CS 4375 Coding Assignment 2

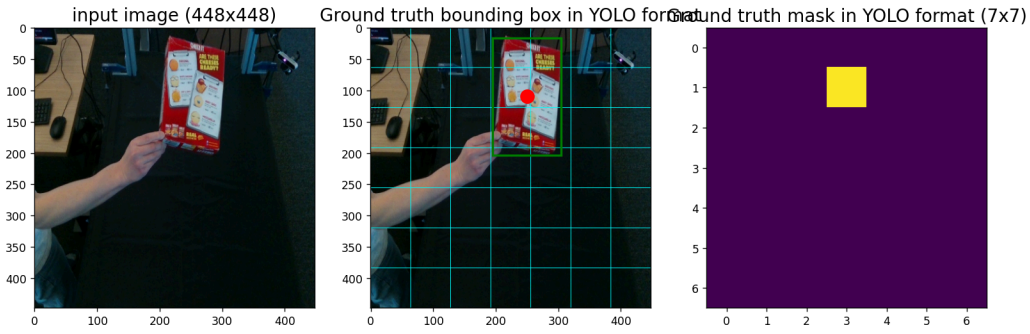
https://github.com/Eukio/CS4375HW2
Eucharist Tan
ETT220002

INTRODUCTION

My assignment seeks to classify cracker boxes using a YOLO architecture. Given an image dataset and corresponding txt file with the coordinates of the bounding box, my model is trained to identify the location of these boxes. From the results of training the model, I will observe the loss and precision vs recall.

IMPLEMENTATION

In a YOLO architecture, the input image is divided into grid boxes to create bounding boxes within a single pass. In predicting the bounding boxes, the model is to generate a confidence score that would later be used to help improve the trained model itself. As my first task, I am to modify the yolo/data.py script to create a dataloader of the box and visualize it with the input image, ground truth bounding box, and ground truth mask in YOLO format.

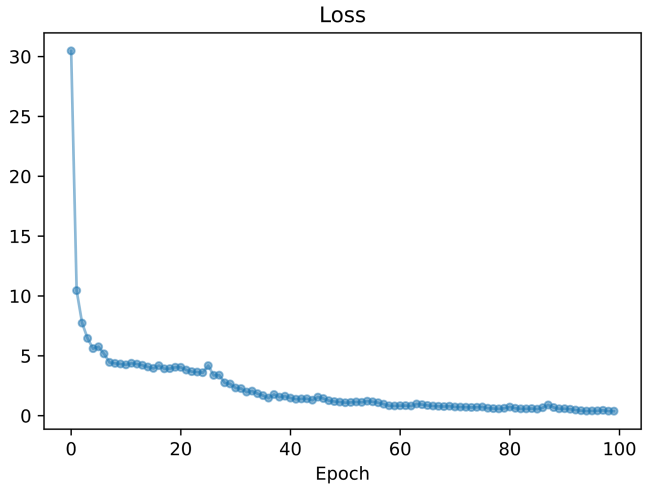


Within the yolo/model.py file, I had to define the YOLO model structure. The model accesses an image size of 448x448 and goes through 9 groupings of convolution layers with the use of ReLU as the activation function to increase the model's complexity and max pooling for feature extraction. After these layers, there is a flattening layer, 2 fully connected layers, and a sigmoid output to help compute a probability score.

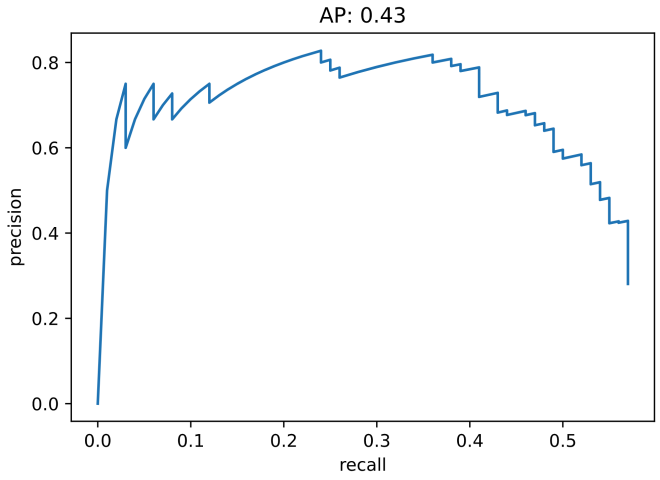
```
(.venv) PS C:\Users\Euki1\Desktop\cs4375-coding-assignment-2\yolo> python model.py
YOLO(
  (network): Sequential(
    (conv1): Conv2d(3, 16, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (relu1): ReLU()
    (pool1): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
    (conv2): Conv2d(16, 32, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (relu2): ReLU()
    (pool2): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
    (conv3): Conv2d(32, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (relu3): ReLU()
    (pool3): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
    (conv4): Conv2d(64, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (relu4): ReLU()
    (pool4): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
    (conv5): Conv2d(128, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (relu5): ReLU()
    (pool5): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
    (conv6): Conv2d(256, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (relu6): ReLU()
    (pool6): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
    (conv7): Conv2d(512, 1024, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (relu7): ReLU()
    (conv8): Conv2d(1024, 1024, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (relu8): ReLU()
    (conv9): Conv2d(1024, 1024, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (relu9): ReLU()
    (flatten): Flatten(start_dim=1, end_dim=-1)
    (fc1): Linear(in_features=50176, out_features=256, bias=True)
    (fc1_relu): ReLU()
    (fc2): Linear(in_features=256, out_features=256, bias=True)
    (fc2_relu): ReLU()
    (fc_output): Linear(in_features=256, out_features=539, bias=True)
    (sigmoid): Sigmoid()
  )
)
input image: torch.Size([1, 3, 448, 448])
network output: torch.Size([1, 11, 7, 7]) torch.Size([1, 11, 7, 7])
(.venv) PS C:\Users\Euki1\Desktop\cs4375-coding-assignment-2\yolo>
```

ANALYSIS AND RESULTS

To train the network, I implemented a loss function within the yolo/loss.py, which sums the loss of the x-coordinate, y-coordinate, width, height, objectness, non object loss, and class probability loss. The result of the data produced a graph that showed the decreasing loss over 100 epochs with batch size 4 and learning rate as 1e-4.



When I ran yolo/test.py, the following produced a precision recall curve with an average precision score AP = 0.4336, revealing that there is evidence that my model is able to learn meaningful information in identifying the cracker boxes.



CONCLUSION

Using the YOLO architecture, I was able to detect cracker boxes through creating grid/bounding boxes, building the architecture of the model, computing the loss, and training the model. The decreasing loss output and precision recall curve depicts the improvement of the model during training.