Visualization and Image Detection

AILAB
Hanyang Univ.

오늘 실습 내용

1. Wandb로 Autoencoder 시각화
2. YOLOv5 학습

1. Wandb로 Autoencoder 시각화

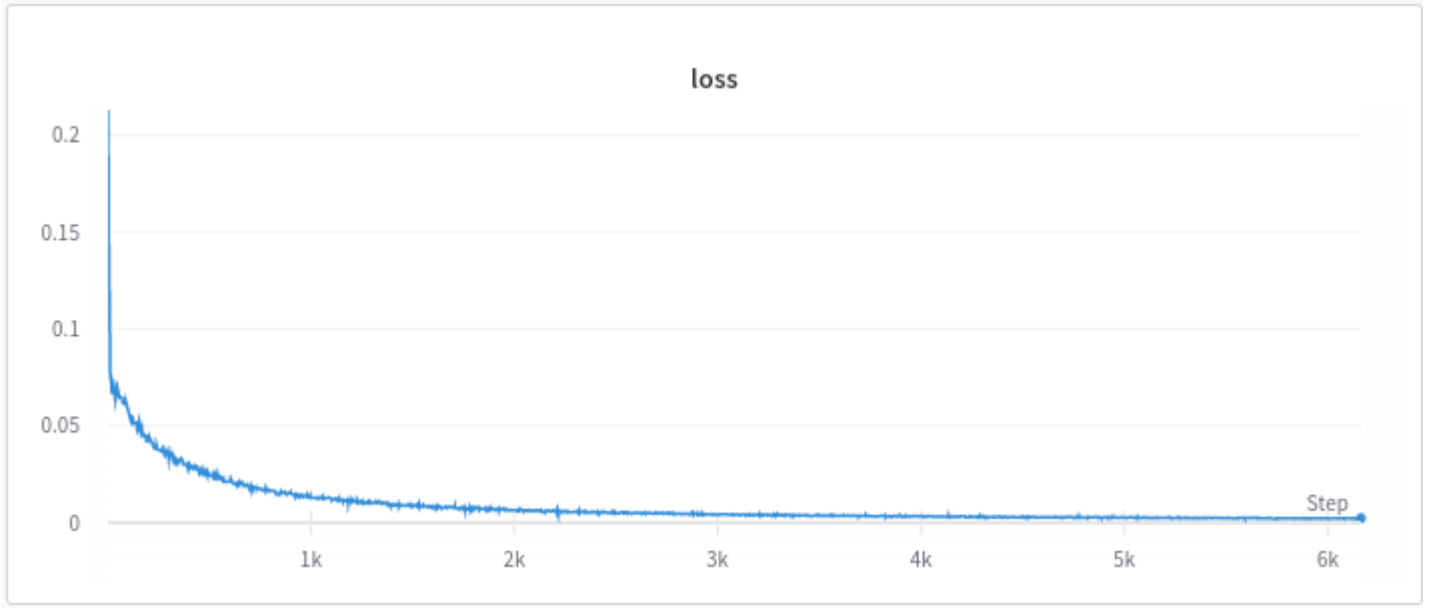
Visualization

Wandb

학습 진행 중 필요한 시각화를 해주는 툴

https://wandb.ai/site

사용을 위해 미리 Sign Up이 필요함!



Visualization

Wandb 간단한 사용법

1. Wandb install
2. 로그인
3. 프로젝트 생성 및 연결
4. 원하는 정보 적기

Visualization

Wandb 간단한 사용법

1. Wandb install
2. 로그인

```
[ ]  !pip install -q wandb
     import wandb
     wandb.login()
```

1. Wandb install

2. 로그인

Visualization

Wandb 간단한 사용법

3. 프로젝트 생성 및 연결

`wandb.init(project="[PROJECT_NAME]")`

```
[2]  config = {
       "dataset": "MNIST",
       "gpu": "colab",
       "model": "Autoencoder",
       "learning_rate": 0.001,
       "batch_size": 128,
     }
     wandb.init(project="week11_wandb_example", config=config)
     wandb.run.name = "wandb example"
```

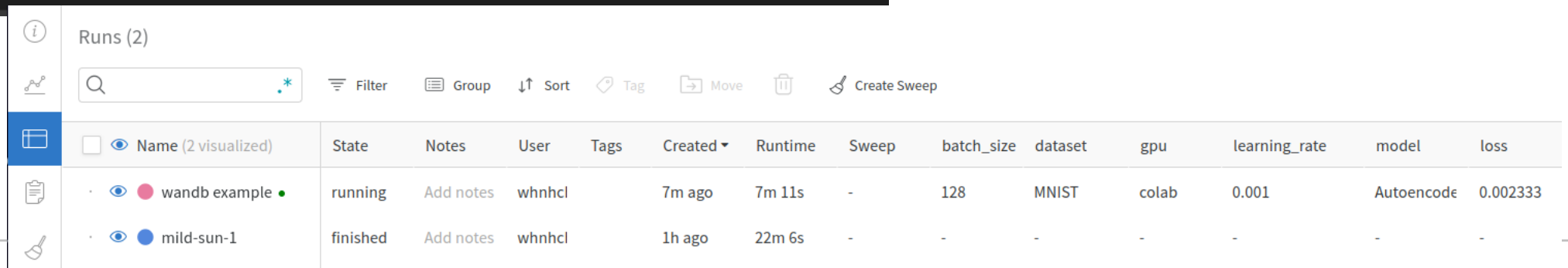
Visualization

Wandb 간단한 사용법

3. 프로젝트 생성 및 연결

`wandb.init(project="[PROJECT_NAME]", config=config)`

```
[2] config = {
      "dataset": "MNIST",
      "gpu": "colab",
      "model": "Autoencoder",
      "learning_rate": 0.001,
      "batch_size": 128,
    }
    wandb.init(project="week11_wandb_example", config=config)
    wandb.run.name = "wandb example"
```



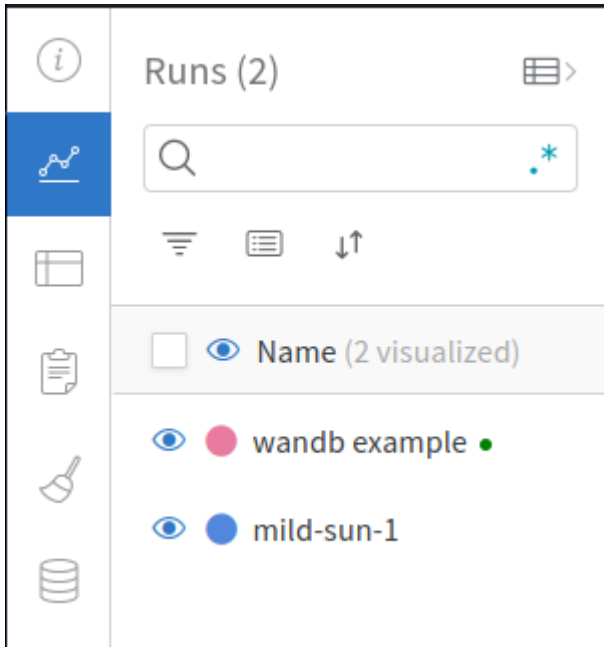
Visualization

Wandb 간단한 사용법

3. 프로젝트 생성 및 연결

`Wandb.run.name = “wandb example”`

```
[2]  config = {
        "dataset": "MNIST",
        "gpu": "colab",
        "model": "Autoencoder",
        "learning_rate": 0.001,
        "batch_size": 128,
    }
    wandb.init(project="week11_wandb_example", config=config)
    wandb.run.name = "wandb example"
```



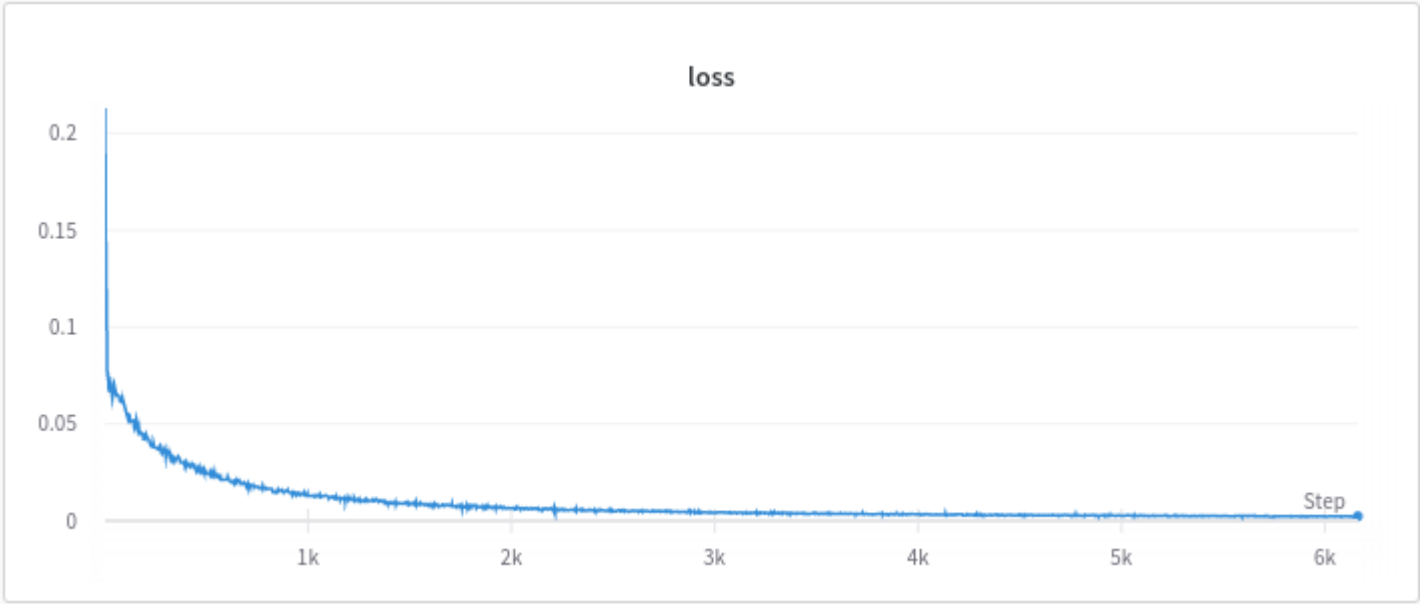
Visualization

Wandb 간단한 사용법

4. 원하는 정보 적기

`wandb.log({“loss”:loss, “[LOG_TITLE]”:[LOGGABLE_VAR]})`

```
epochs = 30

model.train()
for epoch in range(epochs):
    model.train()
    avg_cost = 0
    total_batch_num = len(train_dataloader)

    for step, batch in enumerate(train_dataloader):
      b_x, b_y = batch
      b_x = b_x.view(-1, 784).to(device)
      z, b_x_hat = model(b_x)  # forward propagation
      loss = criterion(b_x_hat, b_x) # get cost

      avg_cost += loss / total_batch_num
      optimizer.zero_grad()
      loss.backward() # backward propagation
      optimizer.step() # update parameters
      wandb.log({"loss": loss})
```



Visualization

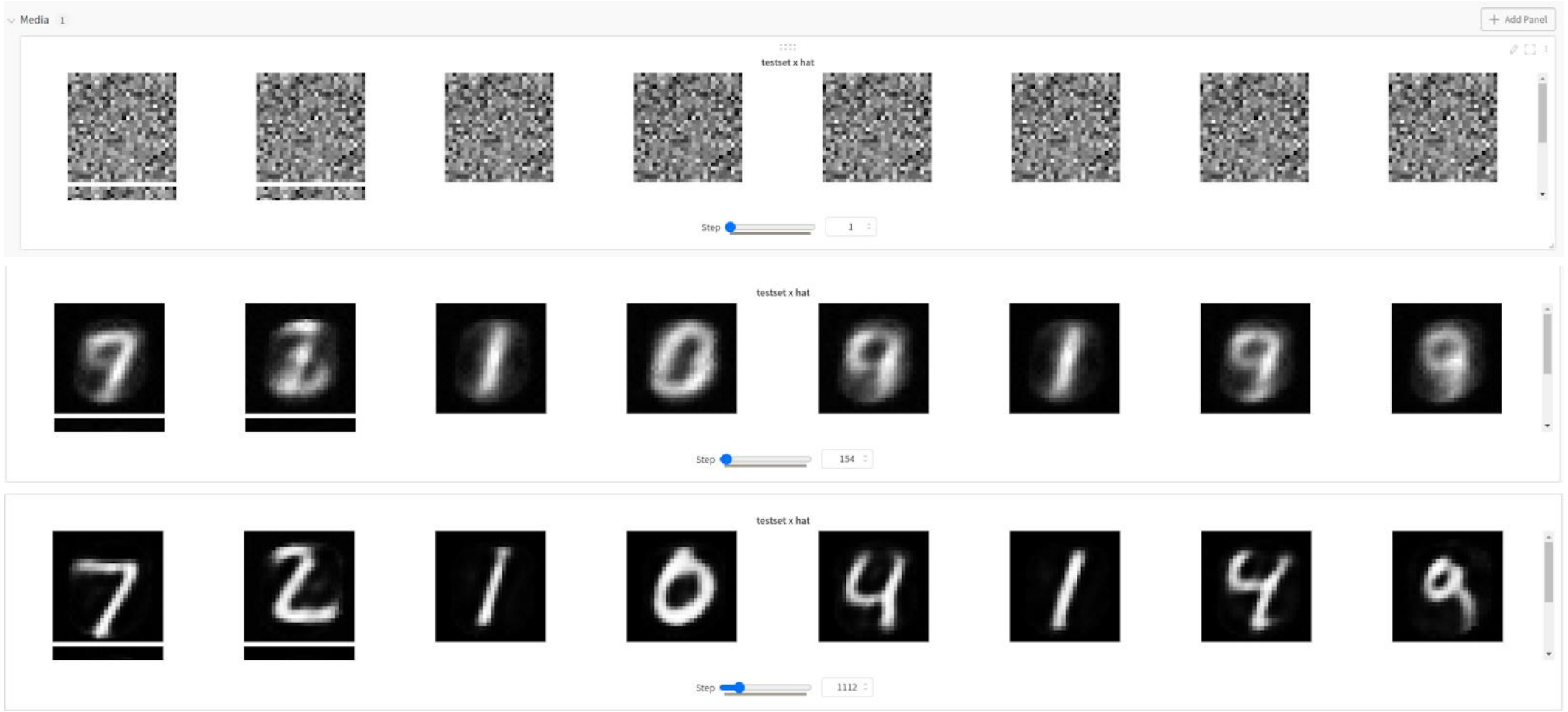
```
wandb.log({“testset x hat”:[wandb.Image(i) for i in test_output]})
```

```
samples = []
for i in range(10):
  samples.append(test_dataset[i][0].view(-1, 784).to(device))
samples = torch.stack(samples).to(device)
samples.shape
```

```
torch.Size([10, 1, 784])
```

```
epochs = 30

model.train()
for epoch in range(epochs):
    model.train()
    avg_cost = 0
    total_batch_num = len(train_dataloader)

    for step, batch in enumerate(train_dataloader):
      b_x, b_y = batch
      b_x = b_x.view(-1, 784).to(device)
      z, b_x_hat = model(b_x)  # forward propagation
      loss = criterion(b_x_hat, b_x) # get cost

      avg_cost += loss / total_batch_num
      optimizer.zero_grad()
      loss.backward() # backward propagation
      optimizer.step() # update parameters
      wandb.log({"loss": loss})

    # observe differences
      if step % 5 == 0:
        model.eval()
        with torch.no_grad():
          test_z, test_output = model(samples)

        test_output = test_output.detach().cpu().reshape(10,28,28)
        wandb.log({'testset x hat': [wandb.Image(i) for i in test_output]})
```

Visualization



Visualization

run 끼리 비교 가능

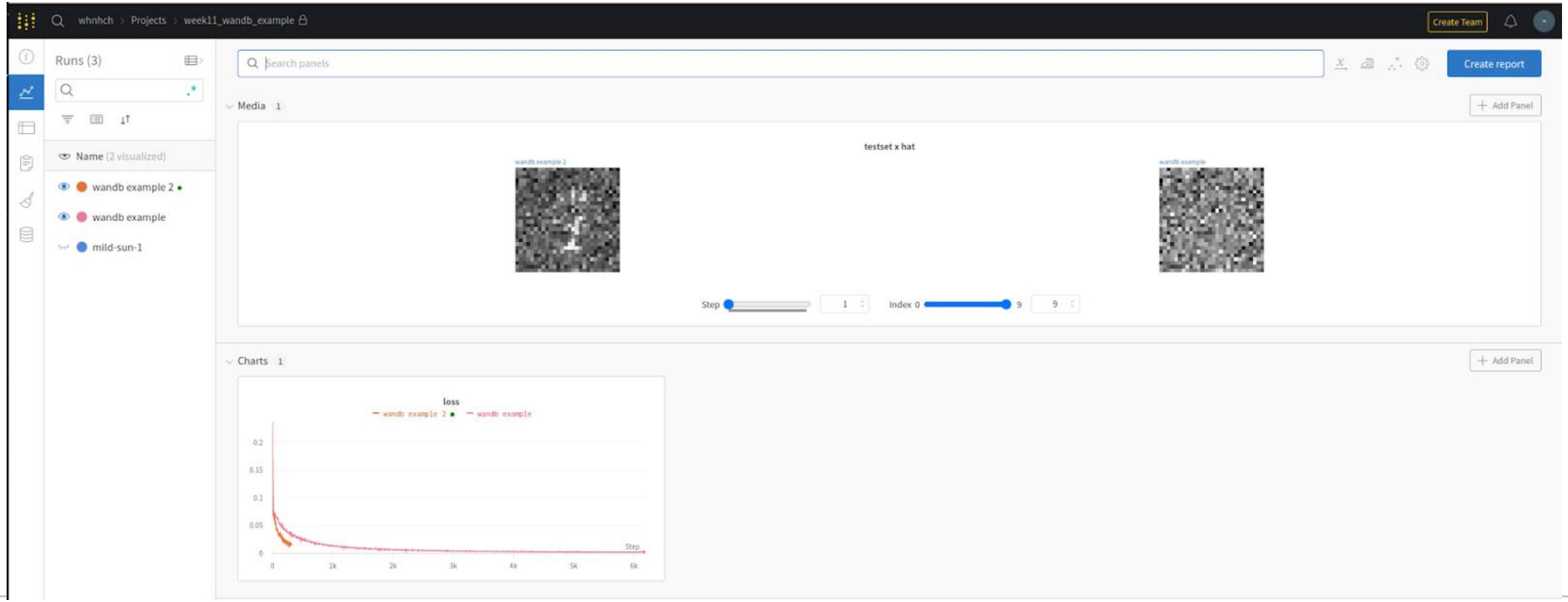
```
[2]  config = {
       "dataset": "MNIST",
       "gpu": "colab",
       "model": "Autoencoder",
       "learning_rate": 0.001,
       "batch_size": 128,
     }
     wandb.init(project="week11_wandb_example", config=config)
     wandb.run.name = "wandb example"
```

```
config = {
  "dataset": "MNIST",
  "gpu": "colab",
  "model": "Autoencoder",
  "learning_rate": 0.005,
  "batch_size": 128,
}
wandb.init(project="week11_wandb_example", config=config)
wandb.run.name = "wandb example 2"
```

```
optimizer = optim.Adam(model.parameters(), lr=0.005) # set optimizer
```

Visualization

run 끼리 비교 가능



2. YOLOv5

YOLOv5

YOLOv5

https://colab.research.google.com/github/ultralytics/yolov5/blob/master/tutorial.ipynb

object detection model, pre-trained model with coco2017

이미지 비디오 등에서 사용가능

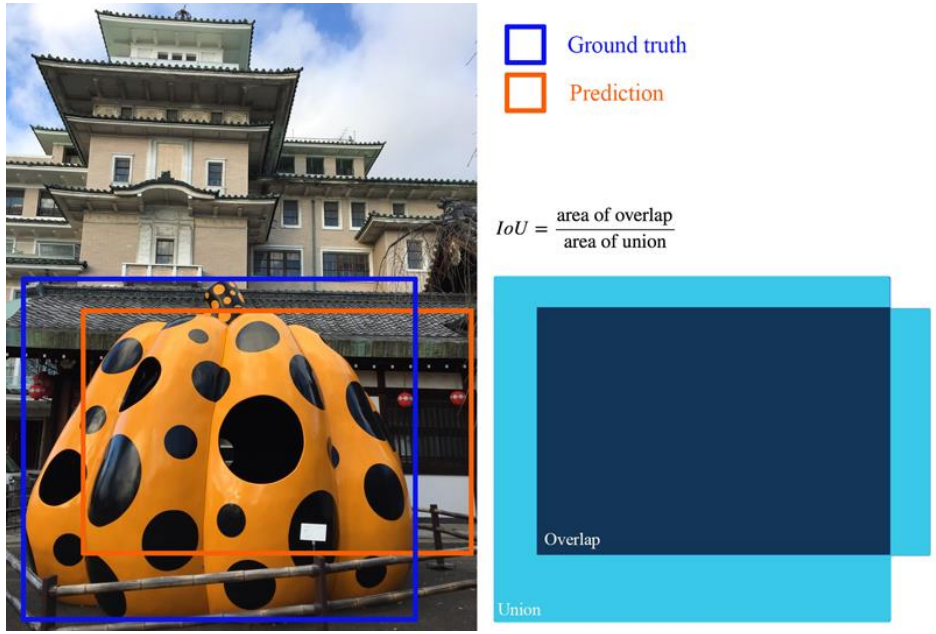
다양한 사이즈의 모델 제공

Model	size (pixels)	mAP^{val} 0.5:0.95	mAP^{val} 0.5	Speed CPU b1 (ms)	Speed V100 b1 (ms)	Speed V100 b32 (ms)	params (M)	FLOPs @640 (B)
YOLOv5n	640	28.0	45.7	**45**	**6.3**	**0.6**	**1.9**	**4.5**
YOLOv5s	640	37.4	56.8	98	6.4	0.9	7.2	16.5
YOLOv5m	640	45.4	64.1	224	8.2	1.7	21.2	49.0
YOLOv5l	640	49.0	67.3	430	10.1	2.7	46.5	109.1
YOLOv5x	640	50.7	68.9	766	12.1	4.8	86.7	205.7
YOLOv5n6	1280	36.0	54.4	153	8.1	2.1	3.2	4.6
YOLOv5s6	1280	44.8	63.7	385	8.2	3.6	12.6	16.8
YOLOv5m6	1280	51.3	69.3	887	11.1	6.8	35.7	50.0
YOLOv5l6	1280	53.7	71.3	1784	15.8	10.5	76.8	111.4
YOLOv5x6 + TTA	1280 1536	55.0 **55.8**	72.7 **72.7**	3136 -	26.2 -	19.4 -	140.7 -	209.8 -

YOLOv5

IoU

More generally, IoU is a measure of the overlap between two bounding boxes.



출처 : https://jonathan-hui.medium.com/map-mean-average-precision-for-object-detection-45c121a31173

YOLOv5

Model	size (pixels)	mAP^{val} 0.5:0.95	mAP^{val} 0.5	Speed CPU b1 (ms)	Speed V100 b1 (ms)	Speed V100 b32 (ms)	params (M)	FLOPs @640 (B)

AP (average precision)

5개의 사과 detection 일때

~~하나의 사과 클래스에서 확률에 따라 ranking~~

Correct? : IoU > 0.5

Rank	Correct?	Precision	Recall
1	True	1.0	0.2
2	True	1.0	0.4
3	False	0.67	0.4
4	False	0.5	0.4
5	False	0.4	0.4
6	True	0.5	0.6
7	True	0.57	0.8
8	False	0.5	0.8
9	False	0.44	0.8

Correct인 것 중에 실제로 정답인 것

$$Precision = \frac{TP}{TP + FP}$$

$$Recall = \frac{TP}{TP + FN}$$

TP = True positive

TN = True negative

FP = False positive

FN = False negative

정답인 것 중에 Correct인 것

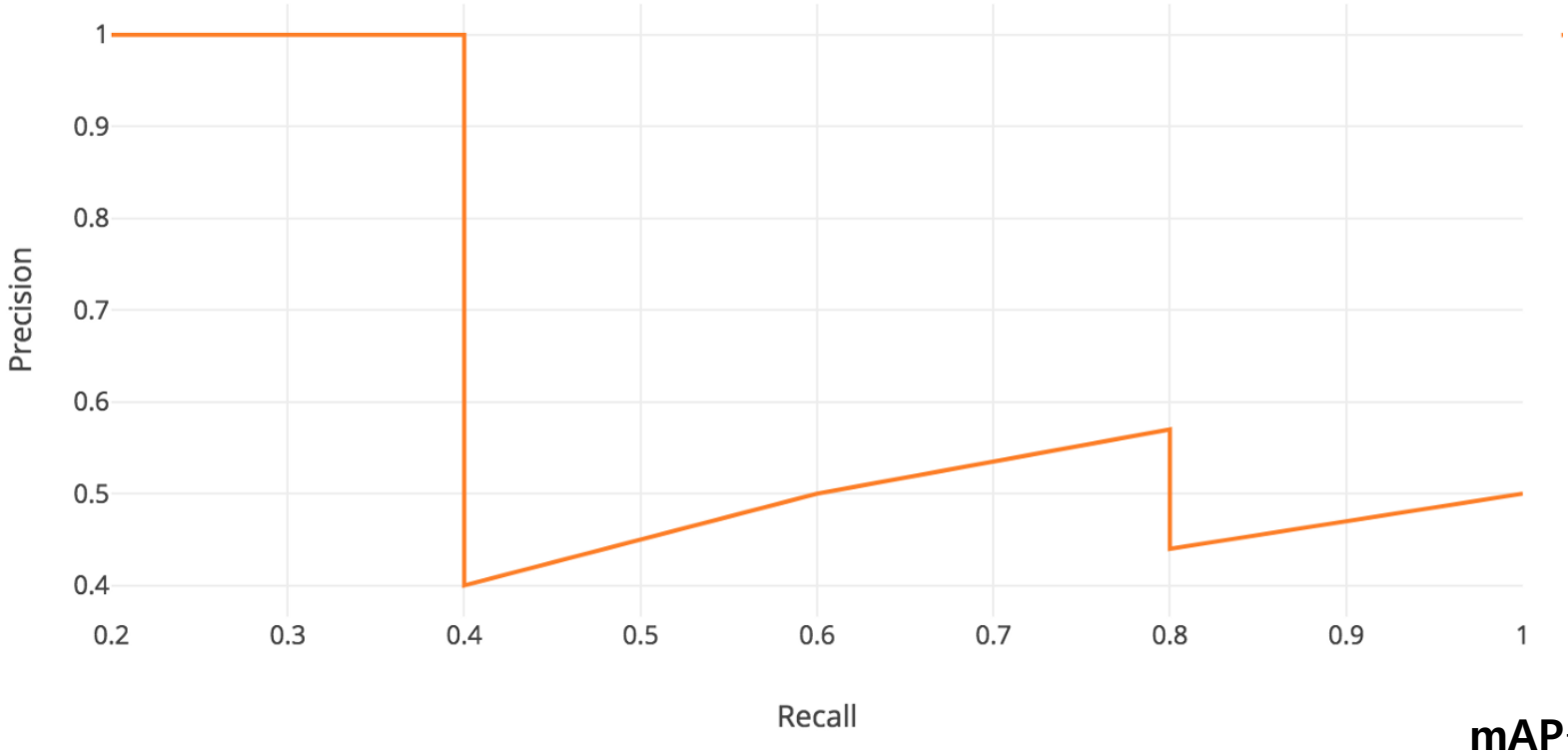
출처 : https://jonathan-hui.medium.com/map-mean-average-precision-for-object-detection-45c121a31173

YOLOv5

AP

앞에서 계산한 precision과 recall로 그래프를 그릴 수 있다.
AP는 이 그래프의 구역을 계산한 값

$$\mathrm{AP} = \int_0^1 p(r)dr$$



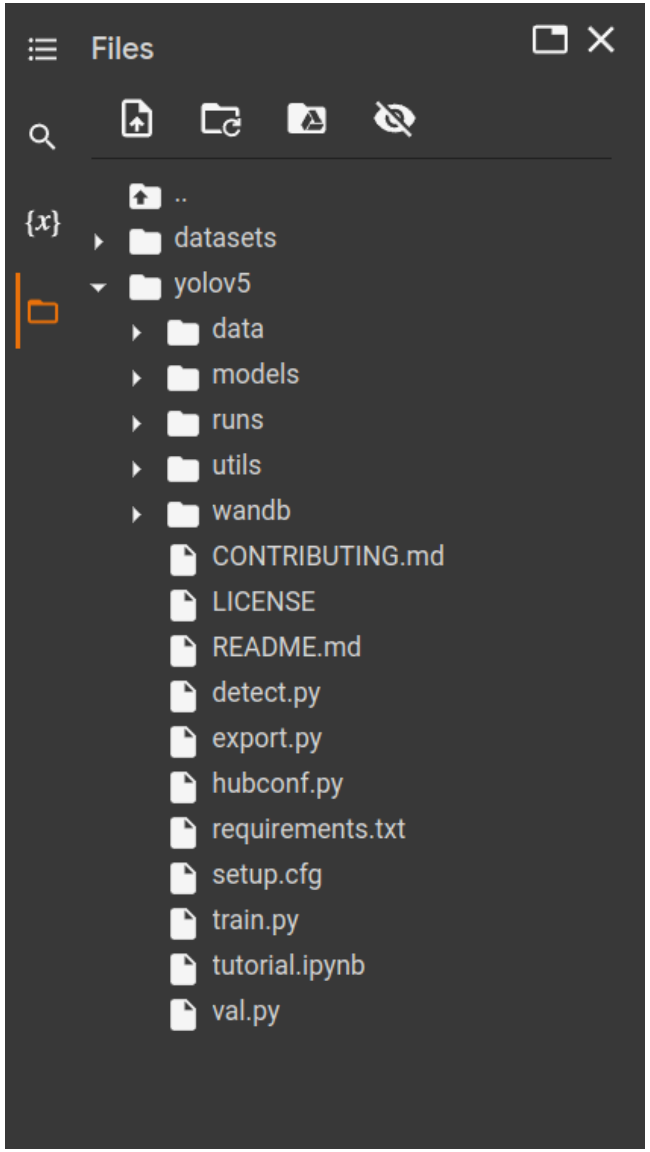
mAP는 모든 클래스에서 이 값을 평균낸 것 (meanAP)

출처 : https://jonathan-hui.medium.com/map-mean-average-precision-for-object-detection-45c121a31173

YOLOv5

YOLOv5 설치

```
[ ]  !git clone https://github.com/ultralytics/yolov5  # clone
     %cd yolov5
     %pip install -qr requirements.txt  # install

     import torch
     import utils
     display = utils.notebook_init()  # checks
```



YOLOv5

YOLOv5 train

train.py로 학습

train.py

```
467
468         callbacks.run('on_train_end', last, best, plots, epoch, results)
469
470     torch.cuda.empty_cache()
471     return results
472
473
474 def parse_opt(known=False):
475     parser = argparse.ArgumentParser()
476     parser.add_argument('--weights', type=str, default=ROOT / 'yolov5s.pt', help='initial weights path')
477     parser.add_argument('--cfg', type=str, default='', help='model.yaml path')
478     parser.add_argument('--data', type=str, default=ROOT / 'data/coco128.yaml', help='dataset.yaml path')
479     parser.add_argument('--hyp', type=str, default=ROOT / 'data/hyps/hyp.scratch-low.yaml', help='hyperparameters path')
480     parser.add_argument('--epochs', type=int, default=300)
481     parser.add_argument('--batch-size', type=int, default=16, help='total batch size for all GPUs, -1 for autobatch')
482     parser.add_argument('--imgsz', '--img', '--img-size', type=int, default=640, help='train, val image size (pixels)')
483     parser.add_argument('--rect', action='store_true', help='rectangular training')
484     parser.add_argument('--resume', nargs='?', const=True, default=False, help='resume most recent training')
485     parser.add_argument('--nosave', action='store_true', help='only save final checkpoint')
486     parser.add_argument('--noval', action='store_true', help='only validate final epoch')
487     parser.add_argument('--noautoanchor', action='store_true', help='disable AutoAnchor')
488     parser.add_argument('--noplots', action='store_true', help='save no plot files')
489     parser.add_argument('--evolve', type=int, nargs='?', const=300, help='evolve hyperparameters for x generations')
490     parser.add_argument('--bucket', type=str, default='', help='gsutil bucket')
491     parser.add_argument('--cache', type=str, nargs='?', const='ram', help='--cache images in "ram" (default) or "disk"')
492     parser.add_argument('--image-weights', action='store_true', help='use weighted image selection for training')
493     parser.add_argument('--device', default='', help='cuda device, i.e. 0 or 0,1,2,3 or cpu')
494     parser.add_argument('--multi-scale', action='store_true', help='vary img-size +/- 50%%')
495     parser.add_argument('--single-cls', action='store_true', help='train multi-class data as single-class')
496     parser.add_argument('--optimizer', type=str, choices=['SGD', 'Adam', 'AdamW'], default='SGD', help='optimizer')
497     parser.add_argument('--sync-bn', action='store_true', help='use SyncBatchNorm, only available in DDP mode')
498     parser.add_argument('--workers', type=int, default=8, help='max dataloader workers (per RANK in DDP mode)')
499     parser.add_argument('--project', default=ROOT / 'runs/train', help='save to project/name')
500     parser.add_argument('--name', default='exp', help='save to project/name')
501     parser.add_argument('--exist-ok', action='store_true', help='existing project/name ok, do not increment')
502     parser.add_argument('--quad', action='store_true', help='quad dataloader')
503     parser.add_argument('--cos-lr', action='store_true', help='cosine LR scheduler')
```

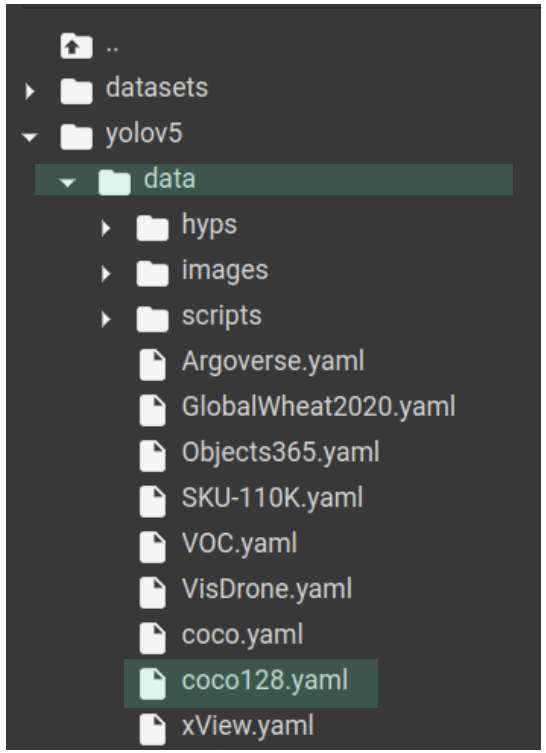
--img : input image size
--batch : batch size
--epochs : training epochs
--data : data path of yaml file
--cache : cache images for faster running
--project : 생성할 프로젝트 이름 (하위폴더/wandb에 적용)
--name : 생성할 run 이름 (하위폴더/wandb에 적용)
--hyp : learning rate와 같은 hyperparameter yaml file path

YOLOv5

YOLOv5

학습 진행할 dataset은 coco128

다른 데이터는 커서 학습시간이 오래 걸림



coco128.yaml

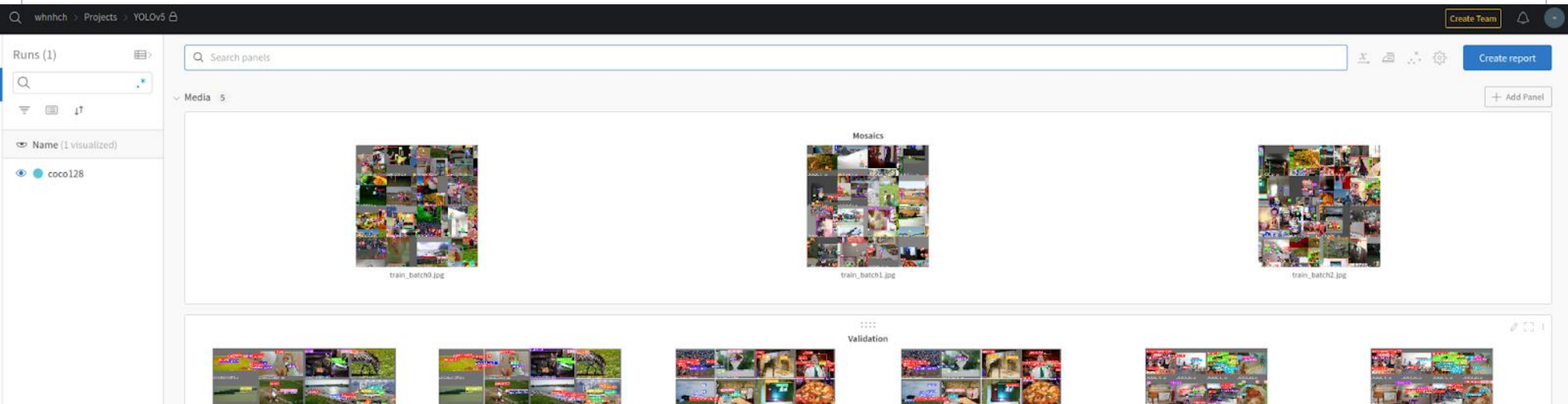
```yaml
# YOLOv5 🚀 by Ultralytics, GPL-3.0 license
# COCO128 dataset https://www.kaggle.com/ultralytics/coco128 (first 128 images from COCO train2017) by
# Example usage: python train.py --data coco128.yaml
# parent
# ├── yolov5
# └── datasets
#     └── coco128  ← downloads here (7 MB)


# Train/val/test sets as 1) dir: path/to/imgs, 2) file: path/to/imgs.txt, or 3) list: [path/to/imgs1, p
path: ../datasets/coco128  # dataset root dir
train: images/train2017  # train images (relative to 'path') 128 images
val: images/train2017  # val images (relative to 'path') 128 images
test:  # test images (optional)

# Classes
nc: 80  # number of classes
names: ['person', 'bicycle', 'car', 'motorcycle', 'airplane', 'bus', 'train', 'truck', 'boat', 'traffic
        'fire hydrant', 'stop sign', 'parking meter', 'bench', 'bird', 'cat', 'dog', 'horse', 'sheep',
        'elephant', 'bear', 'zebra', 'giraffe', 'backpack', 'umbrella', 'handbag', 'tie', 'suitcase',
        'skis', 'snowboard', 'sports ball', 'kite', 'baseball bat', 'baseball glove', 'skateboard', 'su
        'tennis racket', 'bottle', 'wine glass', 'cup', 'fork', 'knife', 'spoon', 'bowl', 'banana', 'ap
        'sandwich', 'orange', 'broccoli', 'carrot', 'hot dog', 'pizza', 'donut', 'cake', 'chair', 'couc
        'potted plant', 'bed', 'dining table', 'toilet', 'tv', 'laptop', 'mouse', 'remote', 'keyboard',
        'microwave', 'oven', 'toaster', 'sink', 'refrigerator', 'book', 'clock', 'vase', 'scissors', 't
        'hair drier', 'toothbrush']  # class names


# Download script/URL (optional)
download: https://ultralytics.com/assets/coco128.zip
```

YOLOv5

YOlOv5 (YOLO)

```
%pip install -q wandb
import wandb

wandb.login()
```

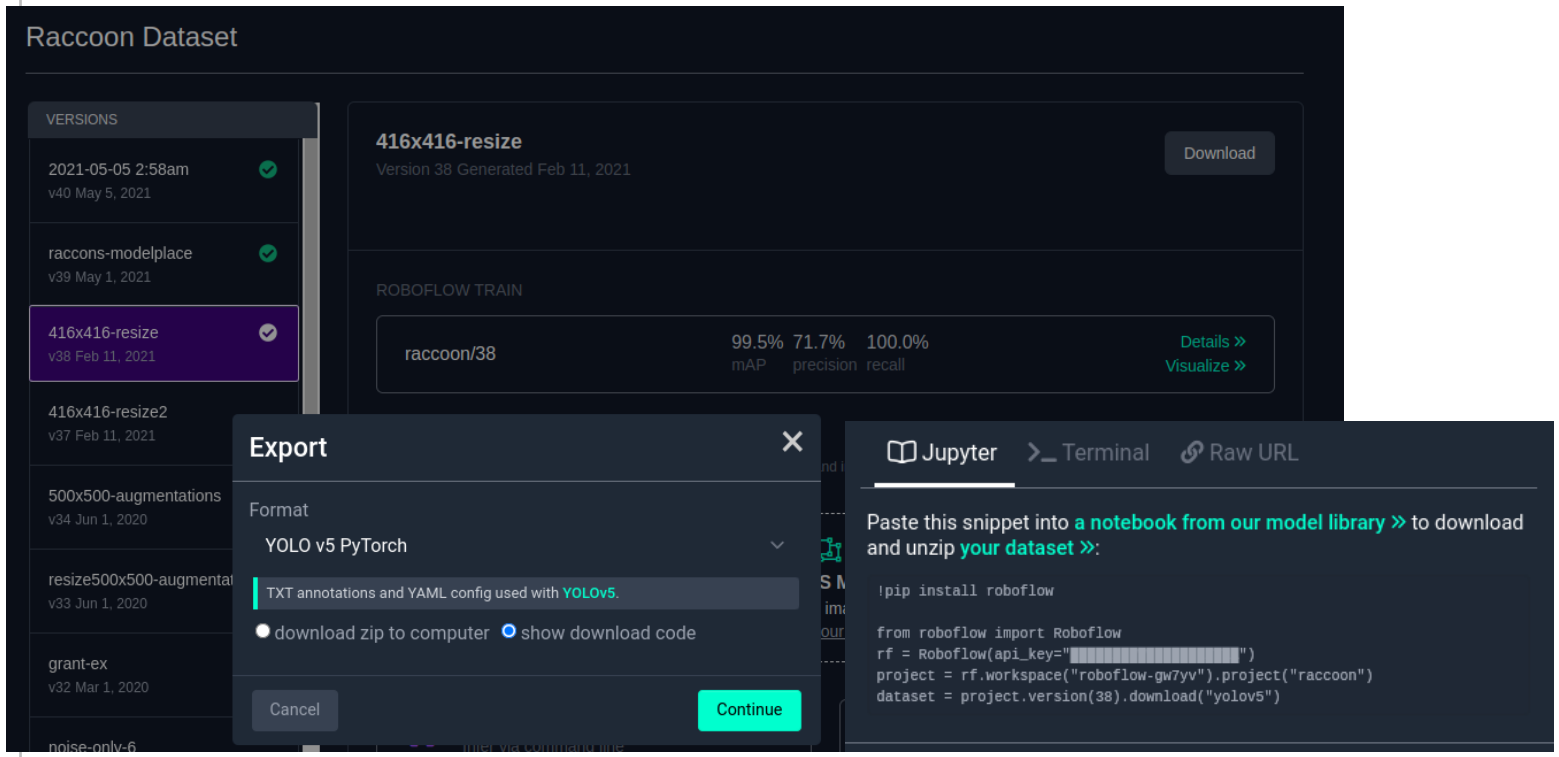
```
!python train.py --batch 16 --img 640 --epochs 3 --data coco128.yaml --weights yolov5s.pt --cache --project yolov5 --name coco128
```



YOLOv5

Custom Detection Dataset

https://app.roboflow.com/login
https://universe.roboflow.com/roboflow-gw7yv/raccoon/38

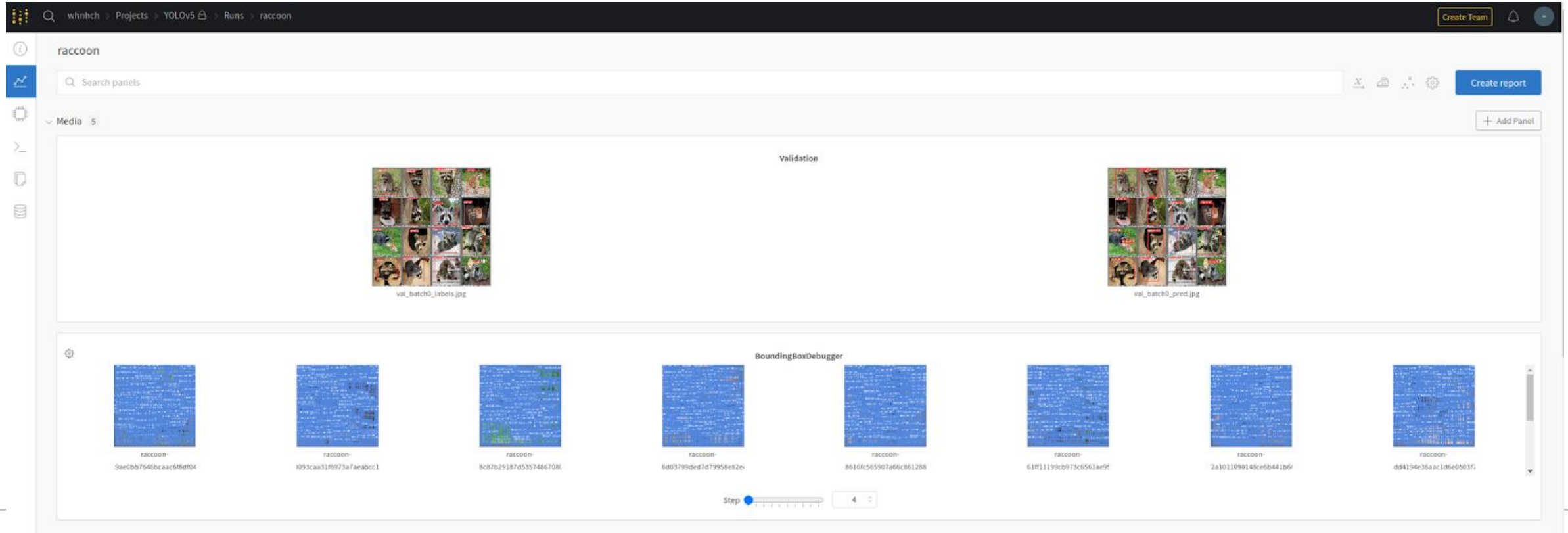



```
[3]  !pip install roboflow

     from roboflow import Roboflow
     rf = Roboflow(api_key="F2JOKGFmWjqifoNK4VHA")
     project = rf.workspace("roboflow-gw7yv").project("raccoon")
     dataset = project.version(38).download("yolov5")
```

YOLOv5

Custom Detection Dataset

```
!python train.py --batch 16 --img 416 --epochs 50 --data Raccoon-38/data.yaml --weights yolov5s.pt --cache --project yolov5 --name raccoon
```

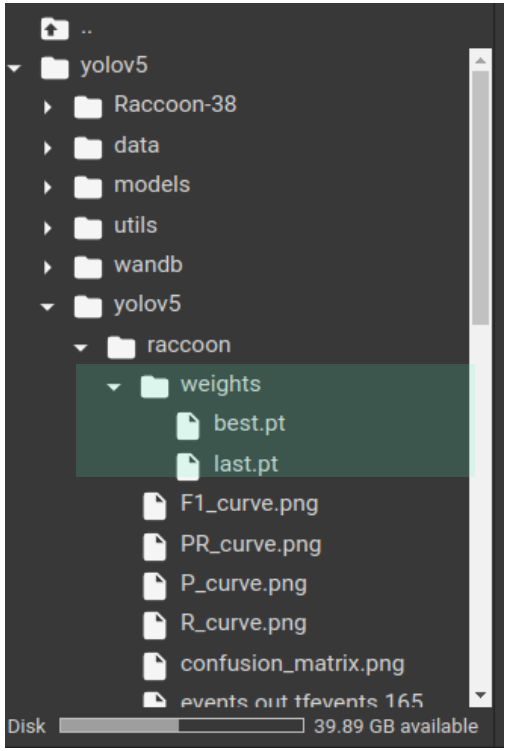


YOLOv5

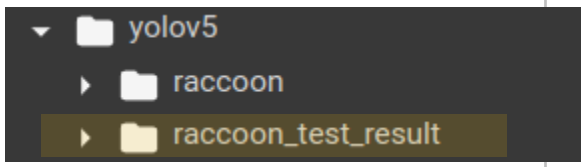
Custom Detection Dataset

detect.py 사용해 test set detection 결과 확인

학습한 pt path




```
!python detect.py --weights yolov5/raccoon/weights/best.pt --img 416 --source Raccoon-38/test/images --project yolov5 --name raccoon_test_result
```



나온 결과 저장하고 싶을 때 zip으로 압축해서 한번에 받기

```
!zip -r yolov5/raccoon_test_result.zip yolov5/raccoon_test_result
```

오늘 실습 내용

1. YOLOv5로 새로운 데이터 학습해보기
https://universe.roboflow.com/joseph-nelson/mask-wearing/1