Yolo v1 중간점검

이다희, 박유림

CONTENTS

구현 순서(계획)

Model

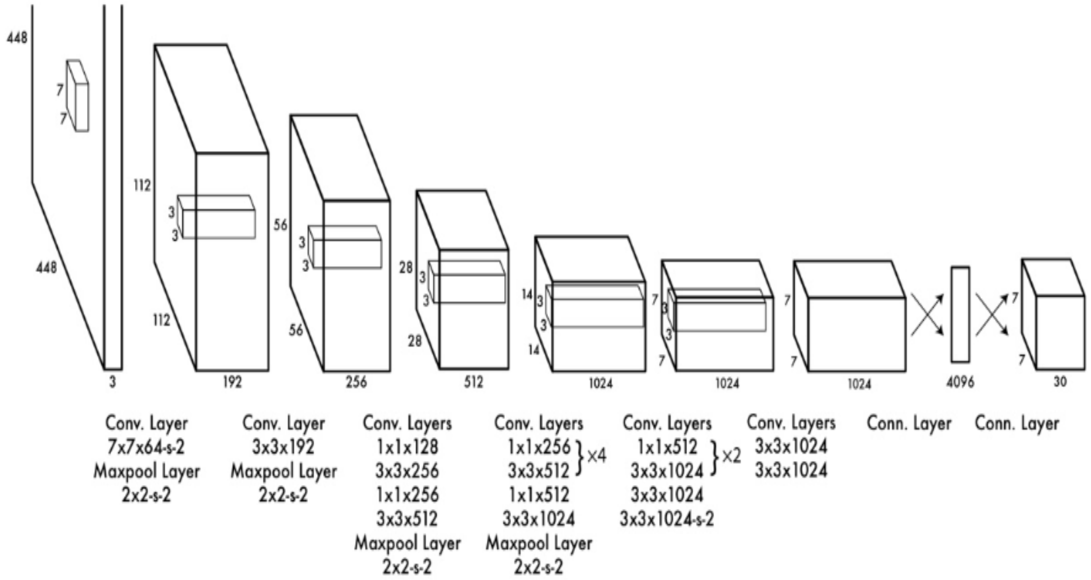
Loss Function

Utility Functions

Data Implement

Architecture model

Architecture model



[1]:
```
import torch.nn as nn
import torch
```

[2]:
```
architecture_config = [
    # Tuple: (kernel_size,num_filters,stride,padding)
    (7,64,2,3),
    "M",
    (3,192,1,1),
    "M",
    (1,128,1,0),
    (3,256,1,1),
    (1,256,1,0),
    (3,512,1,1),
    "M",
    #List: tuples and then last integer represents number of repeats
    [(1,256,1,0),(3,512,1,1),4],
    (1,512,1,0),
    (3,1024,1,1),
    "M",
    [(1,512,1,0),(3,1024,1,1),2],
    (3,1024,1,1),
    (3,1024,2,1),
    (3,1024,1,1),
    (3,1024,1,1),
    ]
#M은 maxpooling
```

Architecture model

```
In [3]: class CNNBlock(nn.Module):
            def __init__(self, in_channels, out_channels, **kwargs):
                super(CNNBlock, self).__init__()
                self.conv = nn.Conv2d(in_channels, out_channels, bias=False, **kwargs)
                self.batchnorm = nn.BatchNorm2d(out_channels)
                self.leakyrelu = nn.LeakyReLU(0.1)

            def forward(self, x):
                return self.leakyrelu(self.batchnorm(self.conv(x)))
```

```
In [4]: class Yolov1(nn.Module):
            def __init__(self, in_channels=3, **kwargs):
                super(Yolov1, self).__init__()
                self.architecture = architecture_config
                self.in_channels = in_channels
                self.darknet = self._create_conv_layers(self.architecture) #built from architecture
                self.fcs = self._create_fcs(**kwargs)
            def forward(self, x):
                x = self.darknet(x)
                return self.fcs(torch.flatten(x, start_dim=1)) #fully connected

            def _create_conv_layers(self, architecture):
                layers = []
                in_channels = self.in_channels

                for x in architecture:
                    if type(x) == tuple:
                        layers += [
                            CNNBlock(
                                in_channels, x[1], kernel_size=x[0], stride=x[2], padding=x[3],
```

```
    def _create_fcs(self, split_size, num_boxes, num_classes):
        S, B, C = split_size, num_boxes, num_classes

        # In original paper this should be
        # nn.Linear(1024*S*S, 4096),
        # nn.LeakyReLU(0.1),
        # nn.Linear(4096, S*S*(B*5+C))

        return nn.Sequential(
            nn.Flatten(),
            nn.Linear(1024 * S * S, 496),#Original paper this should be 4096
            nn.Dropout(0.0),
            nn.LeakyReLU(0.1),
            nn.Linear(496, S * S * (C + B * 5)), #(S,S,30) where C+B*5 = 30
        )
```

```
def test(S=7,B=2,C=20):
  model = Yolov1(split_size=S,num_boxes=B,num_classes=C)
  x = torch.randn((2,3,448,3))
  print(model(x).shape)
```

Loss function and Implement

Loss function and Implementation

```python
class YoloLoss(nn.Module):
    """
    Calculate the loss for yolo (v1) model
    """

    def __init__(self, S=7, B=2, C=20):
        super(YoloLoss, self).__init__()
        self.mse = nn.MSELoss(reduction="sum") #don't actually average
        """
        S is split size of image (in paper 7),
        B is number of boxes (in paper 2),
        C is number of classes (in paper and VOC dataset is 20),
        """
        self.S = S
        self.B = B
        self.C = C

        # These are from Yolo paper, signifying how much we should
        # pay loss for no object (noobj) and the box coordinates (coord)
        self.lambda_noobj = 0.5
        self.lambda_coord = 5

    def forward(self, predictions, target):
        # predictions are shaped (BATCH_SIZE, S*S(C+B*5) when inputted
        predictions = predictions.reshape(-1, self.S, self.S, self.C + self.B * 5) #S*S*30이 되도록 재구성 해야함

        # Calculate IoU for the two predicted bounding boxes with target bbox
        #0-19 for class probabilities / 20 class score/  21-25 4 bbox values
        iou_b1 = intersection_over_union(predictions[..., 21:25], target[..., 21:25]) #intersection_over_union을 IoU로
        iou_b2 = intersection_over_union(predictions[..., 26:30], target[..., 21:25])
        ious = torch.cat([iou_b1.unsqueeze(0), iou_b2.unsqueeze(0)], dim=0) #max iou gonna return to arg max
```

```python
#IOU코드
def IoU(box1, box2):
    # box = (x1, y1, x2, y2)
    box1_area = (box1[2] - box1[0] + 1) * (box1[3] - box1[1] + 1)
    box2_area = (box2[2] - box2[0] + 1) * (box2[3] - box2[1] + 1)

    # obtain x1, y1, x2, y2 of the intersection
    x1 = max(box1[0], box2[0])
    y1 = max(box1[1], box2[1])
    x2 = min(box1[2], box2[2])
    y2 = min(box1[3], box2[3])

    # compute the width and height of the intersection
    w = max(0, x2 - x1 + 1)
    h = max(0, y2 - y1 + 1)

    inter = w * h
    iou = inter / (box1_area + box2_area - inter)
    return iou
```

IOU = Intersection / A+B - Intersection

Loss function and Implementation

```
def forward(self, predictions, target):
    # predictions are shaped (BATCH_SIZE, S*S(C+B*5) when inputted
    predictions = predictions.reshape(-1, self.S, self.S, self.C + self.B * 5) #S*S*30이 되도록 재구성 해야함

    # Calculate IoU for the two predicted bounding boxes with target bbox
    #0-19 for class probabilities / 20 class score/  21-25 4 bbox values
    iou_b1 = IoU(predictions[..., 21:25], target[..., 21:25]) #intersection_over_union을 IoU로
    iou_b2 = IoU(predictions[..., 26:30], target[..., 21:25])
    ious = torch.cat([iou_b1.unsqueeze(0), iou_b2.unsqueeze(0)], dim=0) #max iou gonna return to arg max

    # Take the box with highest IoU out of the two prediction
    # Note that bestbox will be indices of 0, 1 for which bbox was best
    iou_maxes, bestbox = torch.max(ious, dim=0)
    exists_box = target[..., 20].unsqueeze(3)  # in paper this is identity of Iobj_i
```

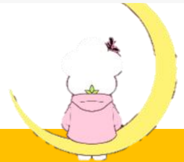
IOU = Intersection / A+B - Intersection

Loss function and Implementation

```
iou_maxes, bestbox = torch.max(ious, dim=0)
exists_box = target[..., 20].unsqueeze(3)  # in paper this is identity of Iobj_i

# ======================= #
#   FOR BOX COORDINATES    #
# ======================= #
# It means the midpoint and the width and the hight , which class the object belongs to

# Set boxes with no object in them to 0. We only take out one of the two
# predictions, which is the one with highest Iou calculated previously.
box_predictions = exists_box * ( #exist box compute the loss, object가 진짜 있을때,
    (
        bestbox * predictions[..., 26:30]
        + (1 - bestbox) * predictions[..., 21:25] #if first bbx is correct, bestbox is gonna be zer
    )
)

box_targets = exists_box * target[..., 21:25]

# Take sqrt of width, height of boxes to ensure that
box_predictions[..., 2:4] = torch.sign(box_predictions[..., 2:4]) * torch.sqrt(
    torch.abs(box_predictions[..., 2:4] + 1e-6)
```

$$\lambda_{\textbf{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left[(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2 \right]$$

$$+ \lambda_{\textbf{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left[\left(\sqrt{w_i} - \sqrt{\hat{w}_i} \right)^2 + \left(\sqrt{h_i} - \sqrt{\hat{h}_i} \right)^2 \right]$$



Loss function and Implementation

```
box_targets = exists_box * target[..., 21:25]

# Take sqrt of width, height of boxes to ensure that
box_predictions[..., 2:4] = torch.sign(box_predictions[..., 2:4]) * torch.sqrt(
    torch.abs(box_predictions[..., 2:4] + 1e-6)
)
box_targets[..., 2:4] = torch.sqrt(box_targets[..., 2:4])

box_loss = self.mse(
    torch.flatten(box_predictions, end_dim=-2), #flatten everything
    torch.flatten(box_targets, end_dim=-2),
)
```

Loss function and Implementation

```
# ==================== #
#   FOR OBJECT LOSS    #
# ==================== #

# pred_box is the confidence score for the bbox with highest IoU
pred_box = (
    bestbox * predictions[..., 25:26] + (1 - bestbox) * predictions[..., 20:21]
)



object_loss = self.mse(
    torch.flatten(exists_box * pred_box),
    torch.flatten(exists_box * target[..., 20:21]),
)
```

$$+\sum_{i=0}^{S^2}\sum_{j=0}^{B}\mathbb{1}_{ij}^{\text{obj}}\left(\underbrace{\boxed{C_i}}_{=1}-\hat{C}_i\right)^2$$

Loss function and Implementation

```
# ======================= #
#   FOR NO OBJECT LOSS    #
# ======================= #

#max_no_obj = torch.max(predictions[..., 20:21], predictions[..., 25:26])
#no_object_loss = self.mse(
#    torch.flatten((1 - exists_box) * max_no_obj, start_dim=1),
#    torch.flatten((1 - exists_box) * target[..., 20:21], start_dim=1),
#)

no_object_loss = self.mse(
    torch.flatten((1 - exists_box) * predictions[..., 20:21], start_dim=1),
    torch.flatten((1 - exists_box) * target[..., 20:21], start_dim=1),
)

no_object_loss += self.mse(
    torch.flatten((1 - exists_box) * predictions[..., 25:26], start_dim=1),
    torch.flatten((1 - exists_box) * target[..., 20:21], start_dim=1)
)
```

$$+ \lambda_{\text{noobj}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{noobj}} \left(C_i - \hat{C}_i \right)^2$$

Loss function and Implementation

```
# ================== #
#   FOR CLASS LOSS   #
# ================== #

#(N,S,S,20)-> (N*S*S,20)
class_loss = self.mse(
    torch.flatten(exists_box * predictions[..., :20], end_dim=-2,),
    torch.flatten(exists_box * target[..., :20], end_dim=-2,),
)

loss = (
    self.lambda_coord * box_loss  # first two rows in paper
    + object_loss  # third row in paper
    + self.lambda_noobj * no_object_loss  # forth row
    + class_loss  # fifth row
)

return loss
```

$$+\sum_{i=0}^{S^2} \mathbb{1}_i^{\text{obj}} \sum_{c \in \text{classes}} \left(p_i(c) - \hat{p}_i(c)\right)^2$$

문제점

문제점

Train 및 성능 평가시 구축 환경 문제

-> 주피터노트북에 했을 때, 그래픽 카드 문제로 GPU 사용을 위한 NVIDIA 드라이버 설치 불가

```
~₩anaconda3₩lib₩site-packages₩torch₩cuda₩__init__.py in _lazy_init()
    227         if 'CUDA_MODULE_LOADING' not in os.environ:
    228             os.environ['CUDA_MODULE_LOADING'] = 'LAZY'
--> 229         torch._C._cuda_init()
    230         # Some of the queued calls may reentrantly call _lazy_init();
    231         # we need to just return without initializing in that case.

RuntimeError: Found no NVIDIA driver on your system. Please check that you have an NVIDIA GPU and installed a driver from http://www.nvidia.com/Download/index.aspx
```

추후계획

추후 계획

Darknet 이용하여 github와 연결하여 모델링

OPENCV와 GPU를 사용할 수 있도록 Makefile을 변경 및 실행

```
%cd /content/darknet

!sed -i 's/OPENCV=0/OPENCV=1/' Makefile
!sed -i 's/GPU=0/GPU=1/' Makefile
!sed -i 's/CUDNN=0/CUDNN=1/' Makefile
!sed -i 's/CUDNN_HALF=0/CUDNN_HALF=1/' Makefile
!sed -i 's/LIBSO=0/LIBSO=1' Makefile

!make
!chmod +x ./darknet
```

추후 계획

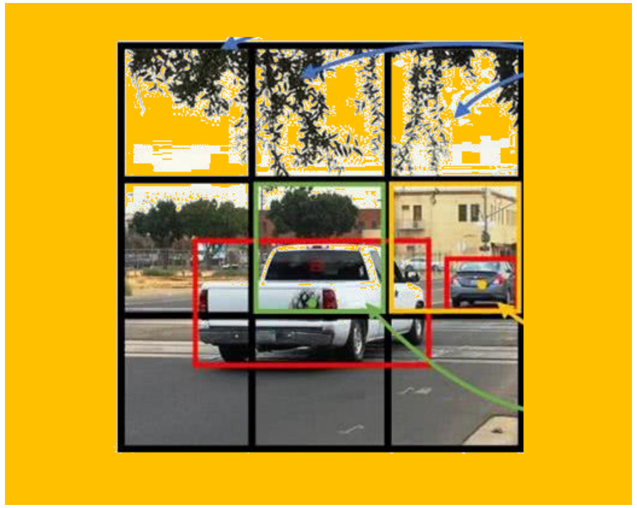
Train 및 성능 평가시 구축 환경 문제

Colab의 가상 GPU환경과 가상 nvidia 드라이버 이용

```
!nvidia-smi
#노트북 그래픽 카드 문제로 인해 코랩으로 진행 - intel그래픽카드로는 nvidia 드라이버 설치 불가
```

```
Mon Jan  9 10:23:09 2023
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 460.32.03    Driver Version: 460.32.03    CUDA Version: 11.2     |
|-------------------------------+----------------------+----------------------+
| GPU  Name        Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute M. |
|                               |                      |               MIG M. |
|===============================+======================+======================|
|   0  Tesla T4            Off  | 00000000:00:04.0 Off |                    0 |
| N/A   47C    P0    27W /  70W |      0MiB / 15109MiB |      0%      Default |
|                               |                      |                  N/A |
+-------------------------------+----------------------+----------------------+

+-----------------------------------------------------------------------------+
| Processes:                                                                  |
|  GPU   GI   CI        PID   Type   Process name                  GPU Memory |
|        ID   ID                                                   Usage      |
|=============================================================================|
|  No running processes found                                                 |
+-----------------------------------------------------------------------------+
```

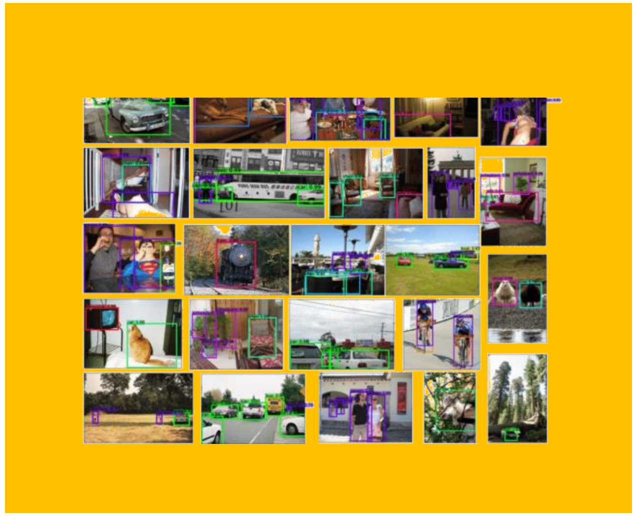
앞으로의 계획



$$mAP = \frac{1}{n}\sum_{k=1}^{k=n} AP_k$$

$$AP_k = \text{the AP of class } k$$

$$n = \text{the number of classes}$$



- 높은 확률을 나타내는 bounding box를 찾기 위해 “non-max suppression”을 구현하고, 확률을 구해볼 예정

- 정밀도 mAP 로 구현한 YOLO 모델의 객관적인 정밀도를 검증할 계획

- 실생활에서 볼 수 있는 이미지 데이터를 적용하려 다운 받았으나, 논문에 나온 PASCAL 데이터를 먼저 가지고 YOLO 논문을 구현해볼 예정

THANK YOU