YOLO 9000 : Better, Faster, Stronger

YOLO9000: Better, Faster, Stronger

Joseph Redmon*†, Ali Farhadi*†
University of Washington*, Allen Institute for AI†
http://pjreddie.com/yolo9000/

Abstract

We introduce YOLO9000, a state-of-the-art, real-time object detection system that can detect over 9000 object categories. First we propose various improvements to the YOLO detection method, both novel and drawn from prior work. The improved model, YOLOv2, is state-of-the-art on standard detection tasks like PASCAL VOC and COCO. Using a novel, multi-scale training method the same YOLOv2 model can run at varying sizes, offering an easy tradeoff between speed and accuracy. At 67 FPS, YOLOv2 gets 76.8 mAP on VOC 2007. At 40 FPS, YOLOv2 gets 78.6 mAP, outperforming state-of-the-art methods like Faster R-CNN with ResNet and SSD while still running significantly faster. Finally we propose a method to jointly train on object detection and classification. Using this method we train YOLO9000 simultaneously on the COCO detection dataset and the ImageNet classification dataset. Our joint training allows YOLO9000 to predict detections for object classes that don't have labelled detection data. We validate our approach on the ImageNet detection task. YOLO9000 gets 19.7 mAP on the ImageNet detection validation set despite only having detection data for 44 of the 200 classes. On the 156 classes not in COCO, YOLO9000 gets 16.0 mAP. But YOLO can detect more than just 200 classes; it predicts detections for more than 9000 different object categories. And it still runs in real-time.

1. Introduction

General purpose object detection should be fast, accurate, and able to recognize a wide variety of objects. Since the introduction of neural networks, detection frameworks have become increasingly fast and accurate. However, most detection methods are still constrained to a small set of objects.

Current object detection datasets are limited compared to datasets for other tasks like classification and tagging. The most common detection datasets contain thousands to hundreds of thousands of images with dozens to hundreds of tags [3] [10] [2]. Classification datasets have millions of images with tens or hundreds of thousands of categories [20] [2].

We would like detection to scale to level of object classification. However, labelling images for detection is far more expensive than labelling for classification or tagging (tags are often user-supplied for free). Thus we are unlikely to see detection datasets on the same scale as classification datasets in the near future.

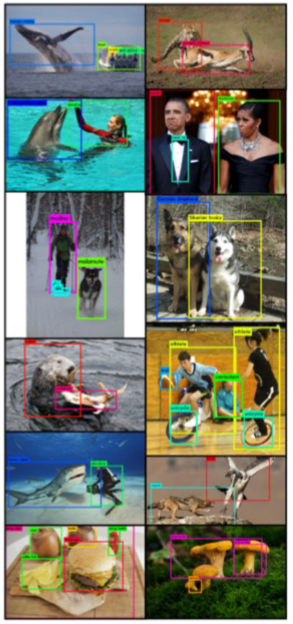


Figure 1: YOLO9000. YOLO9000 can detect a wide variety of object classes in real-time.

We propose a new method to harness the large amount of classification data we already have and use it to expand the scope of current detection systems. Our method uses a hierarchical view of object classification that allows us to combine distinct datasets together.

We also propose a joint training algorithm that allows us to train object detectors on both detection and classification data. Our method leverages labeled detection images to learn to precisely localize objects while it uses classification images to increase its vocabulary and robustness.

Using this method we train YOLO9000, a real-time object detector that can detect over 9000 different object categories. First we improve upon the base YOLO detection system to produce YOLOv2, a state-of-the-art, real-time detector. Then we use our dataset combination method and joint training algorithm to train a model on more than 9000 classes from ImageNet as well as detection data from COCO.

All of our code and pre-trained models are available online at http://pjreddie.com/yolo9000/.

2. Better

YOLO suffers from a variety of shortcomings relative to state-of-the-art detection systems. Error analysis of YOLO compared to Fast R-CNN shows that YOLO makes a significant number of localization errors. Furthermore, YOLO has relatively low recall compared to region proposal-based methods. Thus we focus mainly on improving recall and localization while maintaining classification accuracy.

Computer vision generally trends towards larger, deeper networks [6] [18] [17]. Better performance often hinges on training larger networks or ensembling multiple models together. However, with YOLOv2 we want a more accurate detector that is still fast. Instead of scaling up our network, we simplify the network and then make the representation easier to learn. We pool a variety of ideas from past work with our own novel concepts to improve YOLO's performance. A summary of results can be found in Table 2.

Batch Normalization. Batch normalization leads to significant improvements in convergence while eliminating the need for other forms of regularization [7]. By adding batch normalization on all of the convolutional layers in YOLO we get more than 2% improvement in mAP. Batch normalization also helps regularize the model. With batch normalization we can remove dropout from the model without overfitting.

High Resolution Classifier. All state-of-the-art detection methods use classifier pre-trained on ImageNet [16]. Starting with AlexNet most classifiers operate on input images smaller than 256×256 [8]. The original YOLO trains the classifier network at 224×224 and increases the resolution to 448 for detection. This means the network has to simultaneously switch to learning object detection and adjust to the new input resolution.

For YOLOv2 we first fine tune the classification network at the full 448×448 resolution for 10 epochs on ImageNet. This gives the network time to adjust its filters to work better on higher resolution input. We then fine tune the resulting network on detection. This high resolution classification network gives us an increase of almost 4% mAP.

Convolutional With Anchor Boxes. YOLO predicts the coordinates of bounding boxes directly using fully connected layers on top of the convolutional feature extractor. Instead of predicting coordinates directly Faster R-CNN predicts bounding boxes using hand-picked priors [15]. Using only convolutional layers the region proposal network (RPN) in Faster R-CNN predicts offsets and confidences for anchor boxes. Since the prediction layer is convolutional, the RPN predicts these offsets at every location in a feature map. Predicting offsets instead of coordinates simplifies the problem and makes it easier for the network to learn.

이렇게되면 SSD랑 비슷하지 않나?

We remove the fully connected layers from YOLO and use anchor boxes to predict bounding boxes. First we eliminate one pooling layer to make the output of the network's convolutional layers higher resolution. We also shrink the network to operate on 416 input images instead of 448×448. We do this because we want an odd number of locations in our feature map so there is a single center cell. Objects, especially large objects, tend to occupy the center of the image so it's good to have a single location right at the center to predict these objects instead of four locations that are all nearby. YOLO's convolutional layers downsample the image by a factor of 32 so by using an input image of 416 we get an output feature map of 13×13. 홀수다.

When we move to anchor boxes we also decouple the class prediction mechanism from the spatial location and instead predict class and objectness for every anchor box. Following YOLO, the objectness prediction still predicts the IOU of the ground truth and the proposed box and the class predictions predict the conditional probability of that class given that there is an object.

Using anchor boxes we get a small decrease in accuracy. YOLO only predicts 98 boxes per image but with anchor boxes our model predicts more than a thousand. Without anchor boxes our intermediate model gets 69.5 mAP with a recall of 81%. With anchor boxes our model gets 69.2 mAP with a recall of 88%. Even though the mAP decreases, the increase in recall means that our model has more room to improve.

Dimension Clusters. We encounter two issues with anchor boxes when using them with YOLO. The first is that the box dimensions are hand picked. The network can learn to adjust the boxes appropriately but if we pick better priors for the network to start with we can make it easier for the network to learn to predict good detections.

Instead of choosing priors by hand, we run k-means clustering on the training set bounding boxes to automat-

Original YOLO suffers from several shortcomings.

1) Localization errors
2) Low recall compared region proposal based method.

Original YOLO suffers from several shortcomings.

1) Localization errors
2) Low recall compared region proposal based method.

1. Batch Normalization : 2% Improvement

2. High Resolution

Original YOLO uses 224 x 224, Here, 448 x 448 is used.

3. Anchor Boxes

Used anchor boxes similar to RPNs.

Predict class and objectiveness for every anchor box.

(This increases the number of box predictions from 98 to more than 1,000.
Recall increases while mAP decreases.)

4. Dimension Clusters

Determine the size of anchor boxes with clustering.

1. Batch Normalization : 2% Improvement

2. High Resolution

Original YOLO uses 224 x 224, Here, 448 X 448 is used.

3. Anchor Boxes

Used anchor boxes similar to RPNs.

Predict class and objectiveness for every anchor box.

(This increases the number of box predictions from 98 to more than 1,000.
Recall increases while mAP decreases.)

4. Dimension Clusters

Determine the size of anchor boxes with clustering.

5. Location Prediction

Instead of predicting locations, YOLO9000 predicts box offsets and scaling of box size.

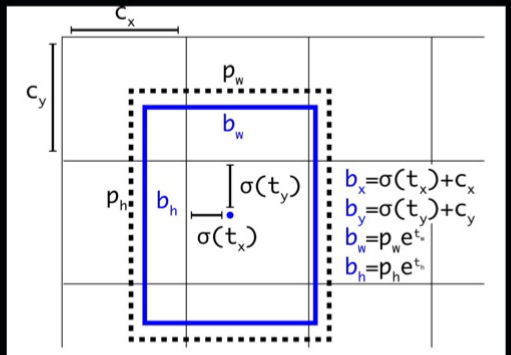
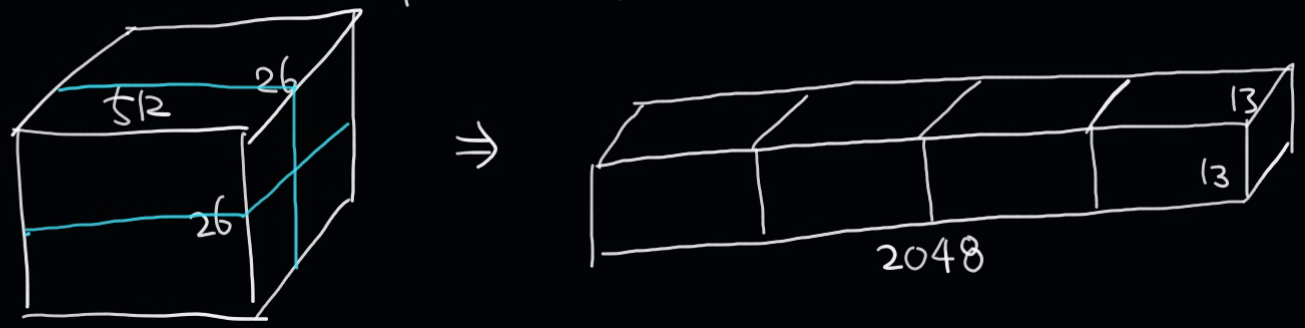


Figure 3: Bounding boxes with dimension priors and location prediction. We predict the width and height of the box as offsets from cluster centroids. We predict the center coordinates of the box relative to the location of filter application using a sigmoid function.

6. Fine-grained Features

Original YOLO uses 13x13 feature map.

YOLO9000 incorporates previous 26x26 feature map



7. Multi-Scale Training

Since YOLO9000 uses conv and pooling, any arbitrary input dimension can be used. (Low resolution → Faster)

Stronger

1. Hierarchical classification

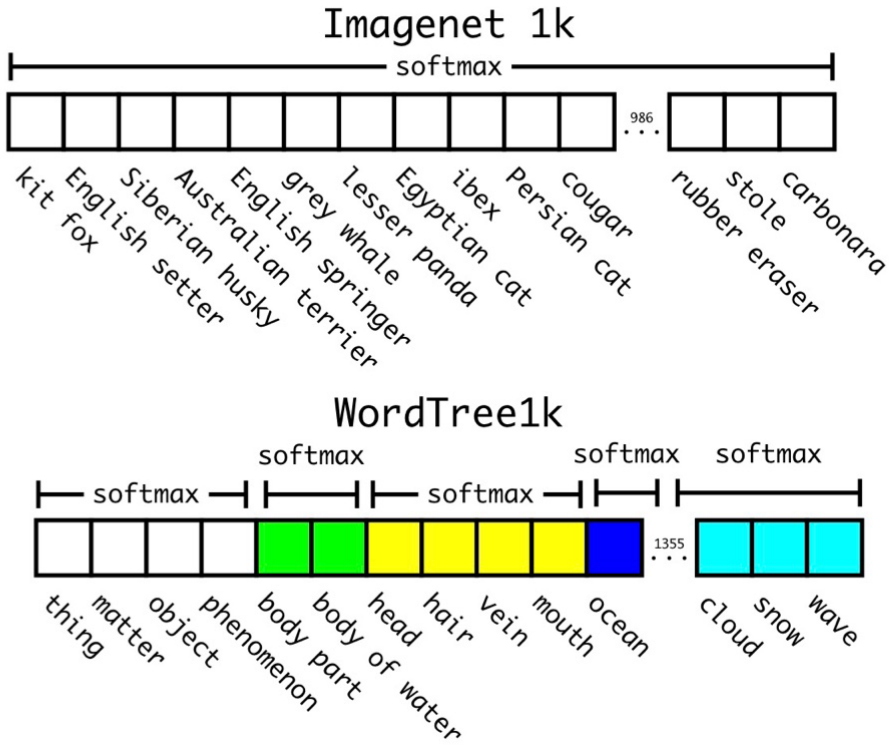
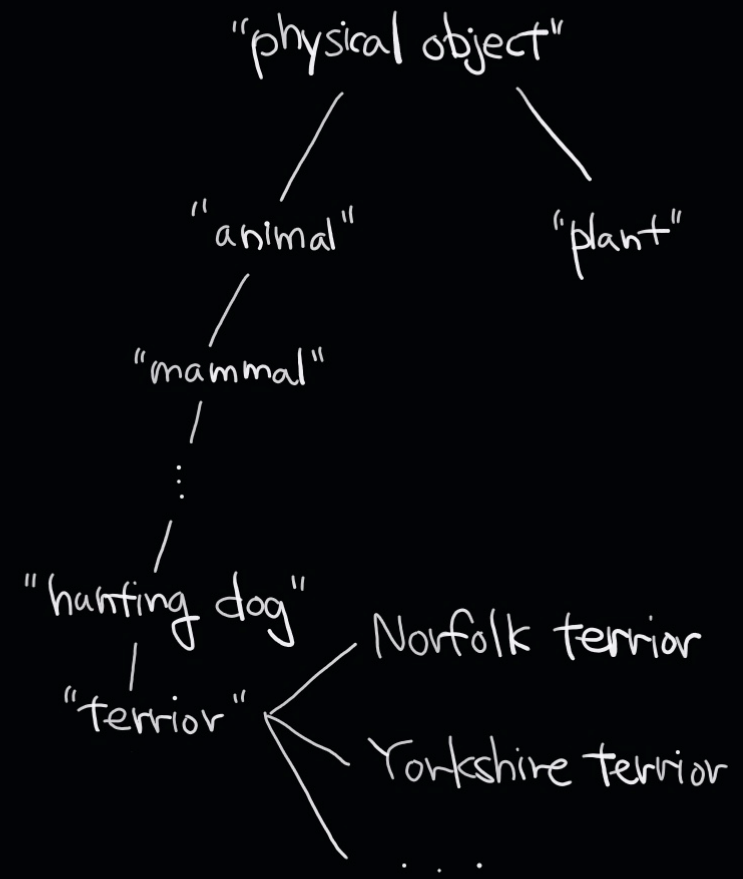


Figure 5: Prediction on ImageNet vs WordTree. Most ImageNet models use one large softmax to predict a probability distribution. Using WordTree we perform multiple softmax operations over co-hyponyms.

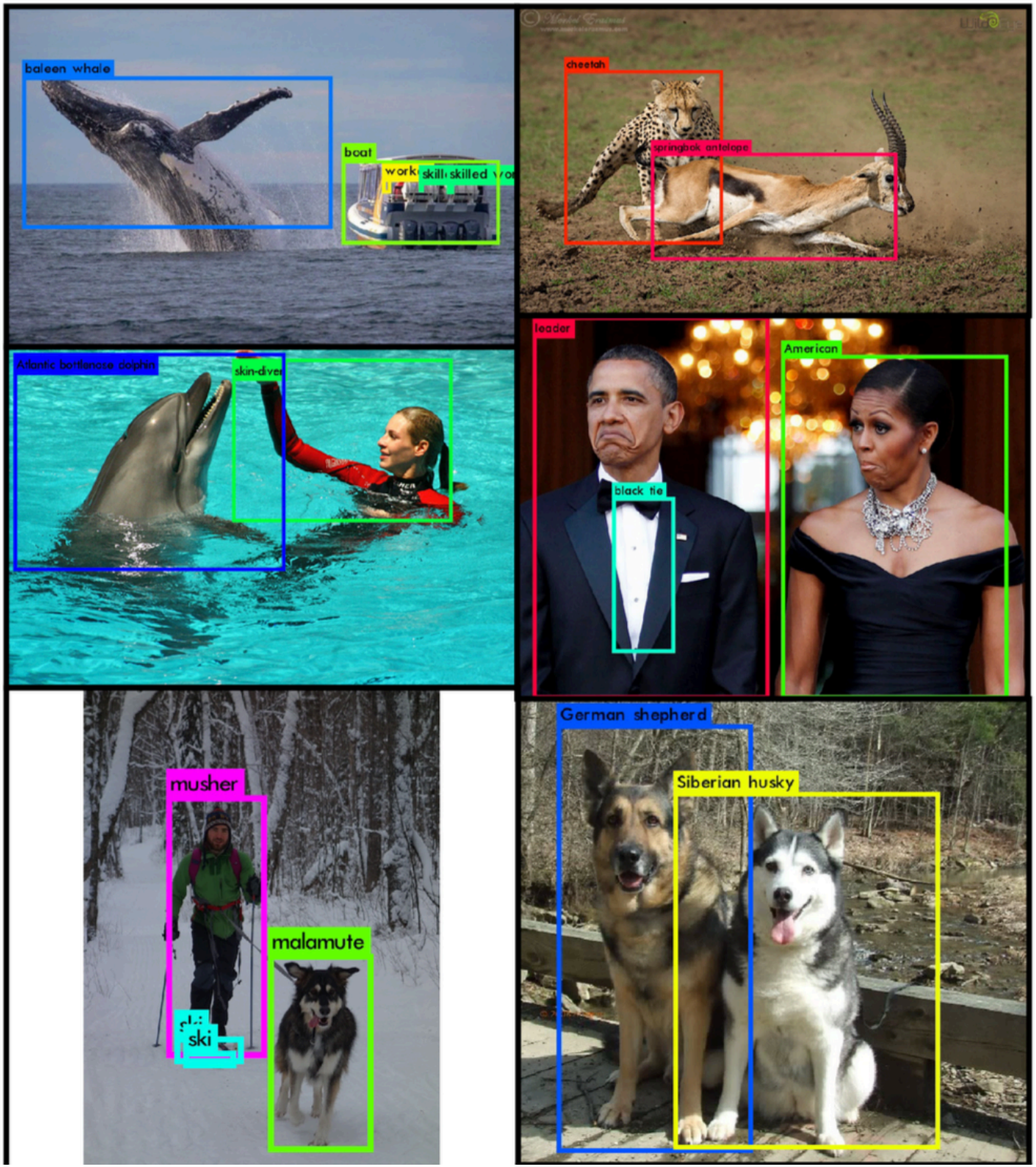
Imagenet labels have a tree structure.

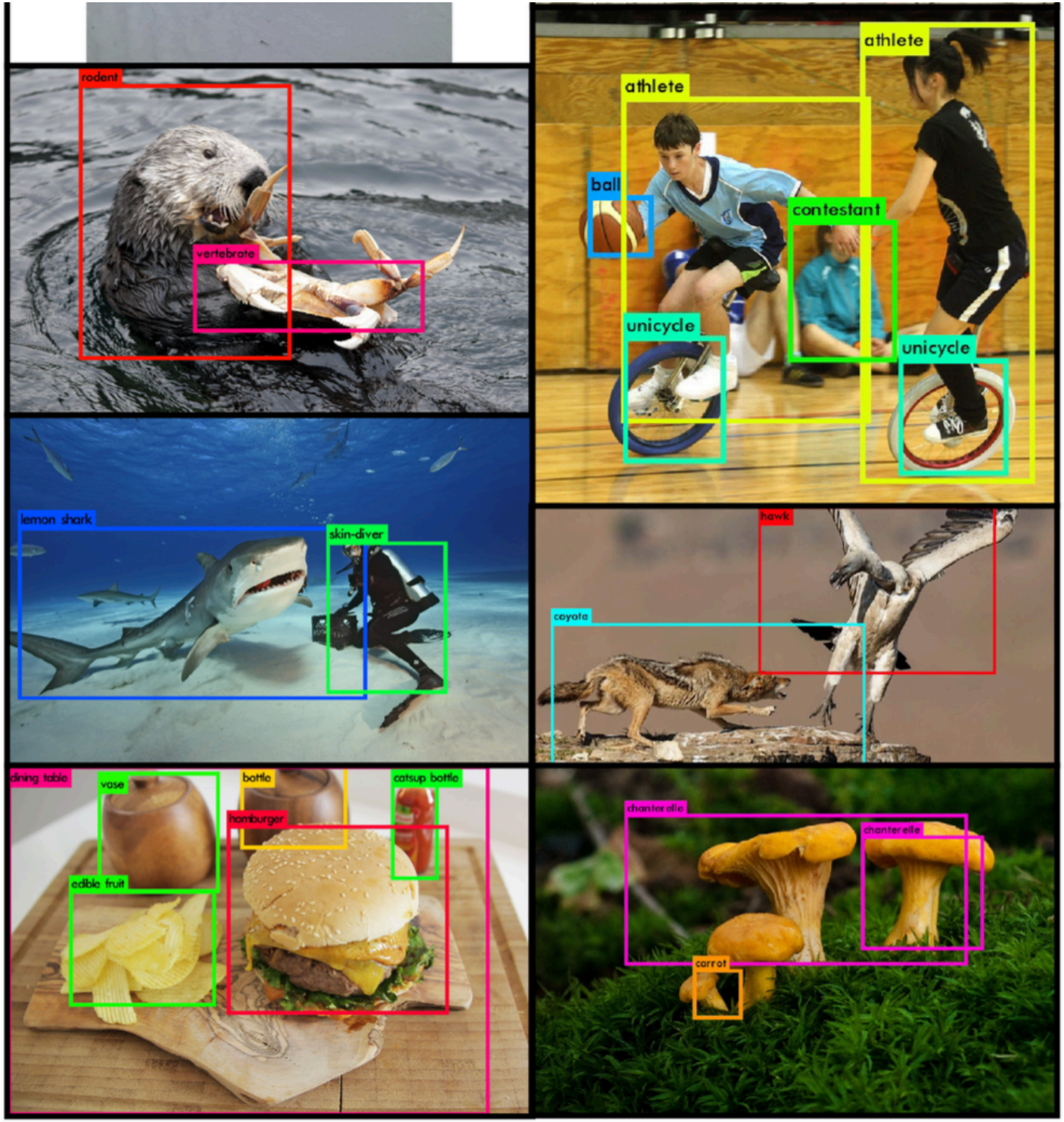


YOLO9000?

During training we mix images from both detection and classification datasets. When our network sees an image labelled for detection we can backpropagate based on the full YOLOv2 loss function. When it sees a classification image we only backpropagate loss from the classification-specific parts of the architecture.

Joint classification and detection. Now that we can combine datasets using WordTree we can train our joint model on classification and detection. We want to train an extremely large scale detector so we create our combined dataset using the COCO detection dataset and the top 9000 classes from the full ImageNet release. We also need to evaluate our method so we add in any classes from the ImageNet detection challenge that were not already included. The corresponding WordTree for this dataset has 9418 classes. ImageNet is a much larger dataset so we balance the dataset by oversampling COCO so that ImageNet is only larger by a factor of 4:1.


baleen whale
boat
cheetah
springbok antelope
Atlantic bottlenose dolphin
skin-diver
leader
black tie
American
musher
malamute
ski
German shepherd
Siberian husky


rodent
vertebrate
athlete
athlete
ball
contestant
unicycle
unicycle
lemon shark
skin-diver
hawk
coyote
dining table
vase
bottle
catsup bottle
hamburger
edible fruit
chanterelle
chanterelle
carrot