Report: YOLO9000: Better, Faster, Stronger

Huiqi Vicky Zheng

February 2025

1 Introduction

Object detection identifies and localizes multiple objects in an image. It is used in robotics, self-driving cars, and security systems. Existing systems often struggle to balance speed, accuracy, and scalability. While two-stage detectors like Faster R-CNN achieve high accuracy, their computational complexity limits real-time applications. Single-shot detectors like SSD improve speed but sacrifice accuracy.

The paper "*YOLO9000: Better, Faster, Stronger*" [4] introduces YOLO9000, a state-of-the-art, real-time object detection system capable of detecting over 9000 object categories. The authors propose several improvements to the original YOLO (You Only Look Once) [5] detection method, resulting in YOLOv2, which achieves state-of-the-art performance on well-known benchmarks like Pascal VOC and COCO. They also suggest a way to train the model using both detection and classification data, letting it detect objects even without labeled detection data. This report explains YOLO9000's methodologies, optimizations, and contributions to computer vision.

2 Methodologies

YOLO9000 builds upon YOLOv1 and introduces architectural and optimization improvements to address challenges such as localization errors and missing objects (low recall).

2.1 Enhancements in YOLOv2

Batch Normalization (BN) [2]: Added to all convolutional layers, this improves convergence rates and eliminates the need for dropout, enhancing generalization. This optimization results in a 2% improvement in mean average precision (mAP).

High-Resolution Classifier: Before training for detection, YOLOv2 first fine-tunes its classification network on high-resolution images (448×448) for 10 epochs. Unlike YOLOv1, which handles resolution shifts (224×224 to 448×448) and detection training simultaneously, YOLOv2 separates these adjustments, improving mAP by 4%.

Convolutional with Anchor Boxes [A.3]: YOLOv2 replaces YOLOv1's unconstrained bounding box predictions with anchor boxes inspired by Faster R-CNN's region proposal networks (RPNs) [6]. Instead of absolute coordinates, it predicts offsets to predefined anchors, improving learning stability. The input resolution is reduced to 416×416, resulting in an odd 13×13 grid. With an odd grid, the center of the image, typically containing larger objects, can be more accurately localized, leading to better predictions (See figure 3). This adjustment slightly lowers mAP but boosts recall and prediction stability.

Dimension Clusters: YOLOv2 uses k-means clustering on training data to find anchor box sizes automatically. Earlier models like Faster R-CNN picked these sizes by hand. YOLOv2 uses a custom distance metric: $d(box, centroid) = 1 - IOU(box, centroid)$. This makes sure clustering focuses on box overlap Intersection over Union (IoU) [A.2] instead of size differences (Euclidean distance). Euclidean distance causes bigger errors for bigger boxes because error grows with size. IoU avoids this problem because it works the same for all box sizes. This method creates anchor boxes that fit the dataset better, so predictions are more accurate.

Direct Location Prediction: YOLOv2 predicts box centers relative to grid cells, rather than as offsets from the top-left corner. This improves stability and localization accuracy. RPN methods predict anchor box offsets without constraints, which could result in predictions scattered across the image, regardless of the grid cell. YOLOv1 adopted a similar strategy, but had unstable training, as the model required significant time to learn sensible offsets. YOLOv2 fixes this limiting box coordinates to grid cells using a sigmoid function σ (limited in $[0, 1]$). To define the correct grid cell for each anchor box, the following steps are used (See figure 4):

- First, the grid cell where the top-left corner of the anchor box belongs is identified by coordinates (c_x, c_y). The anchor box has a size of (p_w, p_h).
- Predicts offsets (t_x, t_y, t_w, t_h) and a confidence score t_o (reflects the likelihood of the box containing an object).

The final box parameters are (b_x, b_y, b_w, b_h), where:

$$b_x = \sigma(t_x) + c_x \ , \quad b_y = \sigma(t_y) + c_y$$

$$b_w = p_w \times e^{t_w}, \quad b_h = p_h \times e^{t_h}$$

And the predicted box's confidence is computed as: $\mathbb{P}(object) \times IoU(b, object) = \sigma(t_o)$. This approach improves the accuracy by 5%.

Fine-Grained Features: YOLOv2 improves small object detection (caused by the loss of detailed features as the network deepens) by combining high-resolution (26×26) features with deeper layers (13×13) via a passthrough layer. This merges fine details from earlier layers with broader context, creating a $13 \times 13 \times 2048$ feature map. This approach slightly boosts accuracy, especially for small objects.

Multi-Scale Training: YOLOv2 changes input image sizes during training every 10 batches. Sizes range from 320×320 to 608×608 (in steps of 32). This helps the model work well at different scales, balance accuracy and speed, and makes it robust to varying image sizes.

Darknet19: YOLOv2 uses a custom network called Darknet19 (See figure 5), which balances speed and accuracy better than older networks like VGG-16 or GoogLeNet [7]:

- **Structure:** 19 convolutional layers (mostly 3×3 filters) and 5 max-pooling layers. Uses 1×1 filters to reduce channels, global average pooling for predictions, and BN for stable and faster training.
- **Efficiency:** Fewer operations (5.58 billion) than VGG-16 (30.69 billion) and GoogleLeNet (8.52 billion, used in YOLOv1) but achieves higher accuracy (91.2% vs. 90.0% vs 88.0% on ImageNet).
- **Training:** Initially trained as a classifier on ImageNet, then fine-tuned for detection by replacing the last three convolutional layers with 3×3 convolutional layers (1024 filters) and one final 1×1 layer to predict 125 values (5 anchor boxes $\times$25 values each (4 coordinates, 1 confidence score and 20 class probabilities)).

2.2 Joint Training for YOLO9000

YOLO9000's most significant contribution is its ability to train on a combination of detection and classification datasets, addressing the data scarcity problem in object detection. This is achieved through a hierarchical classification approach using WordTree, a tree-structured representation of object categories derived from WordNet. WordNet organizes words into a hierarchical structure of synsets, representing semantic relationships between concepts. This hierarchy is crucial for enabling joint training.

Instead of using a traditional softmax function over all classes, which assumes mutual exclusivity, YOLO9000 employs a hierarchical softmax. This is essential because categories in combined datasets (like COCO and ImageNet) may overlap (e.g., "dog" and "Norfolk terrier") (see figures 6 and 7). The hierarchical softmax computes probabilities conditionally along the WordTree hierarchy. Let $S(c)$ denote the set of synsets on the path from the root to class c. Then, the probability of class c is given by:

$$P(c) = \prod_{s \in S(c)} P(s|parent(s))$$

where $parent(s)$ is the parent synset of s in the WordTree. This factorization allows the model to predict fine-grained categories while still being able to fall back to broader categories when uncertain. For example, $\mathbb{P}$(Norfolk terrier) = $\mathbb{P}$(Norfolk terrier | terrier) $\times$ $\mathbb{P}$(terrier | dog) $\times$ $\mathbb{P}$(dog | animal) $\times$ $\mathbb{P}$(animal | physical object)
This hierarchical approach offers several advantages:

1. **Data Efficiency:** It allows the model to learn from both detection data (with precise bounding boxes) and classification data (with image-level labels), effectively increasing the training data size and diversity.
2. **Generalization:** By learning the hierarchical relationships between categories, the model can generalize to novel classes even without explicit bounding box annotations, as long as they are present in the WordTree.
3. **Scalability:** The hierarchical structure makes it possible to scale to a massive number of categories, as demonstrated by YOLO9000's ability to detect over 9000 objects.

3 Results

YOLOv2 shows mAP values between 69 and 78.6, depending on the input resolution. The highest mAP for YOLOv2 is 78.6 at 544×544 resolution. This is competitive with Faster R-CNN ResNet (76.4) and SSD500 (76.8). YOLOv2 also has much higher FPS than other models, especially at lower resolutions, making it suitable for real-time applications. It gives higher accuracy at higher resolutions and faster processing at lower resolutions.

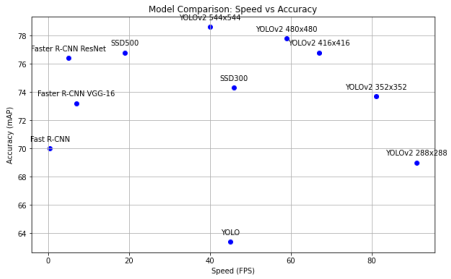

Figure 1: Model Comparison: Speed (FPS) vs Accuracy (mAP)

On the ImageNet detection task, YOLO9000 got an overall mAP of 19.7, despite having only seen classification data for most of the test images. On the 156 object classes it had not seen during training, YOLO9000 got 16.0 mAP. YOLO9000 does well on new categories like animal species. But it has trouble with classes like clothing and equipment. This is because COCO does not have bounding box labels for these categories. For example, it struggles with "sunglasses" or "swimming trunks".

3.1 My Implementation Results

In this section, I present my implementation of the core methods described in the chosen paper. The goal was to implement the key contributions of the paper and demonstrate my understanding of the methodology by applying it to a novel dataset. The code can be found in the GitHub Repository.

I trained and tested the model using a new dataset containing 4 African wildlife species [1]: buffalo, elephant, rhino, and zebra. I aimed to apply the same detection method to wildlife images to ensure the method could be applied effectively to other dataset.

First, I rerun the k-mean clustering on my train dataset and I chose a value of $k = 4$ for the number of anchor boxes by the elbow criteria (See figure 8). To reduce training time, I loaded pretrained weights [3] from the authors. The input images were resized to 416×416 and normalized using ImageNet statistics to maintain consistency with the pretrained model, as the classifier was originally trained on ImageNet.

One challenge I faced was the lack of clarity in the original paper regarding the loss function. To address this, I used a custom loss function, which combines three primary components essential for training the detection model: localization loss (L_{loc}), confidence loss (L_{conf}), and classification loss (L_{class}). L_{loc} quantifies the error in the predicted bounding box location and size. It uses Mean Squared Error (MSE) to compare the predicted box parameters with the ground truth values. The loss is only calculated for grid cells where an object is present. The lambda_coord parameter weights this loss component, giving higher importance to accurate localization. L_{conf} measures the model's confidence in its object presence predictions. It uses Binary Cross-Entropy with Logits loss to compare the predicted objectness score with the ground truth (0 or 1). The loss is calculated separately for grid cells with and without objects, with the lambda_noobj parameter down-weighting the loss for cells without objects. This helps to balance the contribution of these two types of cells to the overall loss. Finally, L_{class} measures the accuracy of the predicted object class. It uses Cross-Entropy loss and is only calculated for grid cells containing objects. It compares the predicted class probabilities with the ground truth class labels.

Another issue was the choice of optimizer. The authors used Stochastic Gradient Descent, but when I implemented it, I encountered NaN loss values. I attempted to reduce the learning rate to overcome this problem, however this significantly increased training time per epoch (from 1 minute to over 4 minutes). Since this made training too computationally expensive, I stopped the training before seeing results. Instead, I switched to the Adam optimizer, which resolved the NaN loss issue and allowed training to proceed normally. The model was initially trained for 50 epochs with a learning rate of 10^{-3}. As the gap between training and validation loss increased, I reduced the learning rate to 10^{-4} and trained for another 20 epochs. However, the performance improved only slightly before stabilizing.

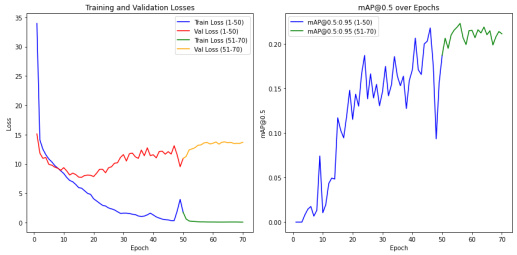

Figure 2: Training Loss vs Validation Loss (right), Train mAP vs Validation mAP (left)

The model achieved a 29.47% mAP on the small wildlife dataset (around 380 images per class). Figure 9 shows that the model performs best on rhinos and buffalo, with weaker results for elephants and zebras. The low zebra performance is surprising given their distinct stripes, while the high rhino performance, despite less distinct features, merits investigation. By visualizing the predictions, one can see that there are more missing detections than misclassification. An issue is the unequal distribution of animal instances within images. While image counts per class are balanced, images often contain multiple animals, skewing the instance distribution. This likely explains the lower zebra performance, the model struggles to detect all zebras, particularly in multi-animal images, leading to more missing detections than for buffalo.

This experiment demonstrates that the core methods from the paper are generalizable to new object categories. While the results are encouraging, accuracy could be improved by expanding the dataset, applying data augmentation for better generalization, and refining the loss function (like handling imbalanced object instances) to enhance localization.

4 Relation of Optimization to Computer Vision

YOLO9000 employs several optimization strategies. Firstly, the k-means clustering of anchor boxes solves a optimization problem to maximize IoU. This data-driven approach outperforms manual anchor selection in Faster R-CNN by ensuring that the number of priors remains small, yet the average IoU remains high. This leads to more accurate bounding box predictions and improved overall detection performance. The random input resizing creates a stochastic effects and improved robustness to varying image scales. And the direct location prediction uses sigmoid function that convert an unconstrained regression problem into a bounded optimization task. This constraint helps to stabilize training and prevents the model from predicting bounding boxes that are far outside the relevant grid cell, leading to improved localization accuracy. Finally, BN stabilizes training by addressing the "internal covariate shift" problem, where the distribution of activations changes

as the network trains, making it difficult for the model to learn. BN normalizes the activations within each mini-batch, stabilizing the training process and allowing for higher learning rates.

5 Contributions and Novelty

YOLO9000 represents a significant advancement in object detection by addressing the limitations of existing methods. Traditional object detection systems like Faster R-CNN and SSD are constrained by the need for large labeled datasets and are limited to detecting a small set of objects. YOLO9000 leverages the hierarchical structure of object classification in ImageNet to combine datasets and expand the scope of detection, bridging the gap between classification and detection datasets.

YOLO9000 achieves real-time inference speeds while also significantly increasing the number of detectable object categories. This combination of speed and scalability was a major advance over previous methods.

Through architectural improvements like BN, anchor box optimization, and multi-scale training, YOLO9000 achieves state-of-the-art accuracy on several benchmarks while maintaining real-time performance.
YOLO9000's joint training approach was a significant departure from previous object detection methods, which typically relied on large, labeled detection datasets. By leveraging the vast amount of classification data available, YOLO9000 paved the way for more scalable and generalizable object detection systems. Its impact can be seen in subsequent research that has explored similar joint training strategies and hierarchical classification methods.

6 Critical Evaluation

While YOLO9000 represents a significant advancement in object detection, it also has some limitations. The performance of YOLO9000 depends heavily on the quality and completeness of WordNet. WordNet may not capture all the nuances of object categories, and it may be biased towards certain domains. Furthermore, constructing and maintaining WordNet can be a significant undertaking. Errors or inconsistencies in WordNet can directly impact the model's performance. Like any machine learning model, YOLO9000 is susceptible to biases present in the training data. If the datasets used for training are not representative of the real world, the model may exhibit biased behavior. For example, if the training data contains mostly images of animals in their natural habitats, the model may struggle to detect animals in other contexts. Moreover, while joint training is a powerful technique, it also has limitations. The model's performance on novel categories without bounding box annotations is still lower than its performance on categories with labeled data. This suggests that while the model can learn to recognize new categories from classification data, it still benefits significantly from having explicit bounding box information.

YOLO9000, like other object detectors, involves a trade-off between speed and accuracy. The choice of input resolution allows for some flexibility in this trade-off, but there is an inherent limit.

Despite these limitations, YOLO9000's contributions are undeniable. It demonstrated the feasibility of joint training for object detection and paved the way for more scalable and generalizable systems. Its real-time performance and ability to detect thousands of object categories make it a valuable tool for a wide range of applications.

7 Conclusion

YOLO9000 represents a landmark in real-time object detection, pushing the boundaries of scalability and efficiency. The authors successfully address several limitations of the original YOLO framework and introduce a novel joint training algorithm that enables the model to detect a vast number of object categories in real-time. The optimization techniques employed, such as BN, dimension clusters, and multi-scale training, significantly enhance the model's performance and robustness.

While the model has some limitations, particularly in detecting certain object classes, its ability to generalize to novel classes without labeled detection data is a major achievement. YOLO9000 sets a new standard for real-time object detection and demonstrates the power of combining optimization techniques with hierarchical classification.

8 References

[1] *African Wildlife Dataset*. https://www.kaggle.com/datasets/biancaferreira/african-wildlife.

[2] Sergey Ioffe and Christian Szegedy. *Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift*. 2015. arXiv: 1502.03167.

[3] Joseph Redmon. *Yolov2 Weights*. URL: https://pjreddie.com/media/files/yolov2.weights.

[4] Joseph Redmon and Ali Farhadi. *YOLO9000: Better, Faster, Stronger*. 2016. arXiv: 1612.08242.

[5] Joseph Redmon et al. *You Only Look Once: Unified, Real-Time Object Detection*. 2016. arXiv: 1506.02640.

[6] Shaoqing Ren et al. *Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks*. 2016. arXiv: 1506.01497.

[7] Christian Szegedy et al. *Going Deeper with Convolutions*. 2014. arXiv: 1409.4842.

A Appendix

A.1 Batch normalization

Batch normalization (also known as batch norm) is a method used to accelerate the training of neural networks by normalizing layer inputs. It re-centers and re-scales these inputs, improving convergence rates and overall training stability.

A.2 Intersection over Union (IoU)

IoU measures the overlap between the predicted bounding box and the ground truth bounding box, defined as: $\text{IoU} = \frac{\text{area of overlap}}{\text{area of union}}$

A.3 Anchor boxes

Anchor boxes are a set of predefined box shapes selected to match ground truth bounding boxes, because most of objects in the training dataset (e.g. person, car, etc.) have a typical height and width ratio. So when predicting bounding boxes we just adjust and refine the size of those anchor boxes to fit the objects, improving prediction stability.

A.4 Figures

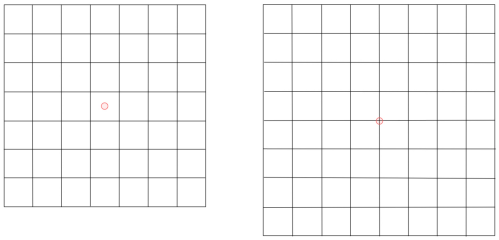

Figure 3: Odd (left, one cell responsible for predicting) and even (right, 4 cells responsible for predicting, where it's unclear which one should be assigned) number of grid cells

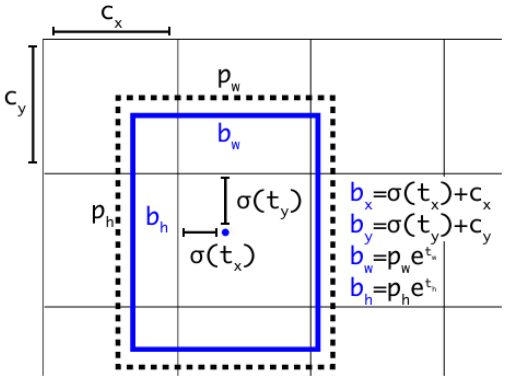

Figure 4: Bounding boxes with dimension priors and location prediction visualization

Type	Filters	Size/Stride	Output
Convolutional	32	3 × 3	224 × 224
Maxpool		2 × 2/2	112 × 112
Convolutional	64	3 × 3	112 × 112
Maxpool		2 × 2/2	56 × 56
Convolutional	128	3 × 3	56 × 56
Convolutional	64	1 × 1	56 × 56
Convolutional	128	3 × 3	56 × 56
Maxpool		2 × 2/2	28 × 28
Convolutional	256	3 × 3	28 × 28
Convolutional	128	1 × 1	28 × 28
Convolutional	256	3 × 3	28 × 28
Maxpool		2 × 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 × 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			

Figure 5: Darknet-19

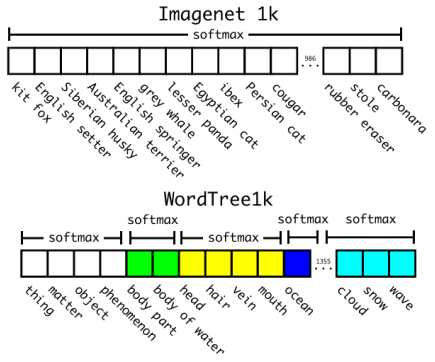

Figure 6: Prediction on ImageNet vs WordTree

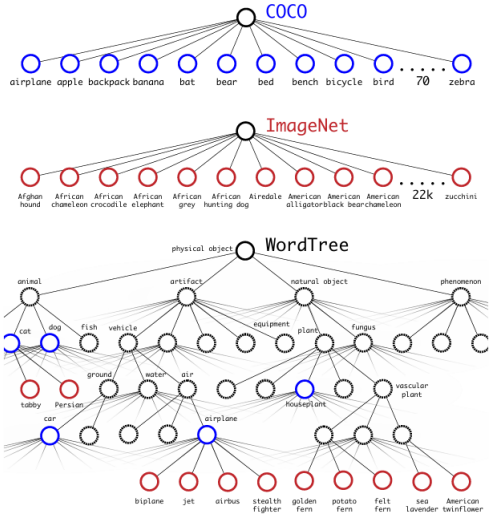

Figure 7: Combining ImageNet and COCO datasets using WordTree hierarchy

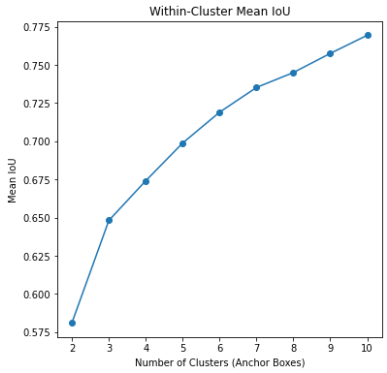

Figure 8: Elbow curve illustrating the optimal number of clusters for the African Wildlife Dataset

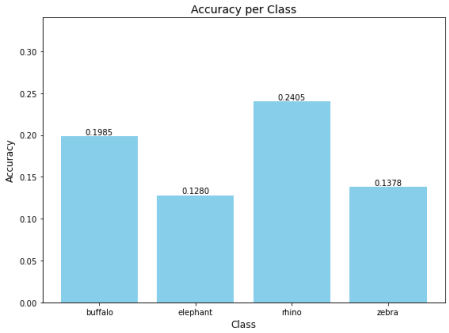

Figure 9: Per class mAP (0.5 : 0.95 IoU) on the African Wildlife Dataset

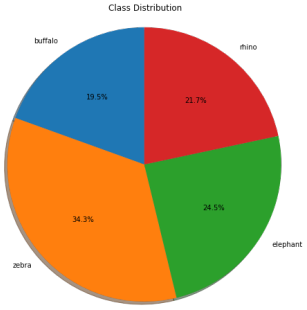

Figure 10: Distribution of animal classes in the test set.

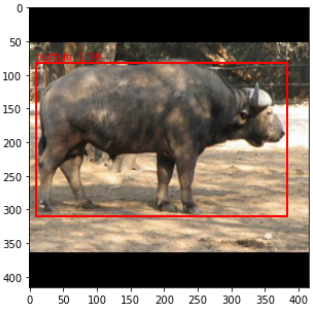

Figure 11: Successful detection of a buffalo with high confidence.

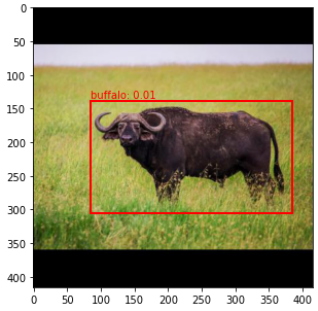

Figure 12: Successful buffalo detection with low confidence.

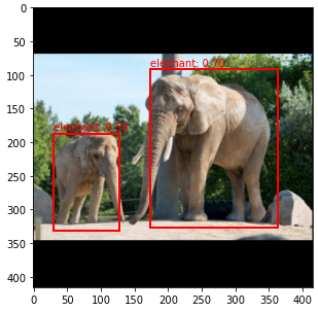

Figure 13: Successful detection of two elephants.

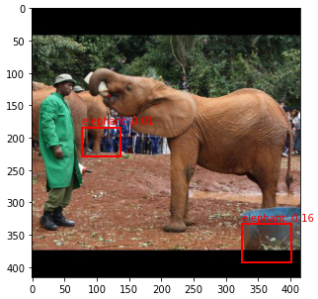

Figure 14: Failure to detect elephants.

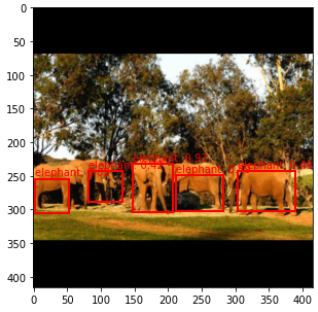

Figure 15: Correct elephant detection with suboptimal bounding boxes.

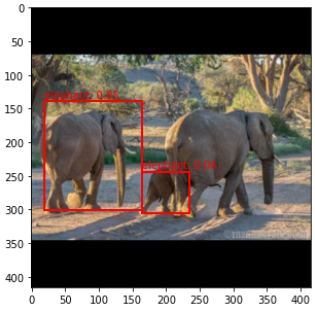

Figure 16: Partial elephant detection: two of three elephants detected.

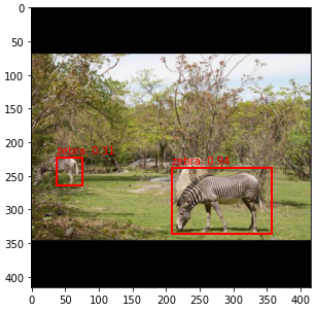

Figure 17: Successful zebra detection, despite partial visibility (but bounding box suboptimal).

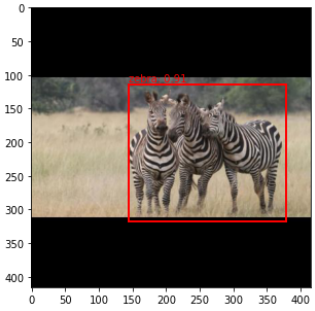

Figure 18: Erroneous grouping of three zebras as a single detection.

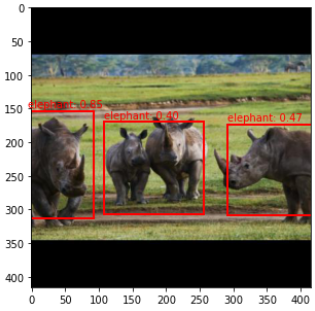

Figure 19: Misidentification of rhinos as elephants.