ECE 565 Computer Vision and Image Processing

Final Project: Modifying YOLOv2

Vincent Trosky

Instructor: Prof. Joohee Kim

Introduction

"You Only Look Once version 2" (YOLOv2) is an object detection algorithm capable of completing real-time object detection tasks. Developed by Joseph Redmon and Ali Farhadi, it builds upon the original YOLO algorithm, addressing its limitations and improving both accuracy and speed. YOLOv2 is a deep learning-based algorithm that approaches object detection as a single regression problem. After dividing an input image into a grid, the algorithm uses anchor bounding boxes around objects and produces class probabilities for each grid cell. The algorithm then uses this data and performs feature extraction based on prior training over a set of image classes using a deep convolutional neural network (CNN). In this report, multiple modifications were attempted, none of which outperformed the baseline model by 50 epochs.

1. YOLOv2 Baseline Method

1.1 Architecture

The CNN architecture for the YOLOv2 baseline deep learning framework can be seen in Fig. 1 below.

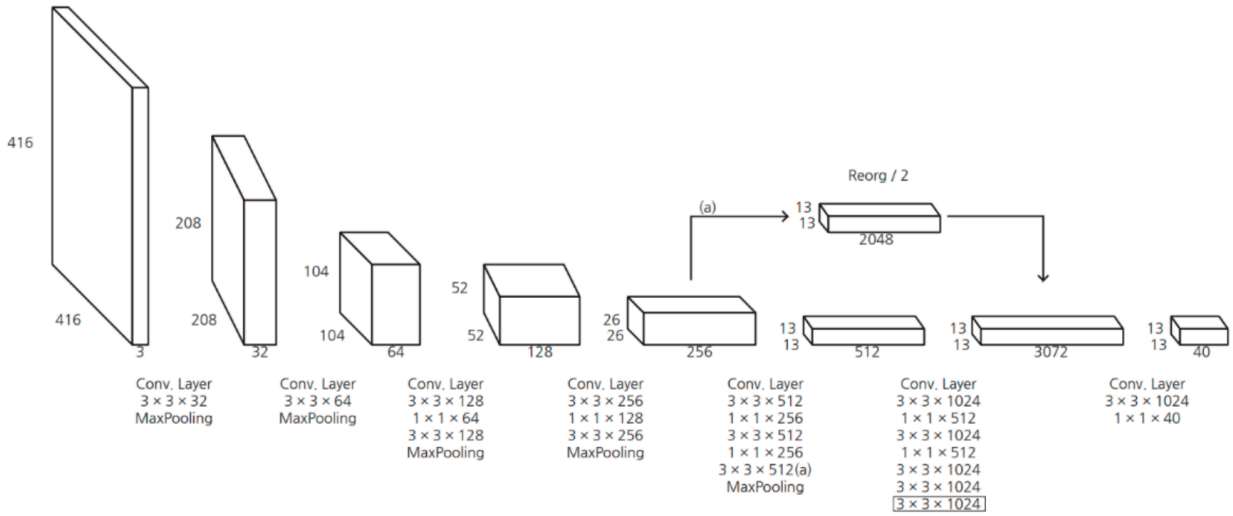

Fig. 1: YOLOv2 Baseline Architecture Diagram

This block diagram outlines the progression of outputs between convolutional and MaxPooling layers of the CNN. The input to the CNN is an image and the output is a set of

bounding boxes that represent the objects in the input image and their corresponding class probabilities. Each block represents a feature map where resolution is determined by the width and height of each block. For instance, the resolution of the far left map is 416 x 416. The feature map spatial dimensions decrease throughout the CNN, therefore producing more compact representations of the input. The arrows seen near the end of the CNN represent a pass-through layer, which facilitates the easy access of a feature map from previous layers directly to deeper layers.

1.2 Loss Function

Loss functions are used to quantify the difference between an estimated parameter and the true value of the parameter. In YOLOv2 training, the loss function quantifies two parameter errors simultaneously: bounding box location error and image classification error. The formula for the loss function is given in Fig. 2 below.

$$
\begin{aligned}
loss_t = \sum_{i=0}^{W}\sum_{j=0}^{H}\sum_{k=0}^{A} \quad & 1_{Max\ IOU < Thresh}\ \lambda_{noobj} * \left(-b_{ijk}^{o}\right)^2 \\
& + 1_{\,t<12800}\ \lambda_{prior} * \sum_{r\ \epsilon\ (x,y,w,h)} \left(prior_k^r - b_{ijk}^r\right)^2 \\
& + 1_k^{truth}\Big(\ \lambda_{coord} * \sum_{r\ \epsilon\ (x,y,w,h)} \left(truth^r - b_{ijk}^r\right)^2 \\
& \qquad + \lambda_{obj} * \left(IOU_{truth}^k - b_{ijk}^o\right)^2 \\
& \qquad + \lambda_{class} * \Big(\sum_{c=1}^{C} \left(truth^c - b_{ijk}^c\right)^2\Big)\Big)
\end{aligned}
$$

Fig. 2: YOLOv2 Loss Function

W and *H* represent the row number and column of an input image's grid cells, which are assigned before training begins. *A* represents the number of anchors in each grid cell, which are a set of predefined bounding boxes which are adjusted using YOLOv2 predictions. This loss function therefore iterates over each grid cell and the number of anchors in each grid cell. The first term in the function calculates the loss for cells with no objects. The second term calculates the loss between anchor boxes and the predictions, but only through the first 12800 iterations. The third term evaluates the accuracy of the location and size of each bounding box. The fourth term evaluates the accuracy of the algorithm's objectness parameter, which describes how confident the algorithm is that an object of interest is present in a given bounding box. The fifth and final term evaluates the accuracy of the algorithm's class prediction for an object, if present.

All terms employ squared error losses and use weights (λ) that can be adjusted depending on how significantly the overall loss function should be impacted by each parameter.

2. Updated Features in YOLOv2

2.1 Batch Normalization

Batch normalization is used to regularize the training process and prevent overfitting by removing dropout layers. This technique normalizes the activations of each convolutional layer and improves the mean Average Precision (mAP) by 2% from YOLOv1.

2.2 High Resolution Classifier

The high resolution classifier operates on feature maps at a relatively high resolution compared to the lower-resolution feature maps used for object localization. This classifier focuses on detecting small objects or objects with fine details. It typically consists of convolutional layers followed by fully connected layers and softmax activation for class prediction. This technique improved the mAP by 4% compared to YOLOv1.

2.3 Anchor Boxes

YOLOv2 introduces the concept of anchor boxes to improve the localization accuracy of detected objects. Anchor boxes are pre-defined bounding boxes of different shapes and sizes. Each grid cell predicts multiple bounding boxes, with each box being adjusted based on the anchor boxes. This allows YOLOv2 to handle objects of various sizes and aspect ratios effectively. To improve prior anchor box sizes, k-means clustering is run on training set bounding boxes.

2.4 Updated Training

In YOLOv1, a deep CNN called Darknet-24 was used and consisted of 24 convolutional layers and 2 fully connected layers. The updated network is Darknet-19, consisting of 19 convolutional layers followed by MaxPooling layers. YOLOv2 also employs multi-scale training, where it trains on images of different resolutions. The end result is a model capable of operating on images of different sizes and improves its performance in terms of precision, speed, efficiency. Table 1 below outlines the process of Darknet-19.

Type	Filters	Size/Stride	Output
Convolutional	32	3×3	224×224
Maxpool		$2 \times 2/2$	112×112
Convolutional	64	3×3	112×112
Maxpool		$2 \times 2/2$	56×56
Convolutional	128	3×3	56×56
Convolutional	64	1×1	56×56
Convolutional	128	3×3	56×56
Maxpool		$2 \times 2/2$	28×28
Convolutional	256	3×3	28×28
Convolutional	128	1×1	28×28
Convolutional	256	3×3	28×28
Maxpool		$2 \times 2/2$	14×14
Convolutional	512	3×3	14×14
Convolutional	256	1×1	14×14
Convolutional	512	3×3	14×14
Convolutional	256	1×1	14×14
Convolutional	512	3×3	14×14
Maxpool		$2 \times 2/2$	7×7
Convolutional	1024	3×3	7×7
Convolutional	512	1×1	7×7
Convolutional	1024	3×3	7×7
Convolutional	512	1×1	7×7
Convolutional	1024	3×3	7×7
Convolutional	1000	1×1	7×7
Avgpool		Global	1000
Softmax			

Table 1: Darknet-19

3. Improvement Attempt

3.1 Primary Attempt

Multiple approaches have been attempted to improve the baseline YOLOv2 algorithm, but so far, only one has been realized. Darknet-19 was modified so that the max-pooling layers were interchanged for convolutional layers. The sizes and strides remained the same (size: 2 x2, stride: stride: 2) between approaches and outputs for each were identical. The motivation for this improvement was to minimize the amount of information loss during this downscaling operation. Max Pooling layers are used commonly to reduce the spatial dimensions (width and height) of the input volume while preserving the most important information by choosing the maximum value in a sliding window position (see Fig. 3).

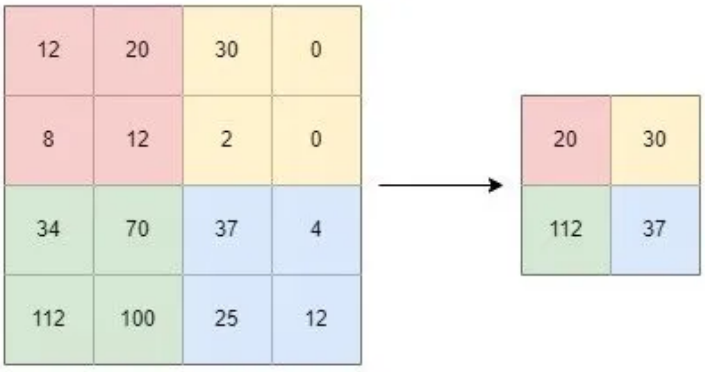

Fig. 3 Max Pooling Visualization

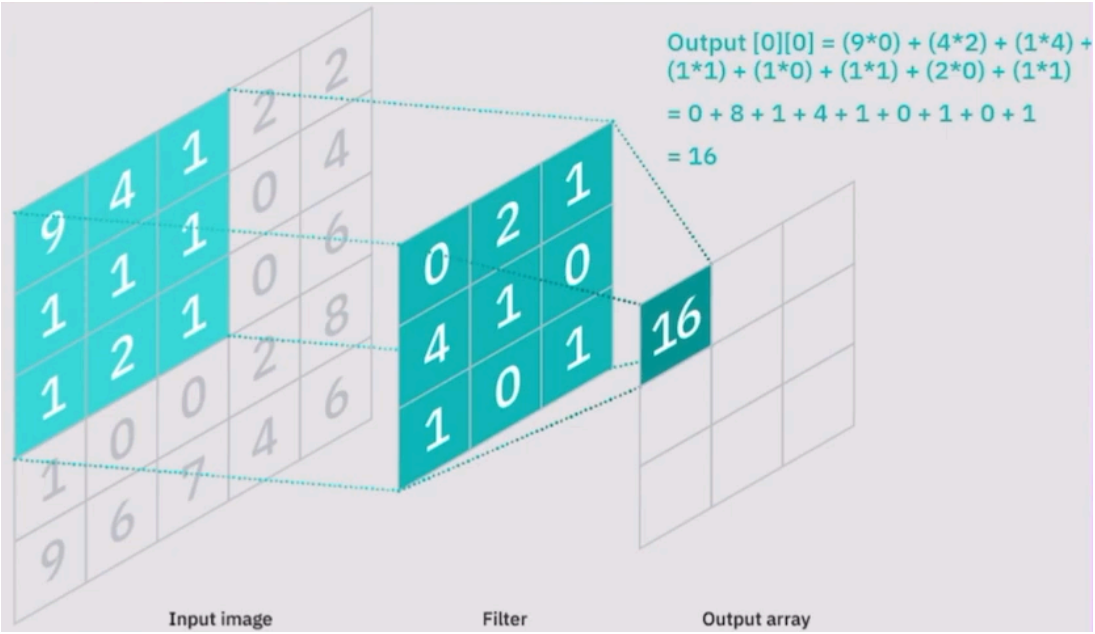

Fig. 4: Convolutional Layer Visualization

A key difference between the Max Pooling and convolutional layers is the amount of information transferred between the two approaches. The Max Pooling approach neglects a significant amount of the information contained in the sliding window. The convolutional layer uses filter weights to strategically increase the transmission of more important information between input and output.

4. Modified YOLOv2 Performance Evaluation

4.1 Mean Average Precision (mAP)

Average Precision measures the average precision across an object class. mAP takes the average across all object classes and represents a metric that conveys overall precision. YOLOv2 computes mAP by calculating the precision-recall curve for each class and then averaging the area under the curve (AP) across all classes. In this modified approach, the mAP improved as more epochs were completed. However, despite completing 50 epochs, the mAPs for classes in the PASCAL VOC dataset had decreased compared to the baseline model. The baseline model achieved an average mAP of about 70%, while the modified version had decreased to about 28% as seen below.

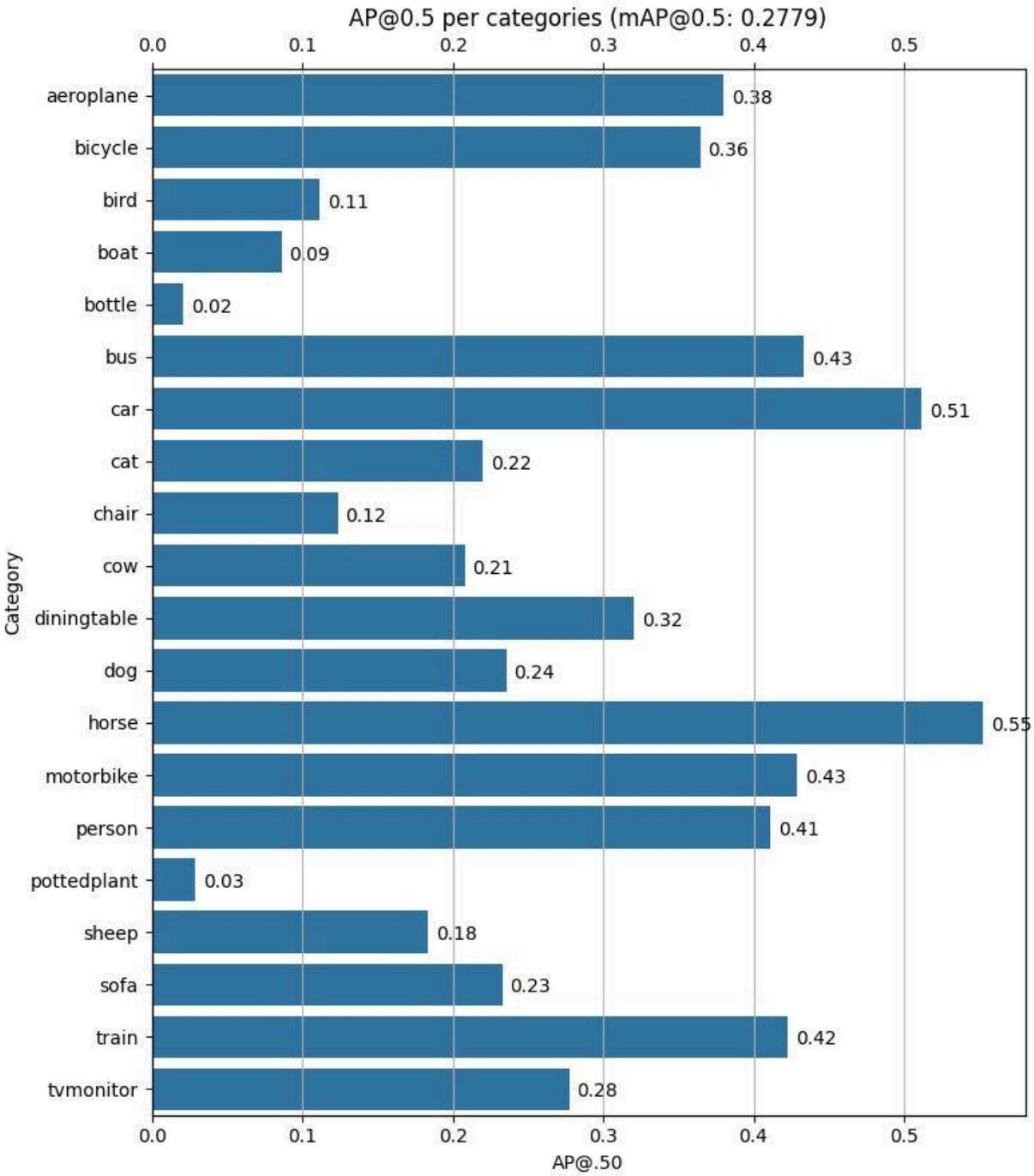

Fig. 5: mAP for PASCAL VOC Dataset Classes at 50 Epochs

A visualization with bounding box predictions is provided for the modified algorithm at 10, 20, 30, 40, and 50 epochs below. Generally, the algorithm improved with more epochs, but

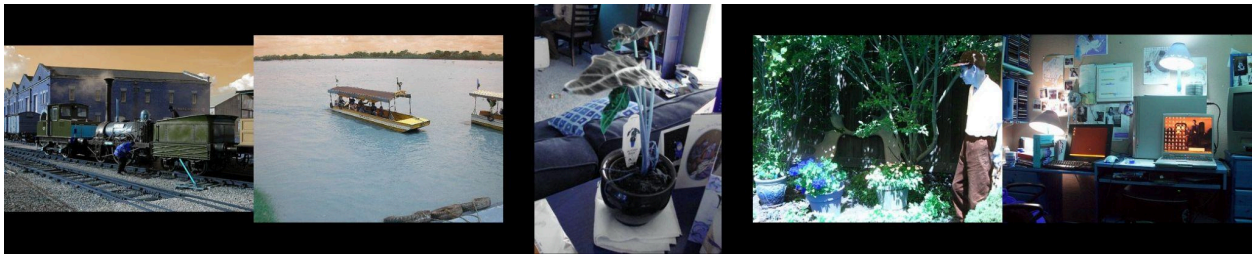

Fig. 6: 10 Epochs

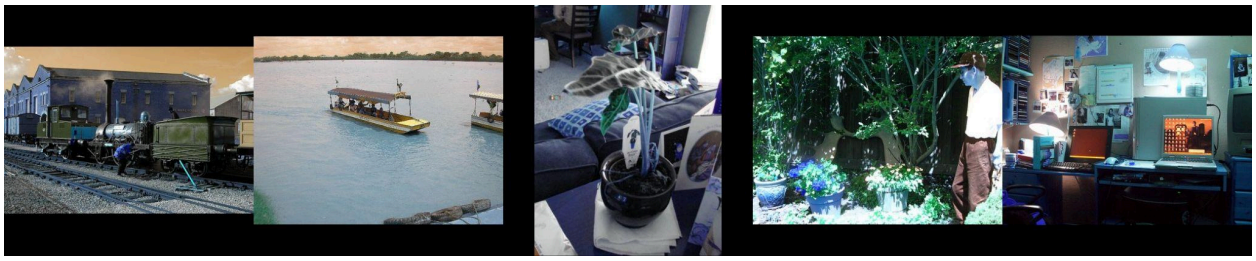

Fig. 7: 20 Epochs

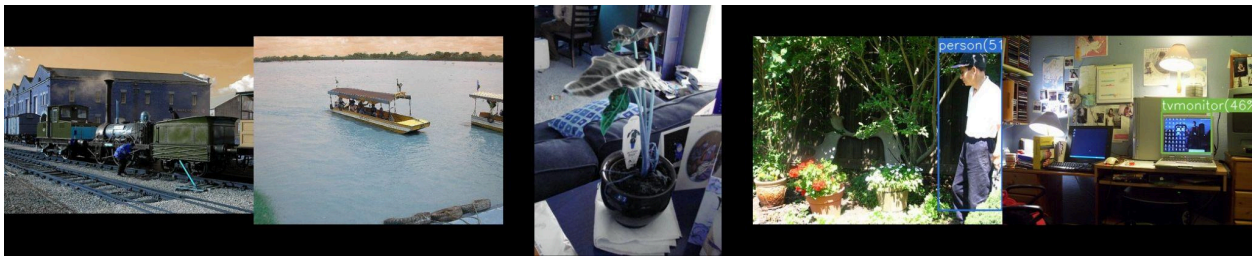

Fig. 8: 30 Epochs

Fig. 8: 40 Epochs

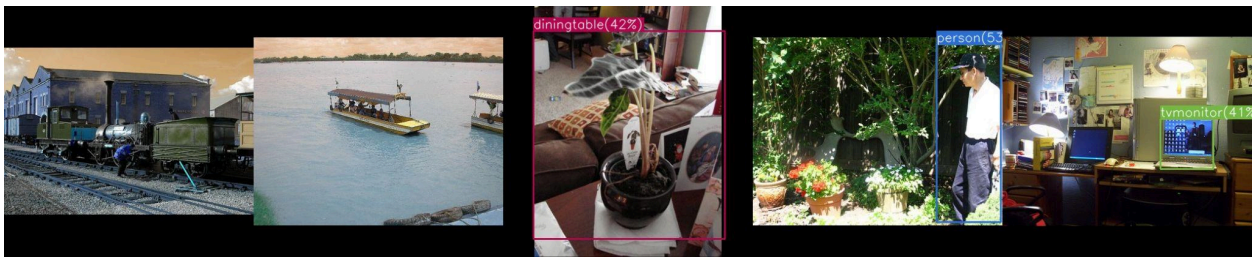

Fig. 8: 50 Epochs

4.2 Frames Per Second (FPS)

Speed and efficiency are crucial in feature extraction software. Performance metrics such as frames per second (FPS) and computational complexity are used to assess the efficiency of YOLOv2. YOLOv2's FPS is influenced by several factors, including the complexity of the CNN backbone, the number of anchor boxes and predictions per grid cell, and the computational resources available (i.e. GPU acceleration). After 50 epochs, the FPS for this approach was 62.63 compared to the baseline 71.88.

References

[1] Redmon, J., & Farhadi, A. (2017). YOLO9000: better, faster, stronger. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 7263-7271).