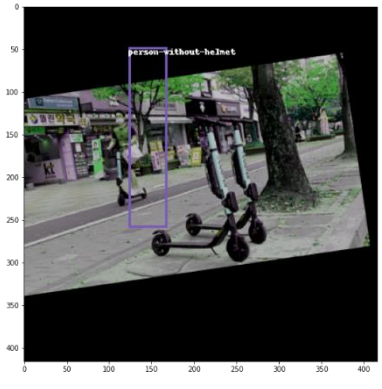
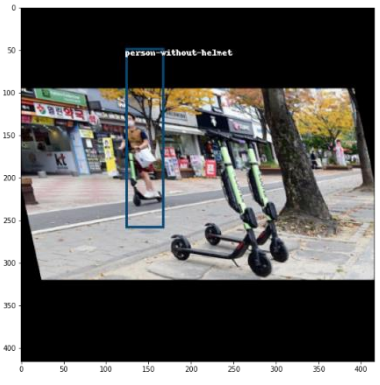
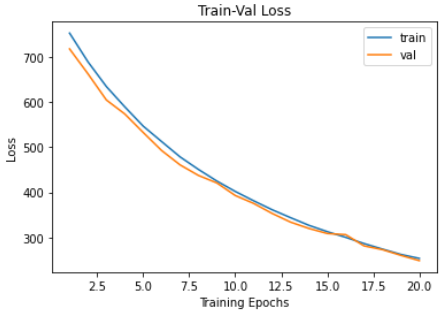
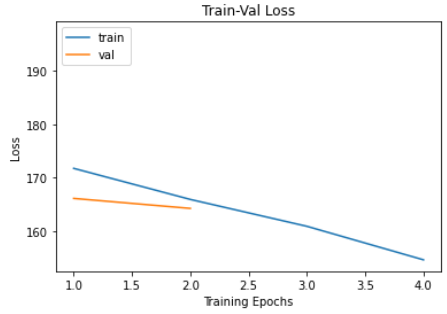
YOLO V3 OBJECT DETECTION

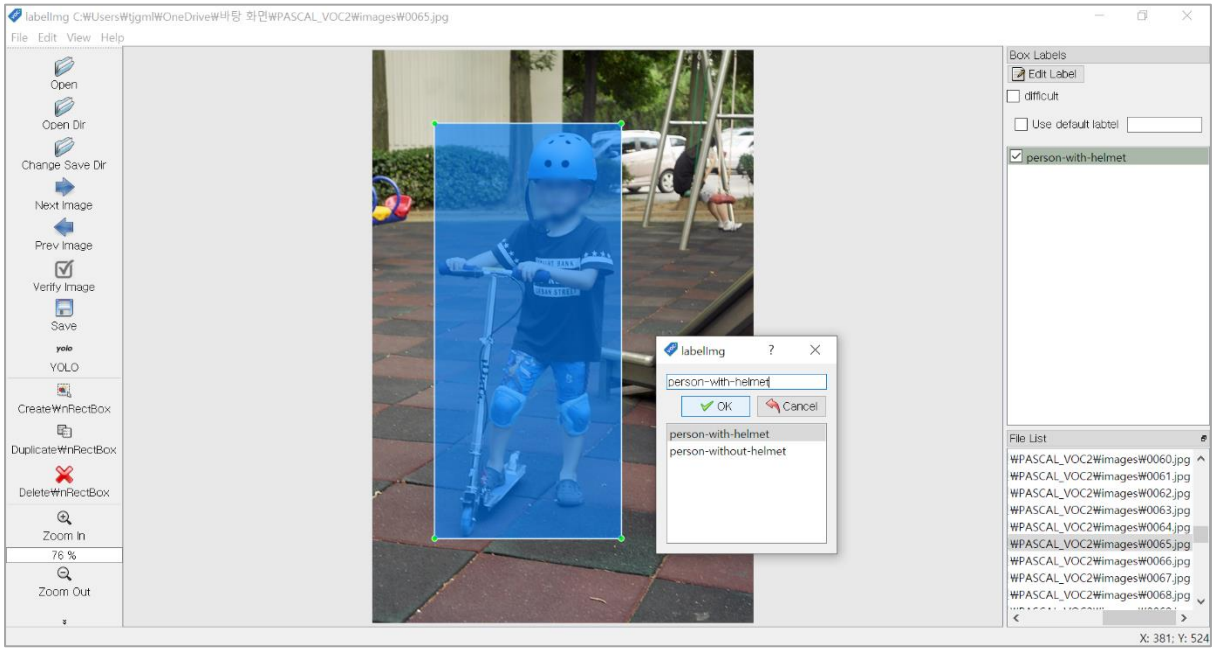
Object	전동 킥보드 주행시 헬멧 미착용 객체 인식

SUMMARY 현재까지의 진행 요약

Model	1차 모델링 (~11.04)		2차 모델링 (~11.08)	
raw data	image 105개		grayscale image 추가로 2배 증폭	
Train:Test	8:2			
인식 정확도 비교	 (bad)  (soso)		 (good)  (good)	
T-V Loss Plot				
Epoch	20		1차 시도 10 -> 2차 시도 4	
Latest Loss	train loss	253.902252	train loss	154.607904
	val loss	248.937134	val loss	197.124678

ANNOTAION 데이터 전처리

Tool	python anaconda labelimg-1.8.1
label	2개 (person-with-helmet, person-without-helmet)
How?	1. 바운딩 박스를 지정하여 라벨링 2. txt 파일 저장



DATA LOAD

Env.	Colaboratory, GPU 사용
데이터	구글 드라이브로 연동

```
from google.colab import drive
drive.mount('/content/drive')
```

1ST TRIAL 1차 모델링 시도 (~11.04)

images	image 105개 (color 103 + grayscale 2)
labels	images에 해당하는 txt 파일
train.csv	**8**:2 (84개)
test.csv	8:**2** (21개)

■ 패키지 설치 & 로드

- 토치 버전에 유의, ToTensorV2에 주의

```
# install transformation package
!pip install -U albumentations
!pip install torchvision
!pip install torch==1.9.0
```

```
import torch
from torch import nn
from torch import optim
from torch.optim.lr_scheduler import ReduceLROnPlateau
from torch.utils.data import Dataset
from torch.utils.data import DataLoader
from torchvision import utils
from torchsummary import summary
import torchvision.transforms.functional as TF
from torchvision.transforms.functional import to_pil_image
from PIL import Image, ImageDraw, ImageFont
import matplotlib.pyplot as plt
import cv2
import os
import copy
import numpy as np
import pandas as pd
import random
import albumentations as A
from albumentations.pytorch import ToTensorV2  #ToTensor -> ToTensorV2 로 변경됨
%matplotlib inline
device = torch.device('cuda' if torch.cuda.is_available() else 'cpu')
```

■ 라벨 클래스 지정

```
# VOC class names
classes = [
    "person-with-helmet",
    "person-without-helmet"
]
```

- 세부 정보

person-with-helmet	헬멧을 쓴 사람
person-without-helmet	헬멧을 쓰지 않은 사람

■ VOCDataset class

```
class VOCDataset(Dataset):
    def __init__(self, csv_file, img_dir, label_dir, transform=None, trans_params=N
one):
        self.annotations = pd.read_csv(csv_file)
        self.img_dir = img_dir
        self.label_dir = label_dir
        self.transform = transform
        self.trans_params = trans_params

    def __len__(self):
        return len(self.annotations)

    def __getitem__(self, index):
        label_path = '/content/PASCAL_VOC2/labels/0100.txt'
        img_path = '/content/PASCAL_VOC2/images/0100.jpg'
        image = np.array(Image.open(img_path).convert("RGB"))

        labels = None
        if os.path.exists(label_path):
            labels = np.array(np.roll(np.loadtxt(fname=label_path, delimiter=" ", n
dmin=2), 4, axis=1).tolist())

        if self.transform:
            # apply albumentations
            augmentations = self.transform(image=image, bboxes=labels)
            image = augmentations['image']
            targets = augmentations['bboxes']

            # for DataLoader
            # lables: ndarray -> tensor
            # dimension: [batch, cx, cy, w, h, class]
            if targets is not None:
                targets = torch.zeros((len(labels), 6))
                targets[:, 1:] = torch.tensor(labels)
        else:
            targets = labels

        return image, targets, label_path
```

■ 트레인 파일, 라벨, 이미지 디렉터리 지정

```
train_csv_file = '/content/PASCAL_VOC2/train.csv'
label_dir = '/content/PASCAL_VOC2/labels'
img_dir = '/content/PASCAL_VOC2/images'

train_ds = VOCDataset(train_csv_file, img_dir, label_dir)

img, labels, _ = train_ds[1]
print('number of data:',len(train_ds))
print('image size:', img.shape, type(img)) # HxWxC
print('labels shape:', labels.shape, type(labels))  # x1,y1,x2,y2
print('lables \n', labels)
```

- output

```
number of data: 104
image size: (277, 500, 3) <class 'numpy.ndarray'>
labels shape: (1, 5) <class 'numpy.ndarray'>
lables
 [[0.351    0.370036 0.106    0.509025 1.      ]]
```

■ 테스트 파일, 라벨, 이미지 디렉터리 지정

```
val_csv_file = '/content/PASCAL_VOC2/test.csv'
label_dir = '/content/PASCAL_VOC2/labels'
img_dir = '/content/PASCAL_VOC2/images'

val_ds = VOCDataset(val_csv_file, img_dir, label_dir)

img, labels, _ = val_ds[1]
print('number of data:',len(val_ds))
print('image size:', img.shape, type(img))
print('labels shape:', labels.shape, type(labels))
print('lables \n', labels)
```

- output

```
number of data: 20
image size: (277, 500, 3) <class 'numpy.ndarray'>
labels shape: (1, 5) <class 'numpy.ndarray'>
lables
 [[0.351    0.370036 0.106    0.509025 1.      ]]
```

■ 정규화 정의

```
# transforms 정의하기
IMAGE_SIZE = 416
scale = 1.0

# for train
train_transforms = A.Compose([
```

```
        # 이미지의 maxsize를 max_size로 rescale
        A.LongestMaxSize(max_size=int(IMAGE_SIZE * scale)),
        # min_size보다 작으면 pad
        A.PadIfNeeded(min_height=int(IMAGE_SIZE * scale), min_width=int(IMAGE_SIZE
* scale), border_mode=cv2.BORDER_CONSTANT),
        # random crop
        A.RandomCrop(width=IMAGE_SIZE, height=IMAGE_SIZE),
        # brightness, contrast, saturation을 무작위로 변경
        A.ColorJitter(brightness=0.6, contrast=0.6, saturation=0.6, hue=0.6, p=0.4)
,
        # transforms 중 하나를 선택해 적용
        A.OneOf([
                # shift, scale, rotate 를 무작위로 적용
                A.ShiftScaleRotate(rotate_limit=20, p=0.5, border_mode=cv2.BORDER_
CONSTANT),
                # affine 변환
                A.IAAAffine(shear=15, p=0.5, mode='constant')
        ], p=1.0),
        # 수평 뒤집기
        A.HorizontalFlip(p=0.5),
        # blur
        A.Blur(p=0.1),
        # Contrast Limited Adaptive Histogram Equalization 적용
        A.CLAHE(p=0.1),
        # 각 채널의 bit 감소
        A.Posterize(p=0.1),
        # grayscale로 변환
        A.ToGray(p=0.1),
        # 무작위로 channel을 섞기
        A.ChannelShuffle(p=0.05),
        # normalize
        A.Normalize(mean=[0,0,0], std=[1,1,1], max_pixel_value=255),
        ToTensorV2()
        ],
        # (x1, y1, x2, y2) -> (cx, cy, w, h)
        bbox_params=A.BboxParams(format='yolo', min_visibility=0.4, label_fields=[]
)
        )

# for validation
val_transforms = A.Compose([
        A.LongestMaxSize(max_size=int(IMAGE_SIZE * scale)),
        A.PadIfNeeded(min_height=int(IMAGE_SIZE * scale), min_width=int(IMAGE_SIZE
* scale), border_mode=cv2.BORDER_CONSTANT),
        A.Normalize(mean=[0, 0, 0], std=[1, 1, 1], max_pixel_value=255),
        ToTensorV2(),
        ],
```

```
        bbox_params=A.BboxParams(format='yolo', min_visibility=0.4, label_fields=[]
)
        )

train_ds.transform = train_transforms
val_ds.transform = val_transforms
```

■ 정규화된 x, y, w, h를 이미지 크기에 맞게 변경

```
def rescale_bbox(bb, W, H):
    x,y,w,h = bb
    return [x*W, y*H, w*W, h*H]

# 바운딩 박스 색상
COLORS = np.random.randint(0, 255, size=(80,3),dtype='uint8')

# image 출력 함수 정의
def show_img_bbox(img, targets, classes=classes):
    if torch.is_tensor(img):
        img=to_pil_image(img)
    if torch.is_tensor(targets):
        targets=targets.numpy()[:,1:]

    W, H = img.size
    draw = ImageDraw.Draw(img)

    for tg in targets:
        id_=int(tg[4])
        bbox=tg[:4]
        bbox=rescale_bbox(bbox,W,H)
        xc,yc,w,h = bbox

        color = [int(c) for c in COLORS[id_]]
        name=classes[id_]

        draw.rectangle(((xc-w/2, yc-
h/2), (xc+w/2, yc+h/2)), outline=tuple(color), width=3)
        draw.text((xc-w/2, yc-h/2), name, fill=(255,255,255,0))
    plt.imshow(np.array(img))
```

■ 정규화 적용 샘플 이미지 확인

```
np.random.seed(2)

grid_size = 2
rnd_ind = np.random.randint(0, len(train_ds), grid_size)
print('image indices:',rnd_ind)
```

```
plt.figure(figsize=(20, 20))
for i, indice in enumerate(rnd_ind):
    img, label, _ = train_ds[indice]
    plt.subplot(1, grid_size, i+1)
    show_img_bbox(img, label)
```

[왼쪽] 좌우 반전, 약간의 기울기 => 사람을 아예 인식하지 못함 => bad

[오른쪽] 채도 변경, 상대적으로 심한 기울기 => 바운딩 박스가 50%정도 벗어남 => soso

■ collate_fn 정의

- collate_fn : 데이터 로더의 인자로 사용되며 배치 단위로 imgs와 targets를 묶는 역할

```
def collate_fn(batch):
    imgs, targets, paths = list(zip(*batch))
    # 빈 박스 제거하기
    targets = [boxes for boxes in targets if boxes is not None]
    # index 설정하기
    for b_i, boxes in enumerate(targets):
        boxes[:, 0] = b_i
    targets = torch.cat(targets, 0)
    imgs = torch.stack([img for img in imgs])
    return imgs, targets, paths
```

```
# make DataLoader
train_dl = DataLoader(train_ds, batch_size=4, shuffle=True, collate_fn=collate_fn)
val_dl = DataLoader(val_ds, batch_size=4, shuffle=True, collate_fn=collate_fn)

# check train_dl
torch.manual_seed(1)
for imgs_batch, tg_batch, path_batch in train_dl:
    break
```

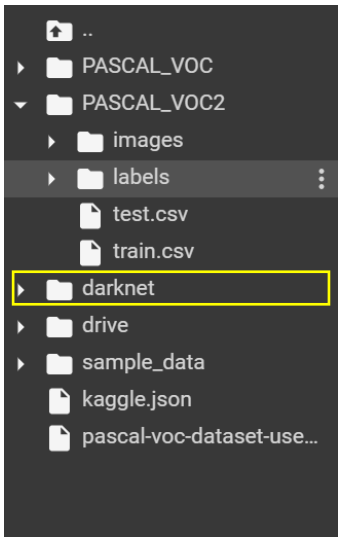
```
print(imgs_batch.shape)
print(tg_batch.shape, tg_batch.dtype)
print(tg_batch)
```

- output

```
torch.Size([4, 3, 416, 416])
torch.Size([4, 6]) torch.float32
tensor([[0.0000, 0.3510, 0.3700, 0.1060, 0.5090, 1.0000],
        [1.0000, 0.3510, 0.3700, 0.1060, 0.5090, 1.0000],
        [2.0000, 0.3510, 0.3700, 0.1060, 0.5090, 1.0000],
        [3.0000, 0.3510, 0.3700, 0.1060, 0.5090, 1.0000]])
```

■ 다크넷 cfg

```
!git clone https://github.com/pjreddie/darknet.git
path2config = '/content/darknet/cfg/yolov3-voc.cfg'
```



■ config 파일 분석 함수 정의

```
def parse_model_config(path2file):
    # cfg 파일 열기
    cfg_file = open(path2file, 'r')
    # 문자열 데이터 읽어오기
    lines = cfg_file.read().split('\n') #['[net]', '# Testing', '# batch=1', '....'
 ]

    # 데이터 전처리
    lines = [x for x in lines if x and not x.startswith('#')] # ['[net]', 'batch=64
', '...']
    lines = [x.rstrip().lstrip() for x in lines]

    blocks_list = []
    for line in lines:
        if line.startswith('['): # [net]
```

```
            blocks_list.append({}) # {}
            blocks_list[-1]['type'] = line[1:-1].rstrip() # [{'type': 'net'}]
        else:
            key, value = line.split('=') # batch=64 -> batch, 64
            value = value.strip() # 공백 제거
            blocks_list[-1][key.rstrip()] = value.strip() # 'batch':'64'

    return blocks_list
```

■ blocks_list 확인

```
blocks_list = parse_model_config(path2config)
print(blocks_list)
```

```
[{'type': 'net', 'batch': '1', 'subdivisions': '1', 'width': '416', 'height':
'416', 'channels': '3', 'momentum': '0.9', 'decay': '0.0005', 'angle': '0',
'saturation': '1.5', 'exposure': '1.5', 'hue': '.1', 'learning_rate': '0.001',
                                        ...
                                    (후략)
```

■ EmptyLayer 정의

```
class EmptyLayer(nn.Module):
    def __init__(self):
        super().__init__()
```

■ YOLO Layer 정의

```
class YOLOLayer(nn.Module):
    def __init__(self, anchors, num_classes, img_dim=416):
        super().__init__()
        self.anchors = anchors # three anchor per YOLO layer
        self.num_anchors = len(anchors) # 3
        self.num_classes = num_classes
        self.img_dim = img_dim
        self.grid_size = 0

    def forward(self, x):
        # x: batch_size, channels, H, W
        batch_size = x.size(0)
        grid_size = x.size(2) # S = 13 or 26 or 52
        device = x.device

        prediction = x.view(batch_size, self.num_anchors, self.num_classes + 5,
                            grid_size, grid_size) # shape = (batch, 3, 25, S, S)

        # (batch, 3, 25, S, S) -> (batch, 3, S, S, 25)
        prediction = prediction.permute(0, 1, 3, 4, 2)
        prediction = prediction.contiguous()
```

```
        obj_score = torch.sigmoid(prediction[..., 4]) # 클래스
        pred_cls = torch.sigmoid(prediction[..., 5:]) # 바운딩 박스 좌표

        if grid_size != self.grid_size:
            # grid_size 갱신, cell index 생성, anchor 정규화
            self.compute_grid_offsets(grid_size, cuda=x.is_cuda)

        # bounding box prediction
        pred_boxes = self.transform_outputs(prediction)
        output = torch.cat((pred_boxes.view(batch_size, -1, 4),
                            obj_score.view(batch_size, -1, 1),
                            pred_cls.view(batch_size, -1, self.num_classes)), -1)
        return output

    def compute_grid_offsets(self, grid_size, cuda=True):
        self.grid_size = grid_size # ex) 13, 26, 52
        self.stride = self.img_dim / self.grid_size # ex) 32, 16, 8

        # cell index 생성
        # 1, 1, S, 1
        self.grid_x = torch.arange(grid_size, device=device).repeat(1, 1, grid_size
, 1).type(torch.float32)
        # 1, 1, 1, S
        self.grid_y = torch.arange(grid_size, device=device).repeat(1, 1, grid_size
, 1).transpose(3, 2).type(torch.float32)

        # anchors를 feature map 크기로 정규화, [0~1] 범위
        scaled_anchors = [(a_w / self.stride, a_h / self.stride) for a_w, a_h in se
lf.anchors]
        # tensor로 변환
        self.scaled_anchors = torch.tensor(scaled_anchors, device=device)

        # shape=(3,2) -> (1,1,3,1)
        self.anchor_w = self.scaled_anchors[:, 0:1].view((1, self.num_anchors, 1, 1
))
        # shape=(3,2) -> (1,1,3,1)
        self.anchor_h = self.scaled_anchors[:, 1:2].view((1, self.num_anchors, 1, 1
))

    def transform_outputs(self, prediction):
        # pridiction (batch, 3, S, S, 25)
        device = prediction.device
        x = torch.sigmoid(prediction[..., 0]) # sigmoid(box x)
        y = torch.sigmoid(prediction[..., 1]) # sigmoid(box y)
        w = prediction[..., 2] # 예측한 바운딩 박스 너비
        h = prediction[..., 3] # 예측한 바운딩 박스 높이
```

```
        pred_boxes = torch.zeros_like(prediction[..., :4]).to(device)
        pred_boxes[..., 0] = x.data + self.grid_x # sigmoid(box x) + cell x 좌표
        pred_boxes[..., 1] = y.data + self.grid_y # sigmoid(box y) + cell y 좌표
        pred_boxes[..., 2] = torch.exp(w.data) * self.anchor_w
        pred_boxes[..., 3] = torch.exp(h.data) * self.anchor_h

        return pred_boxes * self.stride
```

■ 레이어 생성 함수 정의

```
def create_layers(blocks_list):
    hyperparams = blocks_list[0]
    channels_list = [int(hyperparams['channels'])]
    module_list = nn.ModuleList()

    for layer_ind, layer_dict in enumerate(blocks_list[1:]):
        modules = nn.Sequential()

        if layer_dict['type'] == 'convolutional':
            filters = int(layer_dict['filters'])
            kernel_size = int(layer_dict['size'])
            pad = (kernel_size - 1) // 2
            bn = layer_dict.get('batch_normalize', 0)

            conv2d = nn.Conv2d(in_channels=channels_list[-
1], out_channels=filters, kernel_size=kernel_size,
                               stride=int(layer_dict['stride']), padding=pad, bias=
not bn)
            modules.add_module('conv_{0}'.format(layer_ind), conv2d)

            if bn:
                bn_layer = nn.BatchNorm2d(filters, momentum=0.9, eps=1e-5)
                modules.add_module('batch_norm_{0}'.format(layer_ind), bn_layer)

            if layer_dict['activation'] == 'leaky':
                activn = nn.LeakyReLU(0.1)
                modules.add_module('leky_{0}'.format(layer_ind), activn)

        elif layer_dict["type"] == "upsample":
            stride = int(layer_dict["stride"])
            upsample = nn.Upsample(scale_factor = stride)
            modules.add_module("upsample_{}".format(layer_ind), upsample)

        elif layer_dict["type"] == "shortcut":
            backwards=int(layer_dict["from"])
            filters = channels_list[1:][backwards]
            modules.add_module("shortcut_{}".format(layer_ind), EmptyLayer())
```

```
        elif layer_dict["type"] == "route":
            layers = [int(x) for x in layer_dict["layers"].split(",")]
            filters = sum([channels_list[1:][l] for l in layers])
            modules.add_module("route_{}".format(layer_ind), EmptyLayer())

        elif layer_dict["type"] == "yolo":
            anchors = [int(a) for a in layer_dict["anchors"].split(",")]
            anchors = [(anchors[i], anchors[i + 1]) for i in range(0, len(anchors),
 2)]
            mask = [int(m) for m in layer_dict["mask"].split(",")]

            anchors = [anchors[i] for i in mask] # 3 anchors

            num_classes = int(layer_dict["classes"]) # 20
            img_size = int(hyperparams["height"]) # 416

            yolo_layer = YOLOLayer(anchors, num_classes, img_size)
            modules.add_module("yolo_{}".format(layer_ind), yolo_layer)

        module_list.append(modules)
        channels_list.append(filters)

    return hyperparams, module_list
```

```
class Darknet(nn.Module):
    def __init__(self, config_path, img_size=416):
        super(Darknet, self).__init__()
        self.blocks_list = parse_model_config(config_path)
        self.hyperparams, self.module_list = create_layers(self.blocks_list)
        self.img_size = img_size

    def forward(self, x):
        img_dim = x.shape[2]
        layer_outputs, yolo_outputs = [], []

        # blocks_list: config 파일 분석한 결과
        # module_list: blocks_list로 생성한 module
        for block, module in zip(self.blocks_list[1:], self.module_list):
            if block["type"] in ["convolutional", "upsample", "maxpool"]:
                x = module(x)


            elif block["type"] == "shortcut":
                layer_ind = int(block["from"]) # -3
                x = layer_outputs[-
1] + layer_outputs[layer_ind] # shortcut connection
```

```
            #  {'type': 'yolo', 'mask': '3,4,5', 'anchors': '10,13, ...}
            elif block["type"] == "yolo":
                x= module[0](x) # get yolo layer output
                yolo_outputs.append(x)
            elif block["type"] == "route": #  {'type': 'route', 'layers': '-1, 61'}
                x = torch.cat([layer_outputs[int(l_i)]
                               for l_i in block["layers"].split(",")], 1)
            layer_outputs.append(x)
        yolo_out_cat = torch.cat(yolo_outputs, 1) # 3 개의 output 을 하나로 연결
        return yolo_out_cat, yolo_outputs
```

■ 모델 확인

```
# check model
model = Darknet(path2config).to(device)
x=torch.rand(1,3,416,416).to(device)
with torch.no_grad():
    yolo_out_cat, yolo_outputs=model.forward(x)
    print(yolo_out_cat.shape)
    print(yolo_outputs[0].shape,yolo_outputs[1].shape,yolo_outputs[2].shape)
```

- output

```
torch.Size([1, 10647, 25])
torch.Size([1, 507, 25]) torch.Size([1, 2028, 25]) torch.Size([1, 8112, 25])
```

■ summary

```
summary(model, (3, 416, 416))
```

- output

```
----------------------------------------------------------------
        Layer (type)               Output Shape         Param #
================================================================
            Conv2d-1         [-1, 32, 416, 416]             864
       BatchNorm2d-2         [-1, 32, 416, 416]              64
         LeakyReLU-3         [-1, 32, 416, 416]               0
            Conv2d-4         [-1, 64, 208, 208]          18,432
       BatchNorm2d-5         [-1, 64, 208, 208]             128
         LeakyReLU-6         [-1, 64, 208, 208]               0
            Conv2d-7         [-1, 32, 208, 208]           2,048

                                  (...중략...)

================================================================
Total params: 61,626,049
Trainable params: 61,626,049
Non-trainable params: 0
----------------------------------------------------------------
Input size (MB): 1.98
Forward/backward pass size (MB): 884.38
Params size (MB): 235.08
Estimated Total Size (MB): 1121.45
----------------------------------------------------------------
```

■ get_loss_batch

```
def get_loss_batch(output,targets, params_loss, opt=None):
    ignore_thres=params_loss["ignore_thres"]
    scaled_anchors= params_loss["scaled_anchors"] # 정규화된 anchor
    mse_loss= params_loss["mse_loss"] # nn.MSELoss
    bce_loss= params_loss["bce_loss"] # nn.BCELoss, 이진 분류에서 사용

    num_yolos=params_loss["num_yolos"] # 3
    num_anchors= params_loss["num_anchors"] # 3
    obj_scale= params_loss["obj_scale"] # 1
    noobj_scale= params_loss["noobj_scale"] # 100

    loss = 0.0

    for yolo_ind in range(num_yolos):
        yolo_out = output[yolo_ind] # yolo_out: batch, num_boxes, class+coordinates
        batch_size, num_bbxs, _ = yolo_out.shape

        # get grid size
        gz_2 = num_bbxs/num_anchors # ex) at 13x13, 507 / 3
        grid_size=int(np.sqrt(gz_2))

        yolo_out = yolo_out.view(batch_size, num_anchors, grid_size, grid_size, -1)

        pred_boxes = yolo_out[:,:,:,:,:4] # get box coordinates
        x,y,w,h = transform_bbox(pred_boxes, scaled_anchors[yolo_ind])
        pred_conf = yolo_out[:,:,:,:,4] # get confidence
        pred_cls_prob = yolo_out[:,:,:,:,5:]

        yolo_targets = get_yolo_targets({
            'pred_cls_prob':pred_cls_prob,
            'pred_boxes':pred_boxes,
            'targets':targets,
            'anchors':scaled_anchors[yolo_ind],
            'ignore_thres':ignore_thres,
        })

        obj_mask=yolo_targets["obj_mask"]
        noobj_mask=yolo_targets["noobj_mask"]
        tx=yolo_targets["tx"]
        ty=yolo_targets["ty"]
        tw=yolo_targets["tw"]
        th=yolo_targets["th"]
        tcls=yolo_targets["tcls"]
        t_conf=yolo_targets["t_conf"]

        loss_x = mse_loss(x[obj_mask], tx[obj_mask])
```

```
        loss_y = mse_loss(y[obj_mask], ty[obj_mask])
        loss_w = mse_loss(w[obj_mask], tw[obj_mask])
        loss_h = mse_loss(h[obj_mask], th[obj_mask])

        loss_conf_obj = bce_loss(pred_conf[obj_mask], t_conf[obj_mask])
        loss_conf_noobj = bce_loss(pred_conf[noobj_mask], t_conf[noobj_mask])
        loss_conf = obj_scale * loss_conf_obj + noobj_scale * loss_conf_noobj
        loss_cls = bce_loss(pred_cls_prob[obj_mask], tcls[obj_mask])
        loss += loss_x + loss_y + loss_w + loss_h + loss_conf + loss_cls

    if opt is not None:
        opt.zero_grad()
        loss.backward()
        opt.step()

    return loss.item()
```

■ transform_bbox

```
def transform_bbox(bbox, anchors):
    # bbox: predicted bbox coordinates
    # anchors: scaled anchors
    x = bbox[:,:,:,:,0]
    y = bbox[:,:,:,:,1]
    w = bbox[:,:,:,:,2]
    h = bbox[:,:,:,:,3]
    anchor_w = anchors[:,0].view((1,3,1,1))
    anchor_h = anchors[:,1].view((1,3,1,1))

    x=x-x.floor() # 전체 이미지의 x 좌표에서 셀 내의 x 좌표로 변경
    y=y-y.floor() # 전체 이미지의 y 좌표에서 셀 내의 y 좌표로 변경
    w=torch.log(w / anchor_w + 1e-16)
    h=torch.log(h / anchor_h + 1e-16)
    return x, y, w, h
```

■ get_yolo_targets

```
def get_yolo_targets(params):
    pred_boxes = params['pred_boxes']
    pred_cls_prob = params['pred_cls_prob']
    target = params['targets'] # batchsize, cls, cx, cy, w, h
    anchors = params['anchors']
    ignore_thres = params['ignore_thres']

    batch_size = pred_boxes.size(0)
    num_anchors = pred_boxes.size(1)
    grid_size = pred_boxes.size(2)
    num_cls = pred_cls_prob.size(-1)
```

```
    sizeT = batch_size, num_anchors, grid_size, grid_size
    obj_mask = torch.zeros(sizeT, device=device, dtype=torch.uint8)
    noobj_mask = torch.ones(sizeT, device=device, dtype=torch.uint8)
    tx = torch.zeros(sizeT, device=device, dtype=torch.float32)
    ty = torch.zeros(sizeT, device=device, dtype=torch.float32)
    tw = torch.zeros(sizeT, device=device, dtype=torch.float32)
    th = torch.zeros(sizeT, device=device, dtype=torch.float32)

    sizeT = batch_size, num_anchors, grid_size, grid_size, num_cls
    tcls = torch.zeros(sizeT, device=device, dtype=torch.float32)

    # target = batch, cx, cy, w, h, class
    target_bboxes = target[:, 1:5] * grid_size
    t_xy = target_bboxes[:, :2]
    t_wh = target_bboxes[:, 2:]
    t_x, t_y = t_xy.t()
    t_w, t_h = t_wh.t()

    grid_i, grid_j = t_xy.long().t()

    # anchor와 target의 iou 계산
    iou_with_anchors = [get_iou_WH(anchor, t_wh) for anchor in anchors]
    iou_with_anchors = torch.stack(iou_with_anchors)
    best_iou_wa, best_anchor_ind = iou_with_anchors.max(0) # iou가 가장 높은 anchor
추출

    batch_inds, target_labels = target[:, 0].long(), target[:, 5].long()
    obj_mask[batch_inds, best_anchor_ind, grid_j, grid_i] = 1 # iou가 가장 높은 anch
or 할당
    noobj_mask[batch_inds, best_anchor_ind, grid_j, grid_i] = 0

    # threshold 보다 높은 iou를 지닌 anchor, iou가 가장 높은 anchor만 할당하면 됨
    for ind, iou_wa in enumerate(iou_with_anchors.t()):
        noobj_mask[batch_inds[ind], iou_wa > ignore_thres, grid_j[ind], grid_i[ind]
] = 0

    # cell 내에서 x,y로 변환
    tx[batch_inds, best_anchor_ind, grid_j, grid_i] = t_x - t_x.float()
    ty[batch_inds, best_anchor_ind, grid_j, grid_i] = t_y - t_y.float()

    anchor_w = anchors[best_anchor_ind][:, 0]
    tw[batch_inds, best_anchor_ind, grid_j, grid_i] = torch.log(t_w / anchor_w + 1e
-16)

    anchor_h = anchors[best_anchor_ind][:, 1]
    th[batch_inds, best_anchor_ind, grid_j, grid_i] = torch.log(t_h / anchor_h + 1e
-16)
```

```
    tcls[batch_inds, best_anchor_ind, grid_j, grid_i, target_labels] = 1

    output = {
        'obj_mask': obj_mask,
        'noobj_mask': noobj_mask,
        'tx': tx,
        'ty': ty,
        'tw': tw,
        'th': th,
        'tcls': tcls,
        't_conf': obj_mask.float(),
    }
    return output
```

■ get_iou_WH

```
def get_iou_WH(wh1, wh2):
    wh2 = wh2.t()
    w1, h1 = wh1[0], wh1[1]
    w2, h2 = wh2[0], wh2[1]
    inter_area = torch.min(w1, w2) * torch.min(h1, h2)
    union_area = (w1 * h1 + 1e-16) + w2 * h2 - inter_area
    return inter_area / union_area
```

```
opt = optim.Adam(model.parameters(), lr=1e-3)
lr_scheduler = ReduceLROnPlateau(opt, mode='min',factor=0.5, patience=20,verbose=1)
```

■ get_lr

```
def get_lr(opt):
    for param_group in opt.param_groups:
        return param_group['lr']
```

■ loss_epoch

```
def loss_epoch(model,params_loss,dataset_dl,sanity_check=False,opt=Non ):
    running_loss=0.0
    len_data=len(dataset_dl.dataset)
    running_metrics= {}

    for xb, yb,_ in dataset_dl:
        yb=yb.to(device)
        _,output=model(xb.to(device))
        loss_b=get_loss_batch(output,yb, params_loss,opt)
        running_loss+=loss_b
        if sanity_check is True:
            break
```

```
    loss=running_loss/float(len_data)
    return loss
```

■ train_val, 파라미터 지정

```
import time
def train_val(model, params):
    num_epochs=params["num_epochs"]
    params_loss=params["params_loss"]
    opt=params["optimizer"]
    train_dl=params["train_dl"]
    val_dl=params["val_dl"]
    sanity_check=params["sanity_check"]
    lr_scheduler=params["lr_scheduler"]
    path2weights=params["path2weights"]



    loss_history={
        "train": [],
        "val": [],
    }
    best_model_wts = copy.deepcopy(model.state_dict())
    best_loss=float('inf')

    start_time = time.time()
    for epoch in range(num_epochs):
        current_lr=get_lr(opt)
        print('Epoch {}/{}, current lr={}'.format(epoch, num_epochs - 1, current_lr
))
        model.train()
        train_loss=loss_epoch(model,params_loss,train_dl,sanity_check,opt)
        loss_history["train"].append(train_loss)

        model.eval()
        with torch.no_grad():
            val_loss=loss_epoch(model,params_loss,val_dl,sanity_check)
        loss_history["val"].append(val_loss)

        if val_loss < best_loss:
            best_loss = val_loss
            best_model_wts = copy.deepcopy(model.state_dict())
            torch.save(model.state_dict(), path2weights)
            print("Copied best model weights!")
            print('Get best val loss')

        lr_scheduler.step(val_loss)
        if current_lr != get_lr(opt):
            print("Loading best model weights!")
```

```
            model.load_state_dict(best_model_wts)
        print("train loss: %.6f, val loss: %.6f, time: %.4f min" %(train_loss, val_
loss, (time.time()-start_time)/60))
        print("-"*10)
    model.load_state_dict(best_model_wts)
    return model, loss_history


path2models= "/content/models/"
if not os.path.exists(path2models):
        os.mkdir(path2models)

scaled_anchors=[model.module_list[82][0].scaled_anchors,
                model.module_list[94][0].scaled_anchors,
                model.module_list[106][0].scaled_anchors]


mse_loss = nn.MSELoss(reduction="sum")
bce_loss = nn.BCELoss(reduction="sum")
params_loss={
    "scaled_anchors" : scaled_anchors,
    "ignore_thres": 0.5,
    "mse_loss": mse_loss,
    "bce_loss": bce_loss,
    "num_yolos": 3,
    "num_anchors": 3,
    "obj_scale": 1,
    "noobj_scale": 100,
}


params_train={
    "num_epochs": 20,
    "optimizer": opt,
    "params_loss": params_loss,
    "train_dl": train_dl,
    "val_dl": val_dl,
    "sanity_check": False,
    "lr_scheduler": lr_scheduler,
    "path2weights": path2models+"weights.pt",
}
model,loss_hist=train_val(model,params_train)
```

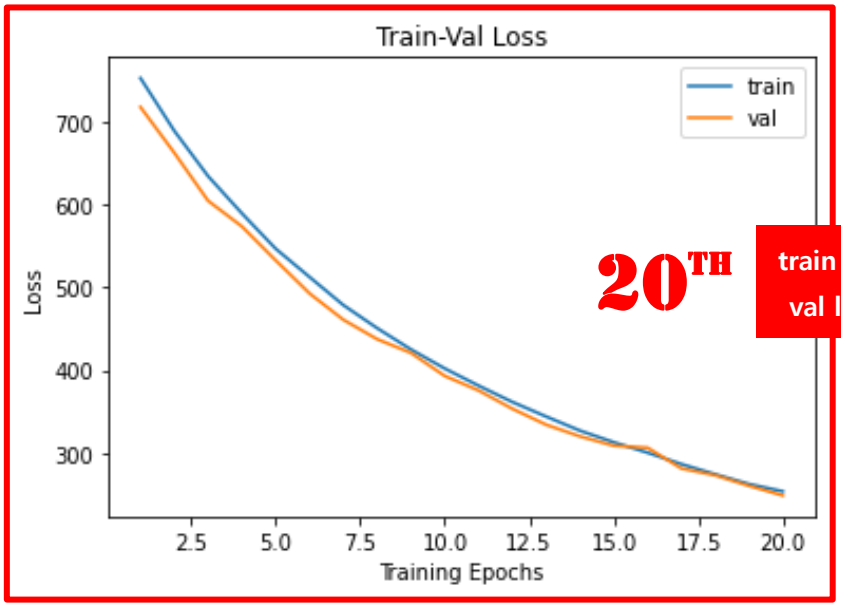
- output

```
Epoch 0/19, current lr=0.001
Copied best model weights!
Get best val loss
train loss: 752.179845, val loss: 717.240051, time: 0.4199 min
```

```
----------
                                        (...중략...)
Epoch 19/19, current lr=0.001
Copied best model weights!
Get best val loss
train loss: 253.902252, val loss: 248.937134, time: 8.3872 min
----------
```

■ Train-Val Loss Plot 출력

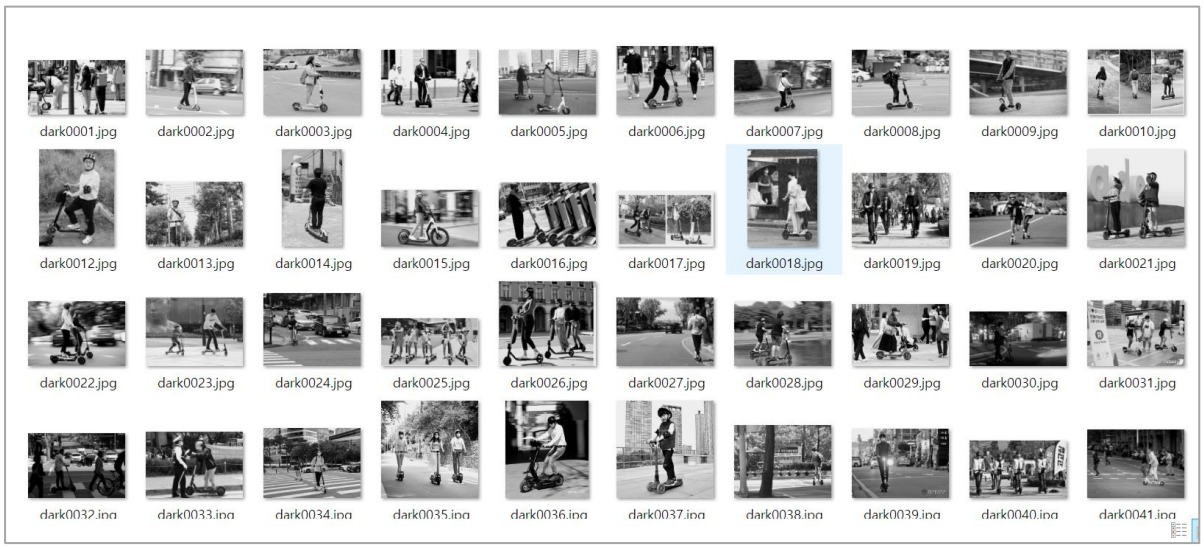
```
num_epochs = params_train['num_epochs']
plt.title('Train-Val Loss')
plt.plot(range(1, num_epochs+1), loss_hist['train'], label='train')
plt.plot(range(1, num_epochs+1), loss_hist['val'], label='val')
plt.ylabel('Loss')
plt.xlabel('Training Epochs')
plt.legend()
plt.show()
```



2ND TRIAL 2차 모델링 시도 (~11.08)

images	image 208개 (color 103 + grayscale 2[1]+103)
labels	images에 해당하는 txt 파일
train.csv	**8**:2 (166개)
test.csv	8:**2** (42개)

- grayscale image 변환



■ 트레인 파일, 라벨, 이미지 디렉터리 지정

```
train_csv_file = '/content/PASCAL_VOC3/train.csv'
label_dir = '/content/PASCAL_VOC3/labels'
img_dir = '/content/PASCAL_VOC3/images'

train_ds = VOCDataset(train_csv_file, img_dir, label_dir)

img, labels, _ = train_ds[1]
print('number of data:',len(train_ds))
print('image size:', img.shape, type(img)) # HxWxC
print('labels shape:', labels.shape, type(labels))  # x1,y1,x2,y2
print('lables \n', labels)
```

- output

```
number of data: 165
image size: (277, 500, 3) <class 'numpy.ndarray'>
labels shape: (1, 5) <class 'numpy.ndarray'>
lables
 [[0.351    0.370036 0.106    0.509025 1.      ]]
```

[1] 기존 데이터에서 11번, 55번 이미지는 흑백, 이 2개를 제외한 103개의 이미지를 변환함

■ 테스트 파일, 라벨, 이미지 디렉터리 지정

```
val_csv_file = '/content/PASCAL_VOC3/test.csv'
label_dir = '/content/PASCAL_VOC3/labels'
img_dir = '/content/PASCAL_VOC3/images'

val_ds = VOCDataset(val_csv_file, img_dir, label_dir)

img, labels, _ = val_ds[1]
print('number of data:',len(val_ds))
print('image size:', img.shape, type(img))
print('labels shape:', labels.shape, type(labels))
print('lables \n', labels)
```

- output

```
number of data: 41
image size: (277, 500, 3) <class 'numpy.ndarray'>
labels shape: (1, 5) <class 'numpy.ndarray'>
lables
 [[0.351    0.370036 0.106    0.509025 1.      ]]
```

■ 정규화 적용 샘플 이미지 확인

[왼쪽] 오른쪽보다 심한 기울기 => 바운딩 박스가 2~30%정도 벗어남 => good

[오른쪽] 바운딩 박스가 거의 벗어나지 않음 => good

■ 모델 확인

```
# check model
model = Darknet(path2config).to(device)
x=torch.rand(1,3,416,416).to(device)
with torch.no_grad():
    yolo_out_cat, yolo_outputs=model.forward(x)
    print(yolo_out_cat.shape)
```

```
    print(yolo_outputs[0].shape,yolo_outputs[1].shape,yolo_outputs[2].shape)
```

- output

```
torch.Size([1, 10647, 25])
torch.Size([1, 507, 25]) torch.Size([1, 2028, 25]) torch.Size([1, 8112, 25])
```

■ epoch 설정 : 20

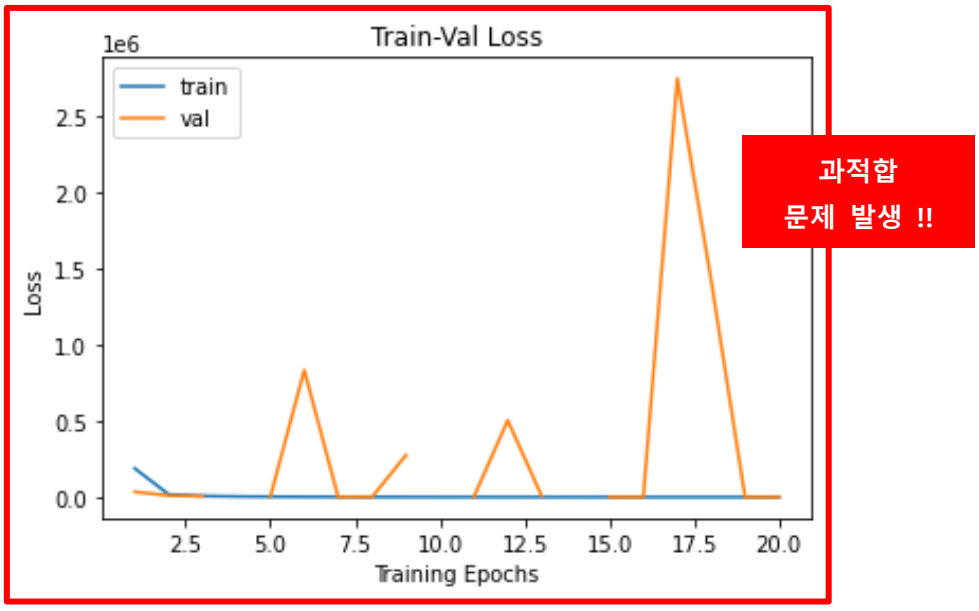
```
params_train={
    "num_epochs": 20,
    "optimizer": opt,
    "params_loss": params_loss,
    "train_dl": train_dl,
    "val_dl": val_dl,
    "sanity_check": False,
    "lr_scheduler": lr_scheduler,
    "path2weights": path2models+"weights.pt",
}
model,loss_hist=train_val(model,params_train)
```

- output

```
Epoch 0/19, current lr=0.001
Copied best model weights!
Get best val loss
train loss: 189044.370005, val loss: 35137.941406, time: 0.6726 min
----------

                                    (...중략...)

Epoch 19/19, current lr=0.001
Copied best model weights!
Get best val loss
train loss: 333.294390, val loss: 325.409363, time: 13.3261 min
----------
```

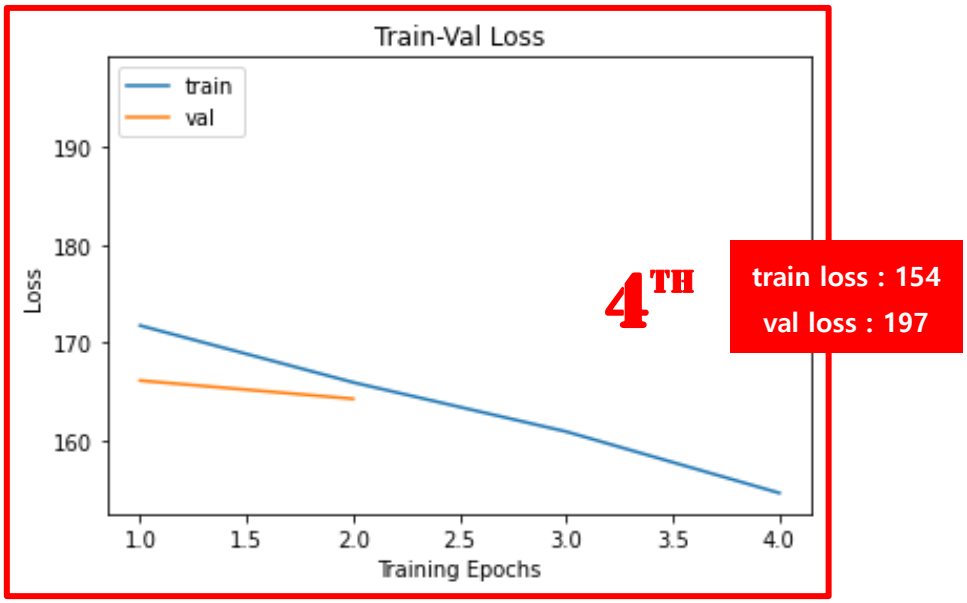


■ epoch 재설정 : 4

```
params_train={
    "num_epochs": 4,
    "optimizer": opt,
    "params_loss": params_loss,
    "train_dl": train_dl,
    "val_dl": val_dl,
    "sanity_check": False,
    "lr_scheduler": lr_scheduler,
    "path2weights": path2models+"weights.pt",
}
model,loss_hist=train_val(model,params_train)
```

- output

```
Epoch 0/3, current lr=0.001
Copied best model weights!
Get best val loss
train loss: 171.753567, val loss: 166.119156, time: 0.6741 min
----------
                                    (...중략...)
Epoch 3/3, current lr=0.001
train loss: 154.607904, val loss: 197.124678, time: 2.6665 min
----------
```



■ 모델 저장

```
model_dir = '/content/drive/MyDrive/Colab Notebooks/YOLO/'
torch.save(model, model_dir + 'yolov3-1108.pth')
torch.save(model, model_dir + 'yolov3-1108.pt')
```