습

Why Capsule Endoscopy?

- 고통 없이 일상 생활을 하면서 간편하게 진행할 수 있다.
- 소장 등 기존 내시경으로 진단이 어려운 부분도 검사가 가능하다.
- 약 10시간, 5만여장의 사진을 촬영하고, 소화기내과 의사가 판독해야 한다.
- 몇 가지 한계와 높은 검사비용으로 널리 보급되지 못하고 있다.

Why Capsule Endoscopy + Deep Learning ?

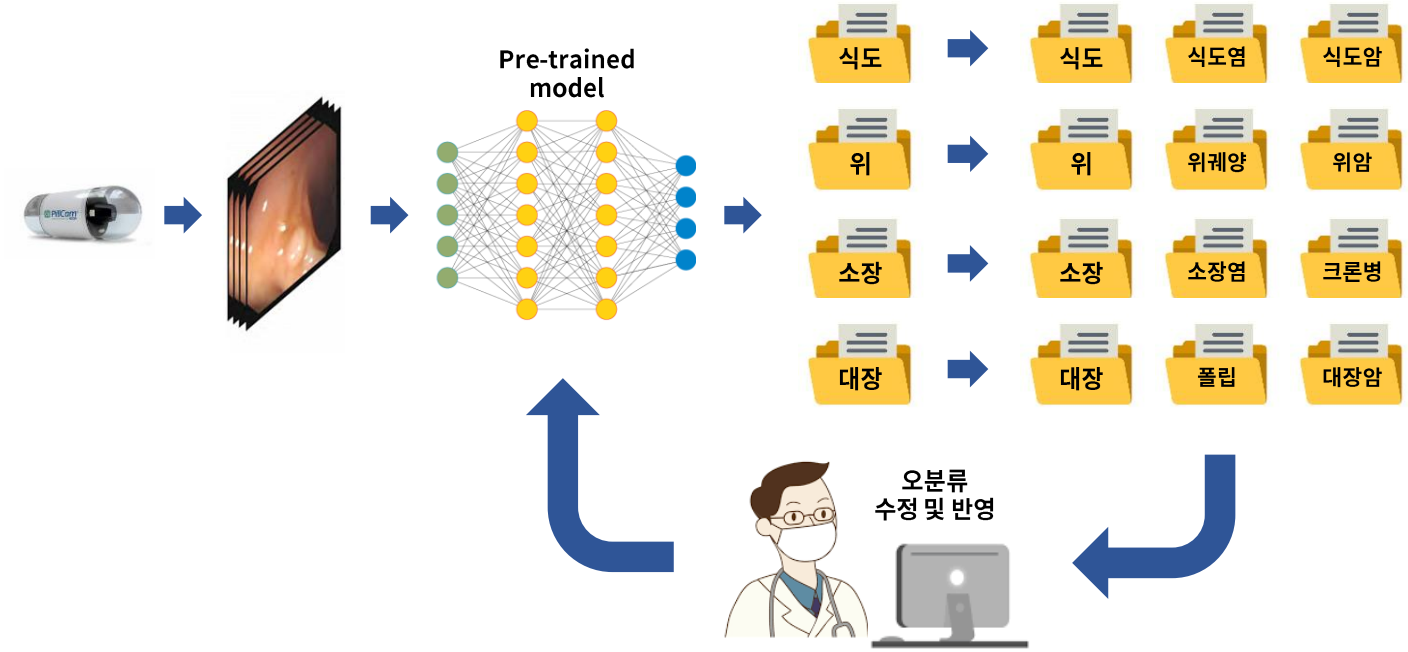


- 캡슐 내시경 검사시 매번 5만여장의 이미지 데이터가 수집된다.
- 빠른 사진 분류로 의료진의 업무 효율을 높일 수 있다.
- 소화기내과 의사는 주 71.5시간 업무, 68.9%가 우울과 불안을 경험
- 기존 내시경도 숙련도에 따라 용종의 미탐지율이 20% 정도 될 수 있다.
- 모델이 내장되면 실시간으로 사진 정보를 분석할 수 있다.
- 메모리, 카메라 등 하드웨어의 발전으로 성능은 높아지고 가격은 하락했다.

2
데이터를 이용한 가설

가설1.

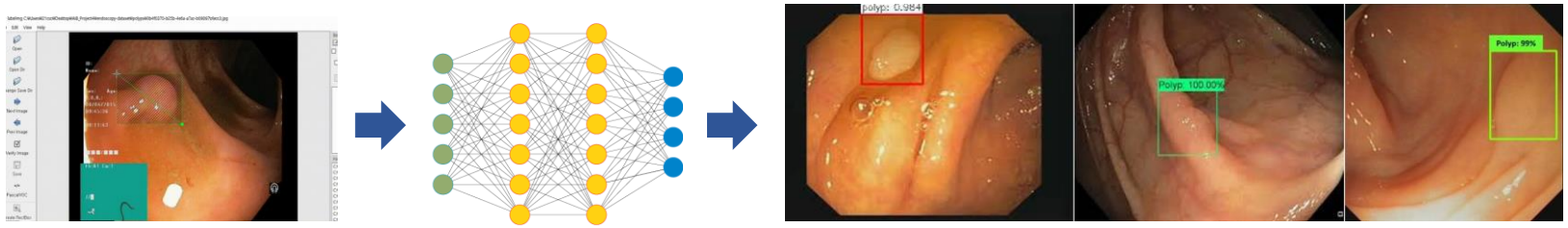
딥러닝 모델이 내시경 사진 분류를 할 수 있을 것이다.



- Image Classification 을 통해 사진이 어떤 부위인지, 어떤 상태인지 판별할 수 있을 것이다.

가설2.

딥러닝 모델이 Polyp Object Detection 을 할 수 있을 것이다.



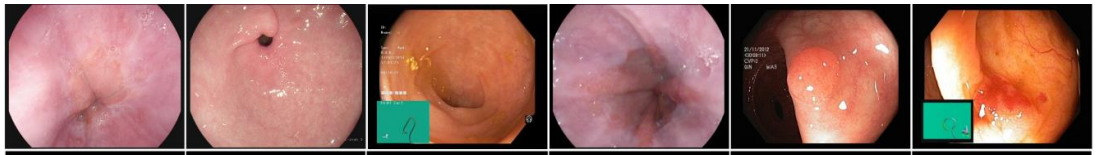
- Polyp Object Detection 을 통해 폴립의 수, 크기 등을 파악할 수 있다.
- 차후 폴립의 5가지 종류에 따라 분류하는 Object Detection model도 만들 수 있을 것이다. (선종성폴립/과형성폴립/톱니폴립/염증성폴립/과오종)

※ 명확히 분리된 객체가 아니어서 픽셀 단위로 구별하는 Segmentation이 아닌 Bounding-box를 활용하는 Object Detection을 선택

3

데이터 전처리

Preprocessing - Image Classification



Normal-Z-line	Normal-pylorus	Normal-cecum	Esophagitis	Polyps	Ulcerative-colitis
정상 식도	정상 십이지장	정상 맹장	식도염	폴립(용종)	궤양성 대장염

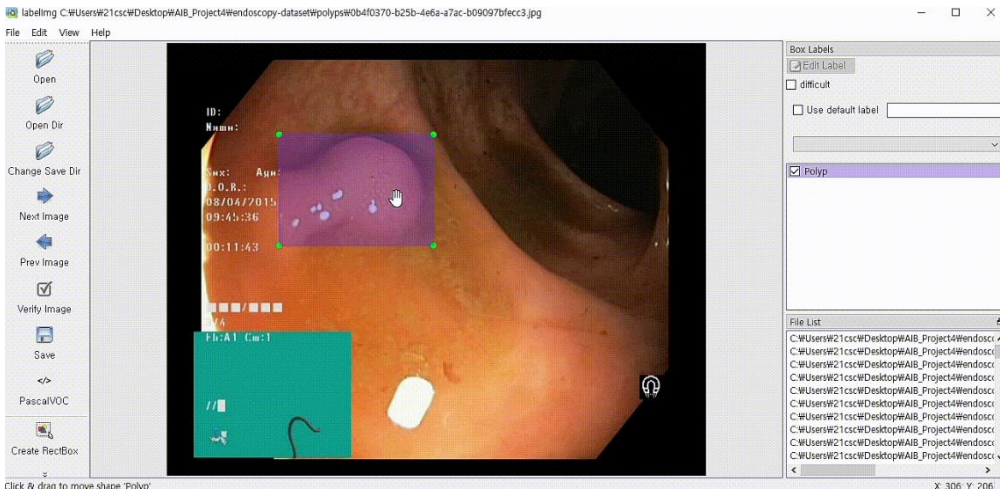
1. 좌측 하단 녹색 창

이미지 좌측 하단 패턴 일정하게 만들까 생각을 했지만,
향후 테스트에 나쁜 영향을 줄 것 같아 그대로 학습시켰다.

2. Image Augmentation

내시경 이미지는 방향성이 없어 회전시켜 증폭하면 좋다고 판단되었으나,
제한된 시간 때문에 차후에 진행하기로 결정하였다.

Preprocessing - Object Detection



Polyp Object Labeling

Polyp에 대한 Object Detection을 위해 LabelIMG를 활용,
500장의 Polyp 이미지에 대해 레이블링을 실시하였다.

향후 도메인 지식을 더 쌓고, 5가지의 Polyp으로 상세하게 분류 할 계획이다.

4
딥러닝 적용

1. Image Classification

Normal-Z-line	Normal-pylorus	Normal-cecum	Esophagitis	Polyps	Ulcerative-colitis
정상 식도	정상 십이지장	정상 맹장	식도염	폴립(용종)	궤양성 대장염

```
# 6개 클래스, 은닉층은 relu, 출력층은 softmax

model = Sequential()

model.add(Conv2D(32, input_shape=(128,128,3), kernel_size=(3,3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2,2)))
model.add(Conv2D(128, (3,3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2,2)))
model.add(Flatten())
model.add(Dense(64, activation='relu'))
model.add(Dense(6, activation='softmax'))

model.summary()
```

```
model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
```

1. Image Classification

```
model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
```

```
model.fit(train_gen, epochs=10, validation_data=val_gen)
```

```
Epoch 1/10
75/75 [==============================] - 898s 12s/step - loss: 1.1461 - accuracy: 0.5567 - val_loss: 0.8305 - val_accuracy: 0.6367
Epoch 2/10
75/75 [==============================] - 129s 2s/step - loss: 0.6396 - accuracy: 0.7308 - val_loss: 0.6931 - val_accuracy: 0.7200
Epoch 3/10
75/75 [==============================] - 128s 2s/step - loss: 0.4982 - accuracy: 0.8029 - val_loss: 0.6801 - val_accuracy: 0.7517
Epoch 4/10
75/75 [==============================] - 131s 2s/step - loss: 0.4695 - accuracy: 0.8050 - val_loss: 0.6772 - val_accuracy: 0.7550
Epoch 5/10
75/75 [==============================] - 128s 2s/step - loss: 0.3256 - accuracy: 0.8817 - val_loss: 0.7243 - val_accuracy: 0.7617
Epoch 6/10
75/75 [==============================] - 128s 2s/step - loss: 0.2571 - accuracy: 0.8988 - val_loss: 0.7773 - val_accuracy: 0.7533
Epoch 7/10
75/75 [==============================] - 128s 2s/step - loss: 0.1846 - accuracy: 0.9396 - val_loss: 0.7914 - val_accuracy: 0.7567
Epoch 8/10
75/75 [==============================] - 131s 2s/step - loss: 0.1591 - accuracy: 0.9417 - val_loss: 0.7916 - val_accuracy: 0.7583
Epoch 9/10
75/75 [==============================] - 130s 2s/step - loss: 0.1296 - accuracy: 0.9513 - val_loss: 0.8828 - val_accuracy: 0.7383
Epoch 10/10
75/75 [==============================] - 130s 2s/step - loss: 0.0905 - accuracy: 0.9688 - val_loss: 0.9110 - val_accuracy: 0.7733
<keras.callbacks.History at 0x7f5588b3bdd0>
```

- Chance Level : 0.1667 (6 classes)
- Train Accuracy : 0.9688
- Validation Accuracy : 0.7733

2. Object Detection


```
%cd /content
!git clone https://github.com/ultralytics/yolov5.git
```

```
/content
Cloning into 'yolov5'...
remote: Enumerating objects: 9185, done.
remote: Counting objects: 100% (1/1), done.
remote: Total 9185 (delta 0), reused 0 (delta 0), pack-reused 9184
Receiving objects: 100% (9185/9185), 9.58 MiB | 25.69 MiB/s, done.
Resolving deltas: 100% (6385/6385), done.
```

```
%cd /content/yolov5/
!pip install -r requirements.txt
```

```
%cat /content/drive/MyDrive/1.codestates/3.project/data.yaml
```

데이터 경로 및 클래스 정보가 담겨 있는 yaml

```
names:
- Polyp
nc: 1
train: /content/drive/MyDrive/1.codestates/3.project/train.txt
val: /content/drive/MyDrive/1.codestates/3.project/val.txt
```

2. Object Detection

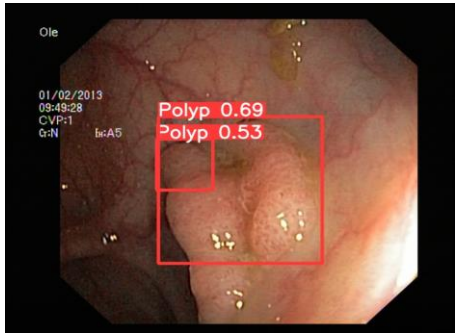
```
%cd /content/yolov5/

!python train.py --img 256 --batch 32 --epochs 30 --data /content/drive/MyDrive/1.codestates/3.project/data.yaml
--cfg ./models/yolov5s.yaml --weights yolov5s.pt --name polyp_yolov5s_results
```

```
/content/yolov5
train: weights=yolov5s.pt, cfg=./models/yolov5s.yaml, data=/content/drive/MyDrive/1.codestates/3.project/data.yaml,
github: up to date with https://github.com/ultralytics/yolov5 ✅
YOLOv5 🚀 v5.0-405-gfad57c2 torch 1.9.0+cu102 CPU

hyperparameters: lr0=0.01, lrf=0.2, momentum=0.937, weight_decay=0.0005, warmup_epochs=3.0, warmup_momentum=0.8, w
Weights & Biases: run 'pip install wandb' to automatically track and visualize YOLOv5 🚀 runs (RECOMMENDED)
TensorBoard: Start with 'tensorboard --logdir runs/train', view at http://localhost:6006/
Overriding model.yaml nc=80 with nc=1
```

Input-size 256 , Batch-size 32 , Epoch 30 , weight yolov5s

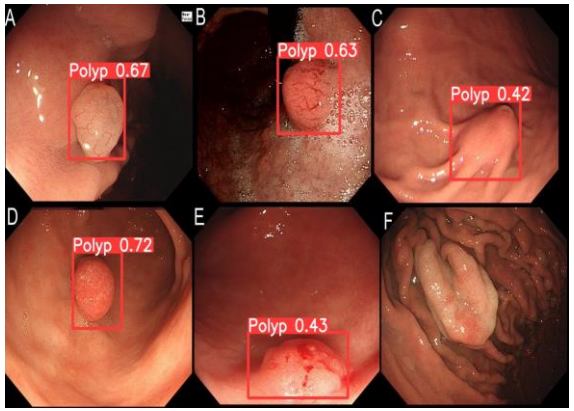
```
               all        100        123      0.838      0.796      0.879      0.603

Epoch   gpu_mem       box       obj       cls    labels  img_size
28/29        0G   0.03682   0.01454         0        33       256: 100% 13/13 [02:05<00:00,  9.67s/it]
           Class     Images     Labels          P          R     mAP@.5 mAP@.5:.95: 100% 2/2 [00:09<00:00,  4.50s/it]
             all        100        123      0.889       0.78      0.887      0.579

Epoch   gpu_mem       box       obj       cls    labels  img_size
29/29        0G   0.03757   0.01601         0        55       256: 100% 13/13 [02:04<00:00,  9.59s/it]
           Class     Images     Labels          P          R     mAP@.5 mAP@.5:.95: 100% 2/2 [00:10<00:00,  5.38s/it]
             all        100        123      0.912      0.764      0.894      0.573

30 epochs completed in 1.129 hours.
Optimizer stripped from runs/train/polyp_yolov5s_results2/weights/last.pt, 14.3MB
Optimizer stripped from runs/train/polyp_yolov5s_results2/weights/best.pt, 14.3MB
Results saved to runs/train/polyp_yolov5s_results2
```

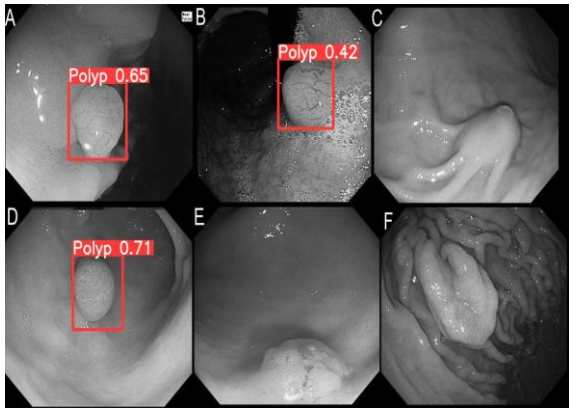
2. Object Detection



여러 개의 Polyp을 잘 찾아내지만

Accuracy 성능이 좋다고 보긴 힘들 것 같다.



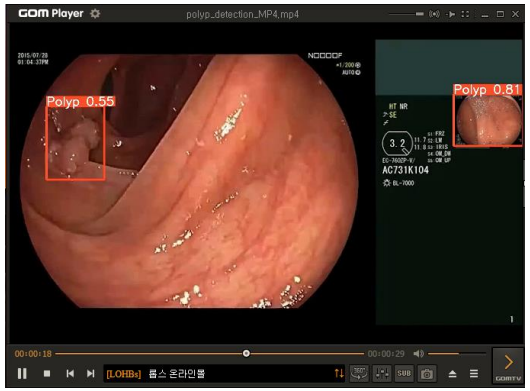


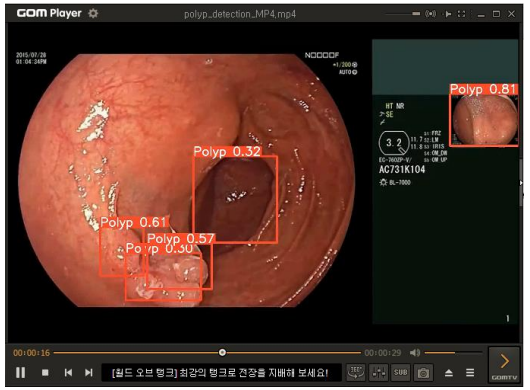
흑백 이미지로 변경할 경우 Detection 성능이 떨어지는 것을 확인하였다.

2. Object Detection

```
# 동영상에서 Object Detection 하기

!python detect.py --source /content/drive/MyDrive/1.codestates/3.project/polyp-MP4.mp4
--weights /content/yolov5/runs/train/polyp_yolov5s_results/weights/best.pt
```

```
detect: weights=['/content/yolov5/runs/train/polyp_yolov5s_results2/weights/best.pt'], source=/content/drive/MyDri
YOLOv5 🚀 v5.0-405-gfad57c2 torch 1.9.0+cu102 CPU

Fusing layers...
Model Summary: 224 layers, 7053910 parameters, 0 gradients, 16.3 GFLOPs
video 1/1 (1/888) /content/drive/MyDrive/1.codestates/3.project/polyp-MP4.mp4: 384x640 5 Polyps, Done. (0.314s)
video 1/1 (2/888) /content/drive/MyDrive/1.codestates/3.project/polyp-MP4.mp4: 384x640 5 Polyps, Done. (0.316s)
video 1/1 (3/888) /content/drive/MyDrive/1.codestates/3.project/polyp-MP4.mp4: 384x640 5 Polyps, Done. (0.293s)
video 1/1 (4/888) /content/drive/MyDrive/1.codestates/3.project/polyp-MP4.mp4: 384x640 5 Polyps, Done. (0.285s)
video 1/1 (5/888) /content/drive/MyDrive/1.codestates/3.project/polyp-MP4.mp4: 384x640 5 Polyps, Done. (0.285s)
video 1/1 (6/888) /content/drive/MyDrive/1.codestates/3.project/polyp-MP4.mp4: 384x640 5 Polyps, Done. (0.292s)
```





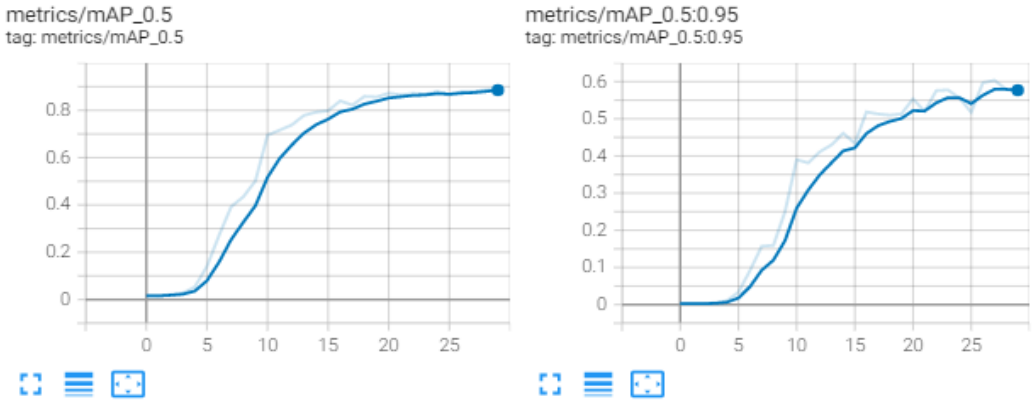
동영상에서의 Object Detection도 가능한 것을 확인하였다.

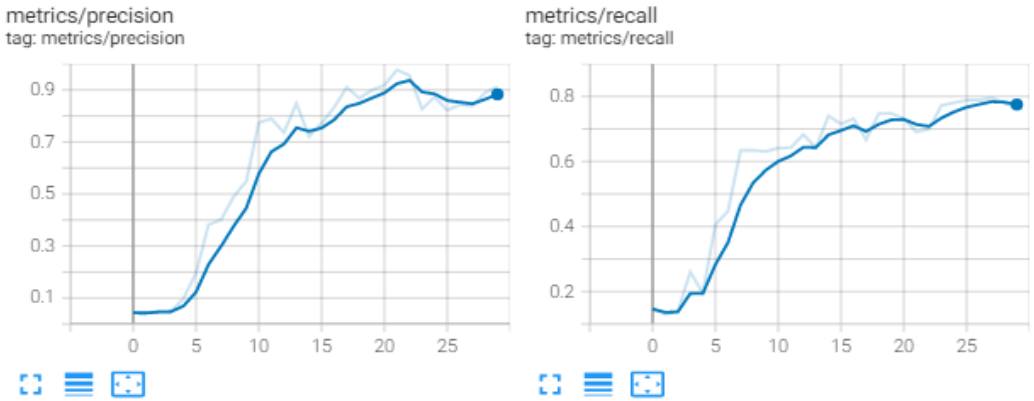
(모든 프레임을 모델에 적용하고 새로운 동영상을 생성하게 된다)

5

딥러닝 모델 해석

Tensorboard (30 Epoch)



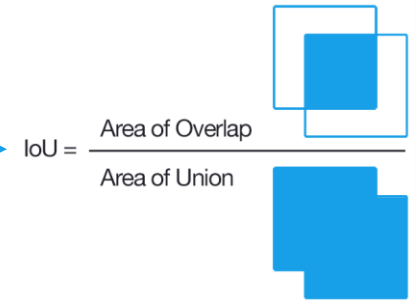


Precision : 0.912 **Recall : 0.764**

mAP_0.5 : 0.894 **mAP_0.5:0.95 : 0.573**

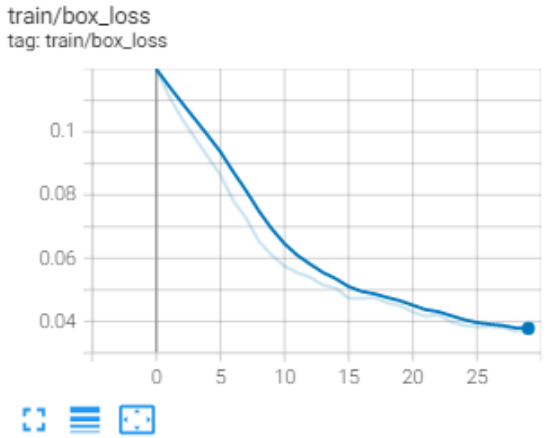
Precision = 올바르게 탐지한 수 / 탐지한 수
Recall = 올바르게 탐지한 수 / 실제 오브젝트 수

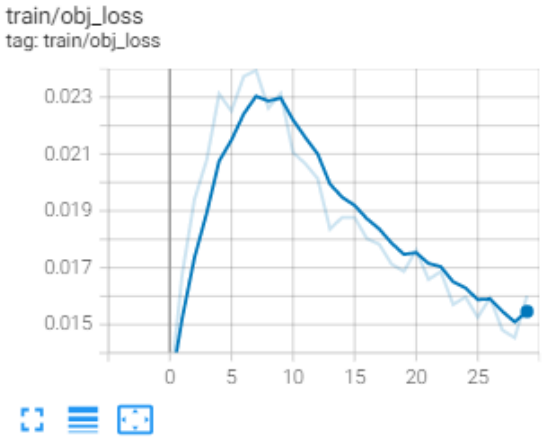
※ mAP_0.5 : IoU가 0.5이상인 평균정밀도(mean average precision)
(IoU가 0.5보다 클 경우 올바른 예측으로 간주한다)



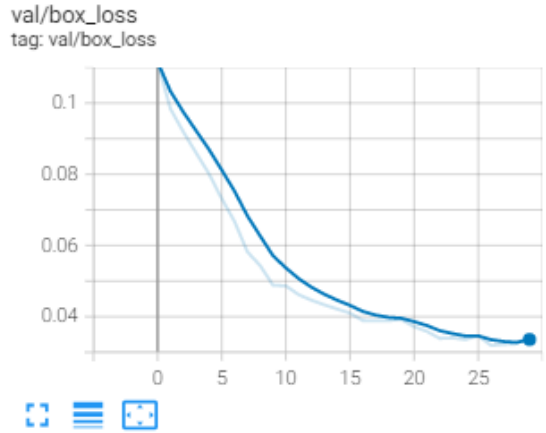
Tensorboard (30 Epoch)

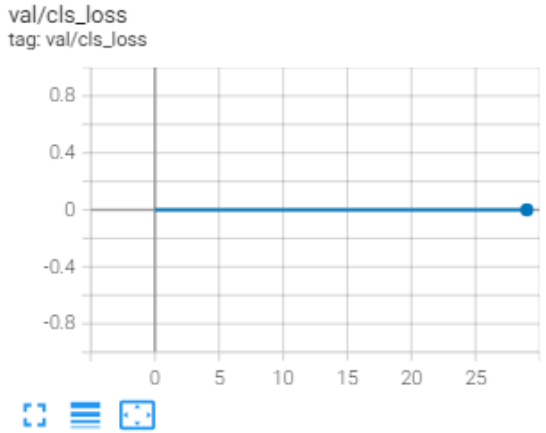
train

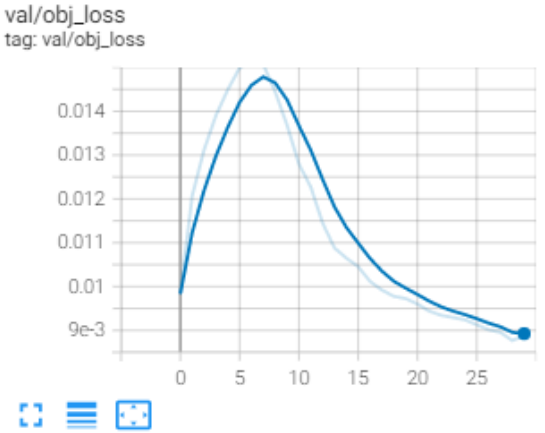




val







감사합니다