Anchor Boxes — The key to quality object detection

# Introduction

One of the hardest concepts to grasp when learning about Convolutional Neural Networks for object detection is the idea of anchor boxes. It is also one of the most important parameters you can tune for improved performance on your dataset. In fact, if anchor boxes are not tuned correctly, your neural network will never even know that certain small, large or irregular objects exist and will never have a chance to detect them. Luckily, there are some simple steps you can take to make sure you do not fall into this trap.

# What are anchor boxes?

When you use a neural network like YOLO or SDD to predict multiple objects in a picture, the network is actually making thousands of predictions and only showing the ones that it decided were an object. The multiple predictions are output with the following format:

Prediction 1: (X, Y, Height, Width), Class  
….  
Prediction ~80,000: (X, Y, Height, Width), Class

Where the(X, Y, Height, Width) is called the “bounding box”, or box surrounding the objects. This box and the object class are labelled manually by human annotators.

In an extremely simplified example, imagine that we have a model that has two predictions and receives the following image:



We need to tell our network if each of its predictions is correct or not in order for it to be able to learn. But what do we tell the neural network it prediction should be? Should the predicted class be:

Prediction 1: Pear  
Prediction 2: Apple

Or should it be:

Prediction 1: Apple  
Prediction 2: Pear

What if the network predicts:

Prediction 1: Apple  
Prediction 2: Apple

We need our network’s two predictors to be able to tell whether it is their job to predict the pear or the apple. To do this there are a several tools. Predictors can specialize in certain size objects, objects with a certain aspect ratio (tall vs. wide), or objects in different parts of the image. Most networks use all three criteria. In our example of the pear/apple image, we could have Prediction 1 be for objects on the left and Prediction 2 for objects on the right side of the image. Then we would have our answer for what the network should be predicting:

Prediction 1: Pear  
Prediction 2: Apple

# Anchor Boxes in Practice

State of the art object detection systems currently do the following:

1. Create thousands of “anchor boxes” or “prior boxes” for each predictor that represent the ideal location, shape and size of the object it specializes in predicting.

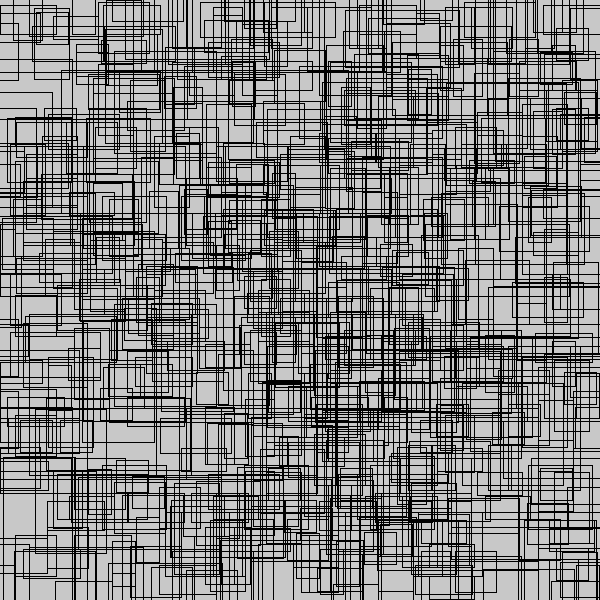
2. For each anchor box, calculate which object’s bounding box has the highest overlap divided by non-overlap. This is called Intersection Over Union or IOU.

3. If the highest IOU is greater than 50%, tell the anchor box that it should detect the object that gave the highest IOU.

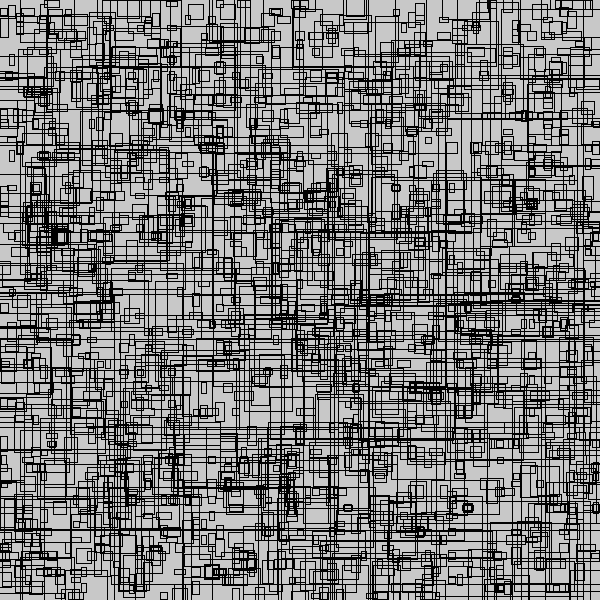
4. Otherwise if the IOU is greater than 40%, tell the neural network that the true detection is ambiguous and not to learn from that example.

5. If the highest IOU is less than 40%, then the anchor box should predict that there is no object.

This works well in practice and the thousands of predictors do a very good job of deciding whether their type of object appears in an image. Taking a look at an [open source implementation of RetinaNet](https://github.com/kuangliu/pytorch-retinanet), a state-of-the-art object detector, we can visualize the anchor boxes. There are too many to visualize all at once, however here are just 1% of them:



Using the default anchor box configuration can create predictors that are too specialized and objects that appear in the image may not achieve an IOU of 50% with any of the anchor boxes. In this case, the neural network will never know these objects existed and will never learn to predict them. We can tweak our anchor boxes to be much smaller, such as this 1% sample:



In the RetinaNet configuration, the smallest anchor box size is 32x32. This means that many objects smaller than this will go undetected. Here is an example from the [WiderFace](http://mmlab.ie.cuhk.edu.hk/projects/WIDERFace/) dataset (Yang, Shuo and Luo, Ping and Loy, Chen Change and Tang, Xiaoou) where we match bounding boxes to their respective anchor boxes, but some fall through the cracks:



In this case, only four of the ground truth bounding boxes overlap with any of the anchor boxes. The neural network will never learn to predict the other faces. We can fix this by changing our default anchor box configurations. Reducing the smallest anchor box size, all of the faces line up with at least one of our anchor boxes and our neural network can learn to detect them!



# Improving Anchor Box Configuration

As a general rule, you should ask yourself the following questions about your dataset before diving into training your model:

1. What is the smallest size box I want to be able to detect?
2. What is the largest size box I want to be able to detect?
3. What are the shapes the box can take? For example, a car detector might have short and wide anchor boxes as long as there is no chance of the car or the camera being turned on its side.

You can get a rough estimate of these by actually calculating the most extreme sizes and aspect ratios in the dataset. YOLO v3, another object detector, uses K-means to estimate the ideal bounding boxes. Another option is to [learn the anchor box configuration](https://arxiv.org/abs/1807.00980).

Once you have thought through these questions you can start designing your anchor boxes. Be sure to test them by encoding your ground truth bounding boxes and then decoding them as though they were predictions from your model. You should be able to recover the ground truth bounding boxes.

Also, remember that if the center of the bounding box and anchor box differ, this will reduce the IOU. Even if you have small anchor boxes, you may miss some ground truth boxes if the stride between anchor boxes is wide. One way to ameliorate this is to lower the IOU threshold from 50% to 40%.

[A recent article](https://www.liip.ch/en/blog/face-detection-an-overview-and-comparison-of-different-solutions-part1) by David Pacassi Torrico comparing current API implementations of face detection highlights the importance of correctly specifying anchor boxes. You can see that the algorithms do well except for small faces. Below are some pictures where an API failed to detect any faces at all, but many were detected with our new model:





If you enjoy this article, you might like reading about [object detection without anchor boxes](https://medium.com/@andersasac/the-end-of-anchors-improving-object-detection-models-and-annotations-73828c7b39f6).

For a more in-depth explanation of anchor boxes you can refer to [Andrew Ng’s Deep Learning Specialization](https://www.youtube.com/watch?v=SXmsPXsYkTw) or Jeremy Howards’s [fast.ai](http://course.fast.ai/lessons/lesson8.html)