CS 182 Spring 2023 — Deep Neural Networks, Anant Sahai

Final Project: YOLO

Nishant Bhakar, Eric Li, Vikas Ummadisetty, Jonathan Yue

1 Homework Assignment without Solutions

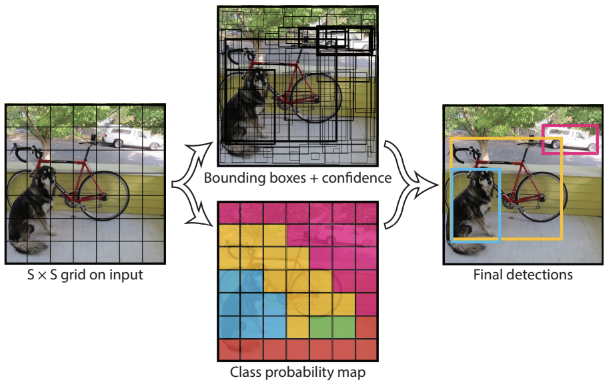
In class, you have learnt about how convolutional neural networks (CNNs) are commonly used to analyze visual imagery. CNNs are known to be shift-invariance based on the weight-sharing architecture of convolutional kernels which slide along input features and provide translational equivariant responses. In previous assignments, we have used CNNs for **object classification** to identify the class of one object in a simple image. In this problem, we will explore how CNNs can be used for **object detection** in the case where multiple relevant objects are present in a single image.

Prior work on object detection repurposes classifiers on sliding windows to perform detection, which is slow and expensive for real-time inference demanding detectors for 45 frames per second. The paper "You Only Look Once: Unified, Real-Time Object Detection" explores an extremely fast method which elegantly frames object detection as a regression problem.

In this problem, you will follow the YOLO.ipynb notebook to understand YOLO.

(a) You have trained a CNN image classifier $f : \mathbb{R}^{28\times 28\times 3} \to \mathbb{R}^n$ which maps a $28 \times 28 \times 3$ image to n conditional class probabilities. Suppose you would like to perform object detection on an image $I \in \mathbb{R}^{W\times H\times C}$ using a $28 \times 28 \times 3$ sliding window approach with stride of S and padding P on each side of I. **How many forward passes of f does this method require?**

(b) Let's examine a faster approach to object detection. In YOLO, the image is divided into a $S \times S$ grid, where a grid box is responsible for detecting an image if its center lies in the box. Each grid predicts B bounding boxes with corresponding confidence scores and C conditional class probabilities, as shown in the figure below. **How might this model's regression formulation be more efficient than a sliding window approach from part (a)?**



(c) One of the limitations of YOLO is that it imposes strong spatial constraints on bounding box prediction since each grid cell can only have one class and predict B bounding boxes. Thus, the model may perform poorly with smaller objects in groups. **Describe an example of an image YOLO may struggle with.**

(d) During training, YOLO uses the Intersection over Union (IoU) metric to benchmark the accuracy of our model. Given a ground truth bounding box and a predicted bounding box, we can calculate IoU by dividing their overlapping area (intersection) by their total area (union). Thus, higher IoU implies better performance. During test-time inference (no ground truth available), we may run into an issue where the model predicts multiple bounding boxes for a single object. **Considering that each bounding box has its own confidence score, propose a scheme that uses IoU to select the best bounding box.**

(e) Refer to the notebook linked above for remaining parts to interact with YOLO. For the Non maximal supression problem, refer to this article. The article contains a psuedocode section which will be helpful.

2 Homework Assignment with Solutions

In class, you have learnt about how convolutional neural networks (CNNs) are commonly used to analyze visual imagery. CNNs are known to be shift-invariance based on the weight-sharing architecture of convolutional kernels which slide along input features and provide translational equivariant responses. In previous assignments, we have used CNNs for **object classification** to identify the class of one object in a simple image. In this problem, we will explore how CNNs can be used for **object detection** in the case where multiple relevant objects are present in a single image.

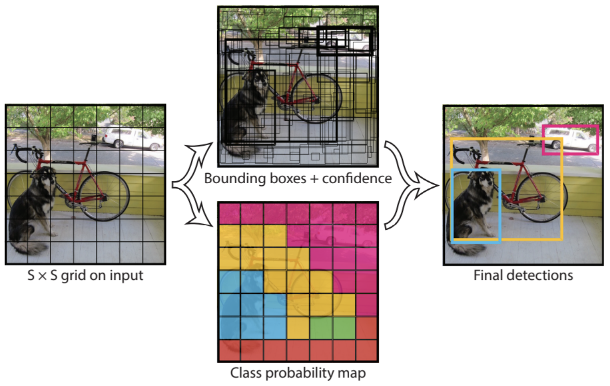
Prior work on object detection repurposes classifiers on sliding windows to perform detection, which is slow and expensive for real-time inference demanding detectors for 45 frames per second. The paper "You Only Look Once: Unified, Real-Time Object Detection" explores an extremely fast method which elegantly frames object detection as a regression problem.

In this problem, you will follow the YOLO.ipynb notebook to understand YOLO.

(a) You have trained a CNN image classifier $f : \mathbb{R}^{28\times28\times3} \to \mathbb{R}^n$ which maps a $28\times28\times3$ image to n conditional class probabilities. Suppose you would like to perform object detection on an image $I \in \mathbb{R}^{W\times H\times C}$ using a $28\times28\times3$ sliding window approach with stride of S and padding P on each side of I. **How many forward passes of f does this method require?**

Solution: $\left(\lfloor\frac{W-28+2P}{S}\rfloor+1\right)\left(\lfloor\frac{H-28+2P}{S}\rfloor+1\right)$

(b) Let's examine a faster approach to object detection. In YOLO, the image is divided into a $S\times S$ grid, where a grid box is responsible for detecting an image if its center lies in the box. Each grid predicts B bounding boxes with corresponding confidence scores and C conditional class probabilities, as shown in the figure below. **How might this model's regression formulation be more efficient than a sliding window approach from part (a)?**



Solution: The YOLO regression formulation allows the whole detection pipeline to be a single network which can be optimized end-to-end directly on detection performance. As its name suggests, YOLO only looks at an image with one forward pass for inference. Meanwhile, the sliding window method looks at the image many times like a convolution operation and may redundantly operate on the same pixels multiple times.

(c) One of the limitations of YOLO is that it imposes strong spatial constraints on bounding box prediction since each grid cell can only have one class and predict B bounding boxes. Thus, the model may perform poorly with smaller objects in groups. **Describe an example of an image YOLO may struggle with.**

Solution: Any example with large number of objects in small area are valid. A flock of birds. Crowd of people. High density traffic areas.

(d) During training, YOLO uses the Intersection over Union (IoU) metric to benchmark the accuracy of our model. Given a ground truth bounding box and a predicted bounding box, we can calculate IoU by dividing their overlapping area (intersection) by their total area (union). Thus, higher IoU implies better performance. During test-time inference (no ground truth available), we may run into an issue where the model predicts multiple bounding boxes for a single object. **Considering that each bounding box has its own confidence score, propose a scheme that uses IoU to select the best bounding box.**

Solution: This is known as Non-maximum Suppression. If many predicted bounding boxes have high IoU values, select the predicted bounding box with the highest confidence and eliminate the rest.

(e) Refer to the notebook linked above for remaining parts to interact with YOLO. For the Non maximal supression problem, refer to this article. The article contains a psuedocode section which will be helpful.

3 Commentary on HW

Key Concepts of the Paper:

In this project, we designed a self-contained homework set for the paper "You Only Look Once: Unified, Real-Time Object Detection" which explores an extremely fast real-time object detection method. Before YOLO, object detectors were created by repurposing image classifiers to perform localization and detection either by sliding window or regional proposals (i.e. R-CNN). In the spirit of improving inference speed with high sensitivity, YOLO is the first of its kind to formulate object detection as a regression problem to spatially separated bounding boxes and associated class probabilities. The entire multi-object detection pipeline is a single neural network which can be optimized end-to-end directly on detection performance. Thus, YOLO aims to eliminate the need for slow complex image detection pipelines to support real-time inference with streaming video. Further, unlike sliding window or regional proposal methods, YOLO predicts using the entire image so it leverages richer contextual information for better performance.

YOLO is designed to unify two components of image detection: bounding box prediction and object class prediction. To accomplish this, YOLO divides the image into an $S \times S$ grid, where each cell predicts B bounding boxes with corresponding confidence scores and C conditional class probabilities. More specifically each bounding box prediction is a 5-tuple $(x, y, w, h, \text{confidence})$, where the $w \times h$ bounding box is centered at (x, y). During training, bounding box confidence is defined as $P(\text{Object}) \cdot \text{IOU}^{\text{truth}}_{\text{pred}}$. In the regression formulation, we want the confidence score to be zero if the object does not exist in the grid cell and IOU of the predicted and ground truth bounding boxes if the object exists in the grid cell. Each grid cell also independently predicts C class probabilities, $P(\text{Class}_i|\text{Object})$. Multiplying bounding box confidence by conditional class probability yields the class-specific confidence score for the grid cell, which encodes both the probability of that class appearing in the box: $P(\text{Class}_i|\text{Object}) \cdot P(\text{Object}) \cdot \text{IOU}^{\text{truth}}_{\text{pred}} = P(\text{Class}_i) \cdot \text{IOU}^{\text{truth}}_{\text{pred}}$.

The YOLO model consists of 24 convolutional layers pretrained on ImageNet classification task, followed by a 2 fully-connected layer classification head. During training, we optimize the following multi-part loss function.

$$\begin{aligned}
&\lambda_{\textbf{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left[(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2 \right] \\
&+ \lambda_{\textbf{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left[\left(\sqrt{w_i} - \sqrt{\hat{w}_i}\right)^2 + \left(\sqrt{h_i} - \sqrt{\hat{h}_i}\right)^2 \right] \\
&+ \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left(C_i - \hat{C}_i\right)^2 \\
&+ \lambda_{\text{noobj}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{noobj}} \left(C_i - \hat{C}_i\right)^2 \\
&+ \sum_{i=0}^{S^2} \mathbb{1}_{i}^{\text{obj}} \sum_{c \in \text{classes}} (p_i(c) - \hat{p}_i(c))^2
\end{aligned}$$

The loss captures three components: classification loss, localization loss, and confidence loss. If the object is detected in the cell, the classification loss for each cell is simply the squared error of class conditional probabilities. The localization loss measures the errors in the predicted boundary box locations and sizes. We only count the box responsible for detecting the object. If the object is in the cell, the confidence loss for each cell is simply the squared error from the ground truth IOU. If the object is not in the cell, the confidence loss is weighted by a smaller coefficient λ_{noobj} to prevent class imbalance since many cells do not end up having objects.

During inference, YOLO is extremely fast since it only requires a single network evaluation, unlike classifier-based methods. Sometimes, there may be an issue with multiple grid cells localizing a detection, which can be easily fixed using Non-Maximum Suppression (NMS), which only keeps the highest confidence bounding box when a cluster of predicted bounding box have high IOU values. Nonetheless, YOLO struggles with small objects in groups such as flocks of birds, since each grid cell can only predict B bounding boxes for one class.

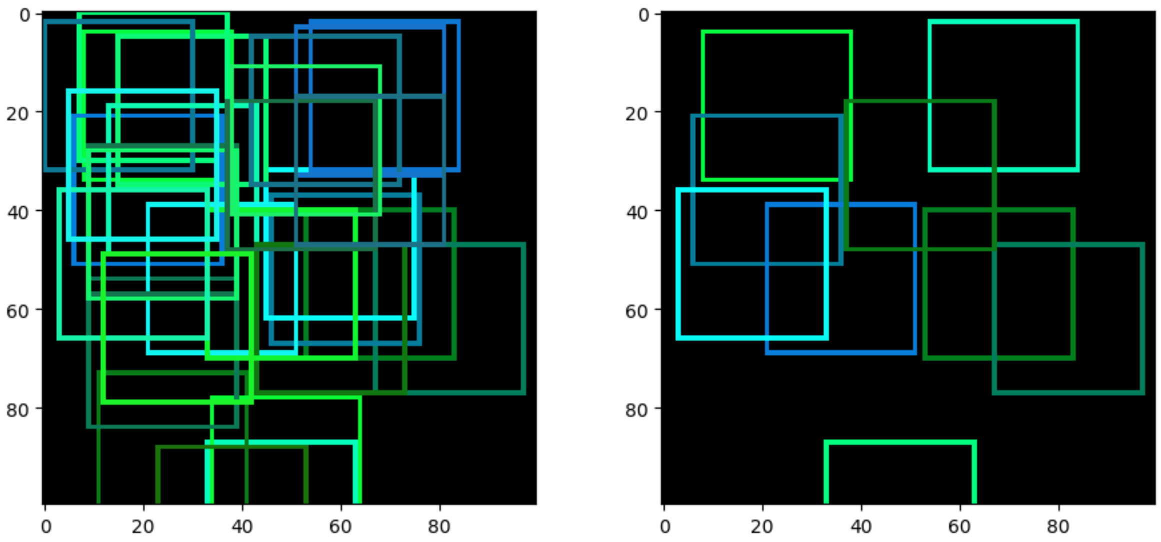
YOLO was evaluated on the Pascal VOC dataset and outperformed state-of-the-art real-time image detectors with higher mean average precision and speed. At the time, FastYOLO was the fastest image detector on record for Pascal VOC and still twice as accurate as other real-time detectors.

How the Homework Set Engages with Key Concepts:

Like other well-written homeworks in CS182, we first started this YOLO homework problem with a description of a familiar topic students have learned about in class. Since the topic is the YOLO object detector, we first provided a review summary of CNN properties (i.e. weight sharing, translational equivariance) before delving into the field of object detection. Then, we provide the paper link to YOLO and introduce its motivations for efficiency, speed, and high sensitivity.

We start the homework question with a few warm-up analytical questions to review convolutions and get the student thinking about the principles of efficient object detection. The first subpart is a warmup that reviews convolution dimensions while also revealing mathematically the inefficiency of sliding-window classifier-based methods for object detectors. The second subpart explains the regression formulation for YOLO, one of the key learning goals of this homework, and asks the student to reason about why this formulation may be better than classifier-based methods, building off the answer from the previous part. In the third subpart, we get the student to start speculating on the limitations of the YOLO model by asking them to generate an example of a scenario where YOLO fails. In the fourth analytical subpart, we ask the student to propose a post-processing optimization (Non-Maximum Suppression) using the principles of IOU, which are used in training the YOLO. After being exposed to this new topic with considerations of the inefficiencies of older methods, the motivations behinds its design, its limitations, and its inner implementation, the student now moves on to interact with the Colab notebook.

In the Colab notebook, we ask the user to implement a few basic warmup coding exercises including non-maximum suppression (NMS). Figure 1 illustrates the sanity check we expect the user to pass. We guide the user through the implementation of YOLO, exposing them to the mechanics of the bounding box predictions and class conditional probability predictions. Lastly, we have the user test the multi-object detector on test images from a synthetic dataset and data of



(a) Before NMS (b) After NMS

Figure 1: Non-maximum suppression sanity check

their own.