Parking Spot Detection Using Neural Network: An Approach to Solve UCI's Parking Issues

Raymond Nguyen
rknguyen@uci.edu

Brennan Tracy
bstracy@uci.edu

Krishen Bhan
krishenb@uci.edu

Pooya Khosravi
pooyak@uci.edu

Timothy Yao
timothy4@uci.edu

Abstract

In this project, we developed a real-time parking spot detection system that uses a deep neural network. This model predicts and localizes vacant and occupied parking spots in videos and images captured from recording devices. To detect multiple parking spots in real time we used Single Shot Multibox Detection (SSD) with VGG16 as our base model. Furthermore, we explored rotated vs. non-rotated bounding boxes in terms of computational complexity and parking spot detection quality.

1 Introduction

1.1 Problem Statement

A major problem facing the University of California, Irvine (UCI) involves the lack of available parking spaces during peak times between 9:00 am and 2:00 pm. Solutions to finding empty parking spaces have often involve placing sensors in each separate parking space. However, this approach can be extremely expensive due to individual sensor costs and potential remodeling of the parking structure. A more feasible approach is the usage of real-time visual object detection. This paper presents a deep network based object detector for parking spot availability around UCI.

1.2 Previous Work

Previous state of the art for object detection have typically consisted of two distinct stages: a region proposal network that performs object localization and a classifier for detecting the types of objects in the proposed regions. Computationally, these can be very expensive and ill-suited for real-world, real-time applications. Example of a two-stage architecture are R-CNN and its variations, Fast R-CNN and Faster R-CNN. Even the fastest high-accuracy detector, Faster R-CNN, operates at only 7 frames per second (FPS). However, there are single shot models such as YOLO [1] and Single Shot MultiBox Detector (SSD) [2] that encapsulate both localization and detection tasks in a single forward sweep of the network. This results in both significantly faster detections and improved deployability on lighter hardware. In order to detect multiple parking spots in real-time, we chose SSD as our base architecture and fine tuned it to our problem space.

1.3 Our Contribution

Our group implemented a SSD neural network that predicts the locations of parking spots and their occupied status. We then further developed the project by prototyping a real-time system identifying parking spots from a video input. We also implemented a modified version of the architecture using rotated bounding boxes and attempted to train the model, but found complications due to computational time complexity.

2 Data

The PKLot dataset was chosen because it contained parking spots with rotated bounding boxes, but due to computational time complexity, we reverted the bounding boxes back to non-rotated bounding boxes. For further validations and flexibility testing, we took pictures of parking spots from UCI.

2.1 PKLot Dataset

The PKLot Dataset [3] consists of 12,417 images of parking lots from 3 different camera angles.

Images are 1280x720 pixels and consist of various lighting levels/weather conditions. Notably, all parking spaces are single lined and the photos were taken at very high angles. Each image has associated labels describing a rotated bounding box for each parking spot and whether the parking spot is occupied. The locations do not change for each image with a given camera angle but the occupied status does.

The feature distributions of our dataset are important for calculating the hyper parameters of our neural network architecture. Individual parking spots varied relatively uniformly in size between 40 pixels and 120 pixels as further away parking spots appeared smaller and the camera locations were on different floors of a building. Rotation angles vary between 0 to -90 degrees resulting in a single canonical representation of a rotated rectangle.

2.2 Preprocessing

We created a non-rotated version of the bounding boxes from the rotated boxes. Then, because the images are taken as a 5-minute time lapse each day and the parking lots don't have too much activity during the day, each parking spot is likely to stay the same over multiple images in a single day. To avoid leaking data between test and validation [3], we randomly selected dates and added them to our training set until we were satisfied with the train/test (80:20) split.

Another limitation of the PKLot dataset is that the 3rd camera angle does not have labels for all parking spots in the image. To handle this, we cropped these images to only include the labeled section of the image.

After the augmentations (Table 1), the images are resized to 300×300 and normalized to ImageNet-stats to fit VGG16's [1] expected input. Notably in regards to later work, these augmentations resulted in parking spaces with scales between $4\% - 40\%$ of the final image.

2.3 Collected Images

In order to test how well our model was generalized to other parking lots and cameras, we took our own photos and videos of the Anteater Parking Structure (APS) from a birds eye view.

2.4 Augmentation

The PKLot dataset is limited to 3 different angles with fixed size and locations for each parking spot. To help generalize our model to detect parking spots with different sizes, locations, and rotations in the images, we implemented the on the fly augmentations summarized in Table 1. These augmentations are used to not only expand our data set in size, but also try to train our model to be more agnostic to camera quality, placement, magnification, resolution, weather conditions, and parking spot locations.

Desired Flexibility	**Augmentation**
Translation	Random Crop, Padding
Size	Random Crop, Padding
Magnification	Random Crop
Rotation	Random Flips
Lighting/Color	Gaussian Noise

Table 1: Summary of the augmentations used for images with both rotated and non-rotated bounding boxes.

3 Approach

3.1 Architecture Design

Base Layers The early network layers are based on VGG16 [1], which is pre-trained on the ImageNet Large Scale Visual Recognition Competition (ILSVRC) classification task [4] as shown in Figure 4. Like the original SSD architecture, the output layer and the dropout layers are removed. Additionally, the last two fully connected layers are converted convolutional layers. The fifth pooling layer is also modified from a 2×2 kernel and 2 stride to a 3×3 kernel and 1 stride. The effect this has no longer halves the dimensions of the feature map from the preceding convolutional layer.

Auxillary Layers As shown in Figure 5, more convolutional layers are stacked on top of our base network. These convolutions provide additional feature maps, each progressively smaller than the last. There are now a total of six feature maps.

Prediction Layers Each added feature map can produce a fixed set of detection predictions using convolutional filters. A set of anchor boxes is defined for every feature map location. Therefore, for every anchor box at a pixel loca-

tion in each feature map, we predict the offsets of the bounding boxes in the form of $(x_{center}, y_{center}, width, height)$ in addition to a set of class scores. This is done by adding two convolutional layers, one for localization predictions and one for class predictions for every anchor box as shown in Figure 6.

3.2 Anchor Box Generation, Scales, and Aspect Ratios

In single shot detection, by using feature maps at different depths in the network, the network can handle objects of varying sizes. Deeper feature maps are smaller corresponding to empirically different receptive fields [5] and therefore appropriate to be trained to predict larger scale objects [2]. Anchor boxes are precalculated, fixed boxes within the feature maps which collectively represent probable and approximate box predictions. They are manually but carefully chosen based on the shapes and sizes of ground truth objects in the PKLot dataset. The scales of the anchor boxes corresponding to m feature maps are computed with:

$$s_k = s_{min} + \frac{s_{max} - s_{min}}{m - 1}(k - 1), \quad k \in [1, m] \tag{1}$$

where there are k layers between the first and last feature map, with s_{min} and s_{max} being the minimum and maximum area of a parking space relative to the image size. Specifically, these values were calculated to be $s_{min} = 0.06$ and $s_{max} = 0.44$ (section 2.2). Aspect ratios of the anchor boxes were left to be $\{1, 2, 3, \frac{1}{2}, \frac{1}{3}\}$. In the SSD architecture, anchor box scales do not need to correspond to layer receptive fields and each feature map learns to be responsive to the corresponding object scale [2].

3.3 Multibox Loss Function

Matching Predictions to Ground Truth We match predictions to their ground truths by finding the Jaccard Overlap (IoU) between the anchor boxes and the ground truth boxes.

$$IoU = \frac{Area_{\cap}}{Area_{\cup}} \tag{2}$$

Each anchor box is first matched with the ground truth box it has the highest overlap with. If an anchor box is matched with an object with a Jaccard overlap of less than 0.5, then it is considered a negative match and positive otherwise. Now, each anchor box has a match, positive or negative and by extension, each prediction has a match, positive or negative. These matches are used as the targets for prediction.

Hard Negative Mining Since most of the anchor boxes are matched as negative after the matching step, there is a significant imbalance between positive and negative training examples. In order to speed up optimization and training times with steeper gradients [6], we sort the matches using the highest confidence loss for each anchor box such that the ratio of negatives and positives is at most 3:1.

Loss Function The overall objective loss function is a weighted sum of the localization loss [A.1] (loc) and the confidence loss [A.2] (conf):

$$L(x, c, l, g) = \frac{1}{N}(L_{conf}(x, c) + \alpha L_{loc}(x, l, g)) \tag{3}$$

where N is the number of matched default boxes and $\alpha = 1$.

3.4 Training

The model was trained over 100 epochs, a batch size of 8, a learning rate of 1×10^{-3}, 0.9 momentum and weight decay of 5×10^{-4}.

3.5 Non-Maximum Suppression [A.3]

For our model, we chose a minimum overlap of 0.3 and a threshold score of 0.7 as NMS parameters because they empirically gave us the best results.

4 Results

Figure 7 illustrates multibox loss over 100 epochs. Figure 2 shows the results of our model from both the test set and Anteater Parking Structure (APS). It can be seen that on the test data, the model predicted the bounding boxes and classification of vacant parking spaces perfectly; however when predicting on APS, the model seems to make consistent errors. For example, it frequently misclassifies the tops of trees as empty parking spots. The model also classifies street lanes as parking spaces. The video in Appendix C shows our

model running in real-time at 51 FPS on Google Cloud's Nvidia P100 GPU. In our video tests (Appendix C), our model appeared to prefer classifying parking spots when they appeared at specific angles. The range of predicted class probabilities on our test set were in the 99th percentile, but the predicted class probabilities on APS ranged from 40% to 70%.

5 Discussion

Our original attempt at tackling this problem used rotated bounding boxes as this would result in the tightest bounds given our dataset. Previous work [7, 8] predicted rotated bounding boxes through modifications to SSD's loss function, anchor box generation, and bounding box parameterization. We originally attempted to recreate this work but found computing the matching and loss to be unfeasible in a reasonable amount of time/money. The bottleneck is computing the pairwise intersection between two sets of general polygons(rather than aligned rectangles). State of the art GPU solutions are limited to an 18x speedup in comparison with the best CPU solution [9] and implementations are proprietary and not open source. We attempted to use CPU-bound R-trees but found that it took 4 seconds to compute loss on a single image (15 hours for one epoch) on a NVIDIA V100 GPU. Given the prolonged training times for the rotated bounding boxes, we opted to convert all the rotated bounding boxes to non-rotated bounding boxes.

Our results show that the model predicted parking spaces perfectly for our test set, but predicted poorly for the images of APS. Our observed frequent misclassifications seem to be due to both our usage of PKLot as a dataset and limitations with our augmentations. Our dataset contains multiple labeled parking spots partially occluded by trees. It also does not contain negative examples of street lines. We suspect these shortcomings caused their corresponding issues. As a general case, we would have preferred our dataset to have more comprehensive variety of occluding and nearby objects. We conjecture that our model preferred parking spots at angles that frequently occurred in the dataset's parallel parking lots. This issue could have been accounted for with affine transformations but since it results in unrealistic images, we decided to forgo including them. PKLot's three camera angles do not cover the wide variety of possible camera angles. Hence, the usage of PKLot poorly generalized our model to images of parking spaces at APS. We believe this issue could be resolved by collecting our own labeled images that are more representative of parking spaces in general. However, due to time constraints, we were unable to label the collected images from APS. Additionally, for future works, we suspect that the implementation of focal loss [10] over multibox loss could lead to more accurate results.

6 Conclusion

This paper introduces alterations the SSD architecture to get better results in solving the problem of identifying occupied and unoccupied parking spots in various parking lots. Our group implemented and trained a model suitable to continuously detect parking spots in real time. We identified what work would need to be done to solve this common problem here at UCI. We believe with some more tuning, our model would be a feasible and cost-effective solution to a growing concern here at UCI.

References

[1] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014. URL `http://arxiv.org/abs/1409.1556`.

[2] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott E. Reed, Cheng-Yang Fu, and Alexander C. Berg. Ssd: Single shot multibox detector. In *ECCV*, 2016.

[3] Paulo RL De Almeida, Luiz S Oliveira, Alceu S Britto Jr, Eunelson J Silva Jr, and Alessandro L Koerich. Pklot–a robust dataset for parking lot classification. *Expert Systems with Applications*, 42(11):4937–4949, 2015.

[4] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR09*, 2009.

[5] Zhou Bolei, Aditya Khosla, Agata Lapedriza, Aude Oliva, and Antonio Torralba. Object detectors emerge in deep scene cnns. 2015.

[6] Abhinav Shrivastava, Abhinav Gupta, and Ross Girshick. Training region-based object detectors with online hard example mining. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 761–769, 2016.

[7] Lei Liu, Zongxu Pan, and Bin Lei. Learning a rotation invariant detector with rotatable bounding box. *CoRR*, abs/1711.09405, 2017.

[8] Shuxin Li, Zhilong Zhang, Biao Li, and Chuwei Li. Multiscale rotated bounding box-based deep learning method for detecting ship targets in remote sensing images. *Sensors*, 18(8):2702, 2018.

[9] Kaibo Wang, Yin Huai, Rubao Lee, Fusheng Wang, Xiaodong Zhang, and Joel H Saltz. Accelerating pathology image data cross-comparison on cpu-gpu hybrid systems. *Proceedings of the VLDB Endowment*, 5(11): 1543–1554, 2012.

[10] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *Proceedings of the IEEE international conference on computer vision*, pages 2980–2988, 2017.

Appendix A Additional Details

A.1 Localization Loss

The localization loss is a Smooth L1 loss between the predicted box (l) and the ground truth box (g) parameters. We regress to offsets for the center (c_x, c_y) of the default bounding box (d) and for its width (w) and height (h).

$$m \in \{cx, cy, w, h\}$$
$$L_{loc}(x, l, g) = \sum_{i}^{N} \sum_{m} x_{ij}^{k} \text{smooth}_{L1}(l_i^m - \hat{g}_j^m)$$
$$\hat{g}_j^{cx} = (g_j^{cx} - d_j^{cx})/d_i^w \qquad \hat{g}_j^{cy} = (g_j^{cy} - d_j^{cy})/d_i^h$$
$$\hat{g}_j^w = \log\left(\frac{g_j^w}{d_i^w}\right) \qquad \hat{g}_j^h = \log\left(\frac{g_j^h}{d_i^h}\right)$$

For the rotated boxes, we will an additional variable angle (a). To calculate the difference for the angle, tangent of the difference is used [7]. So, we implemented the following localization loss:

$$m \in \{cx, cy, w, h, a\}$$
$$L_{loc}(x, l, g) = \sum_{i}^{N} \sum_{m} x_{ij}^{k} \text{smooth}_{L1}(l_i^m - \hat{g}_j^m)$$
$$\hat{g}_j^{cx} = (g_j^{cx} - d_j^{cx})/d_i^w \qquad \hat{g}_j^{cy} = (g_j^{cy} - d_j^{cy})/d_i^h$$
$$\hat{g}_j^w = \log\left(\frac{g_j^w}{d_i^w}\right) \qquad \hat{g}_j^h = \log\left(\frac{g_j^h}{d_i^h}\right)$$
$$\hat{g}_j^a = \tan\left(g_j^a - d_i^a\right)$$

A.2 Confidence Loss

Class or Confidence Loss is softmax loss over multiple classes confidences with background being included in the classes.

A.3 Non-Maximum Supression

Considering the large number of prediction boxes generated from SSD, it is necessary to apply non-maximum suppression in order to filter out a majority of the predicted bounding box. Nonmaximum suppression eliminates boxes that do not meet a certain threshold score. In addition, it eliminates boxes that have a jaccard index overlap of a certain threshold between 0 and 1.

Appendix B Contributions

B.1 Raymond Nguyen

Raymond was involved in selection and research of both the dataset and model architecture. He helped with parsing the data from PKLot into a usable format and additionally collected images of APS for validation. He experimented with various implementations of models such as YOLO, UNET, etc, ultimately deciding on SSD as suitable real-time solution. Raymond is also responsible for re-implementing the SSD architecture in Pytorch and customizing the model to fit the problem space. In terms of more technical details, he helped with data loading, anchor box generation, calculations of scales/aspect ratios, implementation of base, auxillary and prediction layers, and

implementation of the loss function. Furthermore, he was also involved in training and parameter optimization.

B.2 Brennan Tracy

After reading the literature on bounding box based multiple object detection, Brennan inferred the reasoning behind their choices in algorithms and hyperparameters for their specific problems and datasets. Further, based on our differing problem type and dataset, he implemented those algorithms and calculated the hyperparameters. In addition, Brennan implemented utility code for ground truth/prediction visualization, polygon transformations, and dataset analysis. He also split the dataset into training and validation sets based on the PKLot's specifications.

B.3 Pooya Khosravi

Pooya worked on pre-processing and cleaning up the images including creating non-rotated bounding boxes and removing non-labeled spots. He further worked on data augmentation for images and transformations for both rotated and non-rotated bounding boxes. For validation, he collected images of APS. After researching about SSD, DR-Box, and the ways to improve the anchor box computations, Pooya implemented R-trees to speed up Jaccard calculations. He also set up and maintained shared development environments and instances on Google Cloud Platform.

B.4 Krishen Bhan

Krishen worked on the initial implementation of the anchor box representation. Anchor boxes were stored in a R-tree, such that given coordinate points of a polygonal box, overlapping anchor boxes could be determined and visualized in a grid, along with intersection over union values between boxes (with computations from the CPU). Krishen also set up the pipeline to draw labeled bounding box predictions on the frames in a video.

B.5 Timothy Yao

Tim helped with the implementation of the streaming pipeline and YOLO. He researched about the other alternatives of SSD. He worked on the data augmentation for images and transformations for non-rotated bounding boxes. Additional Tim helped with training the model and setting up the GCP environment.

Appendix C Videos

The following are results of our model predicting on videos taken from APS mimicking a steady/rotating surveillance camera.

Moving Camera:

`https://youtu.be/exzxPoTZhBI`

Steady Camera:

`https://youtu.be/yiqaW-X2kuE`

Appendix D Figures

Figures are included in the following pages.

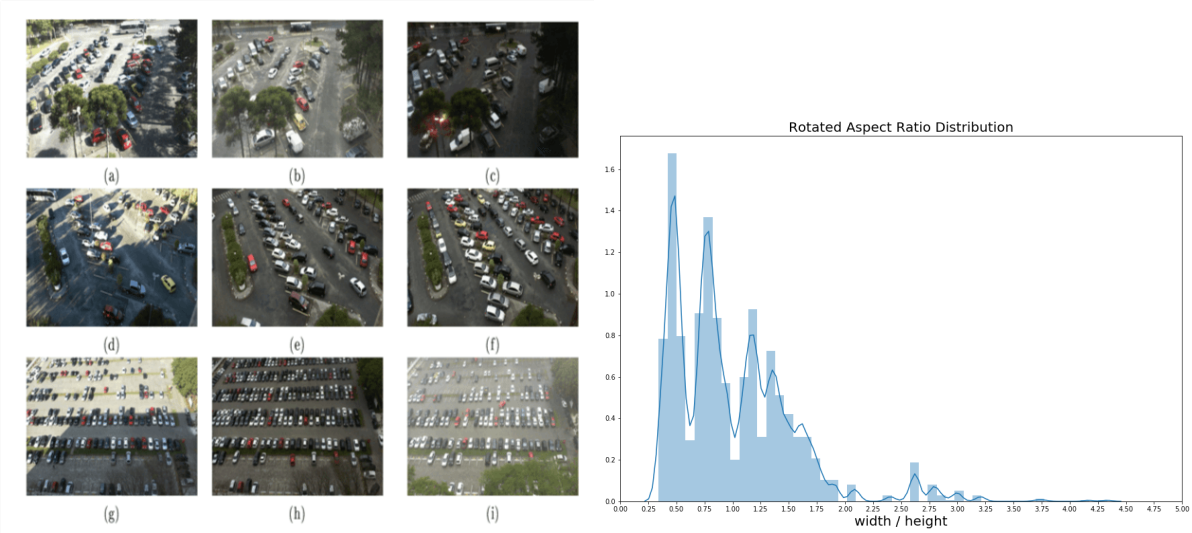

Figure 1: **PKLot samples and analytics:** (Left) Example images from PKLot. (Right) Histogram of parking spot aspect ratio (w/h) distribution. Results are shown to be between 0.25 and 4.5

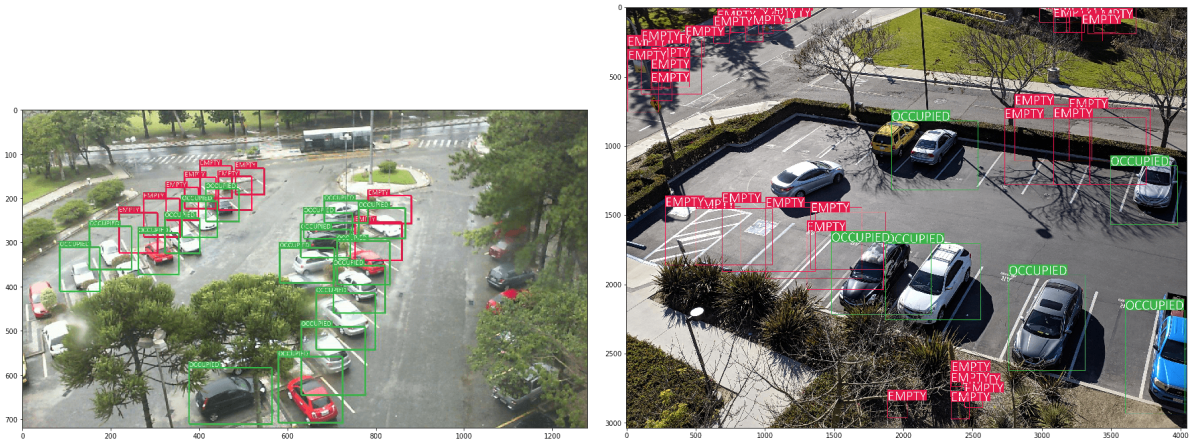

Figure 2: **Model results:** (Left) Labeled predictions from the test set. (Right) Labeled predictions from Anteater Parking Structure (APS).

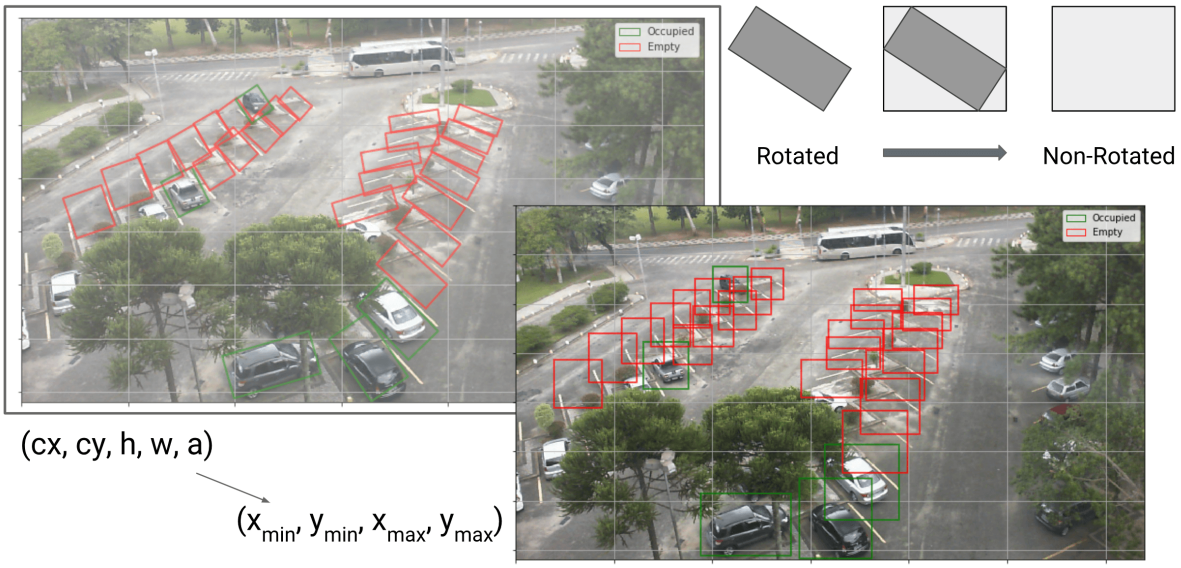

Figure 3: **Visualization of straightening process:** This figure shows a pre-processing step of converting rotated rectangles to straight rectangles bounding them.

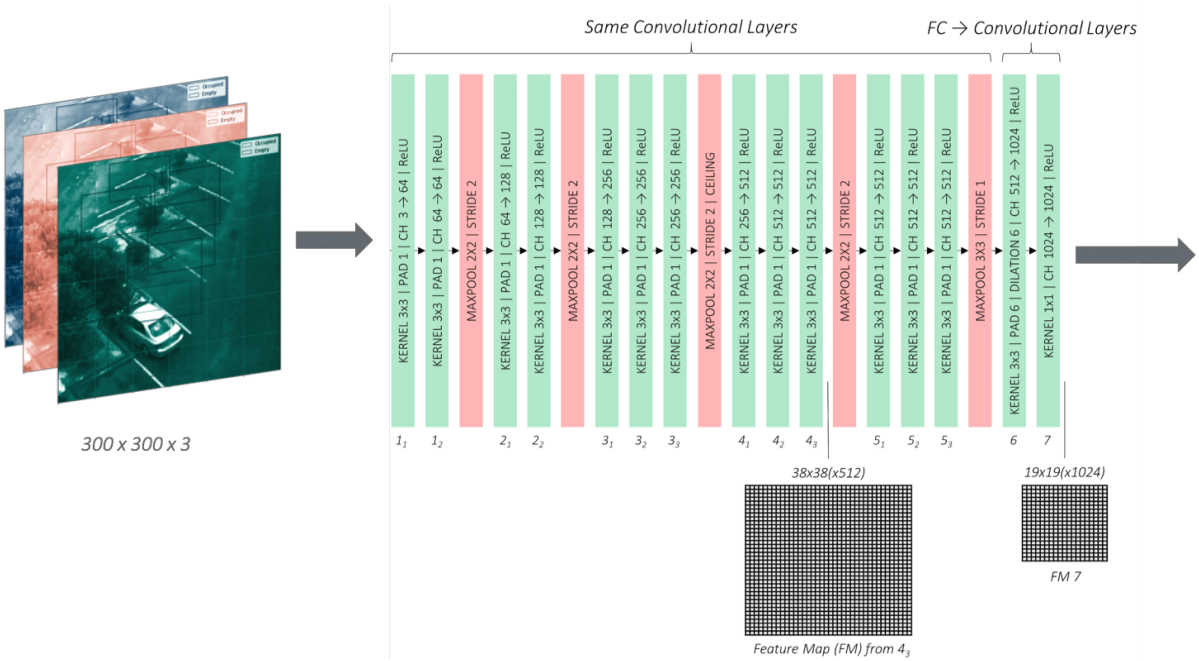

Figure 4: **Using VGG as base model:** This is the VGG16 with the fully connected layers at the end modified to convolution layers.

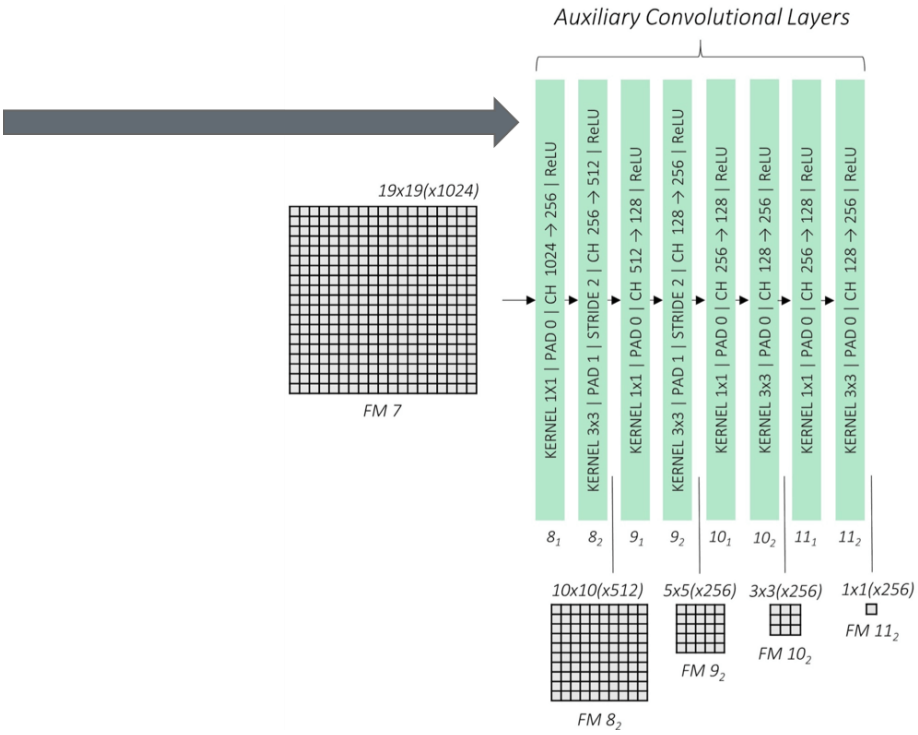

Figure 5: **Auxiliary layers:** Visualization of the auxiliary layers and the feature maps from the 19 x 19 feature map from the base convolution.

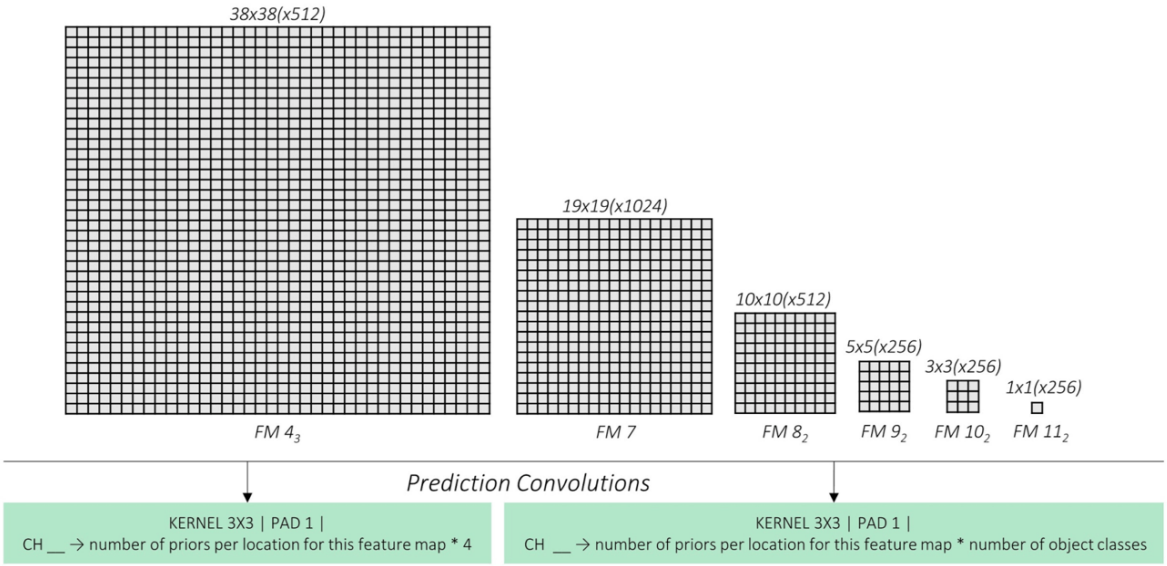

Figure 6: **Prediction layer:** Features maps are passed into the prediction convolutions in order to locate and identify objects in these feature maps.

Figure 7: **Training vs. Test:** This figure shows the training versus test multibox loss over time of 100 epochs.

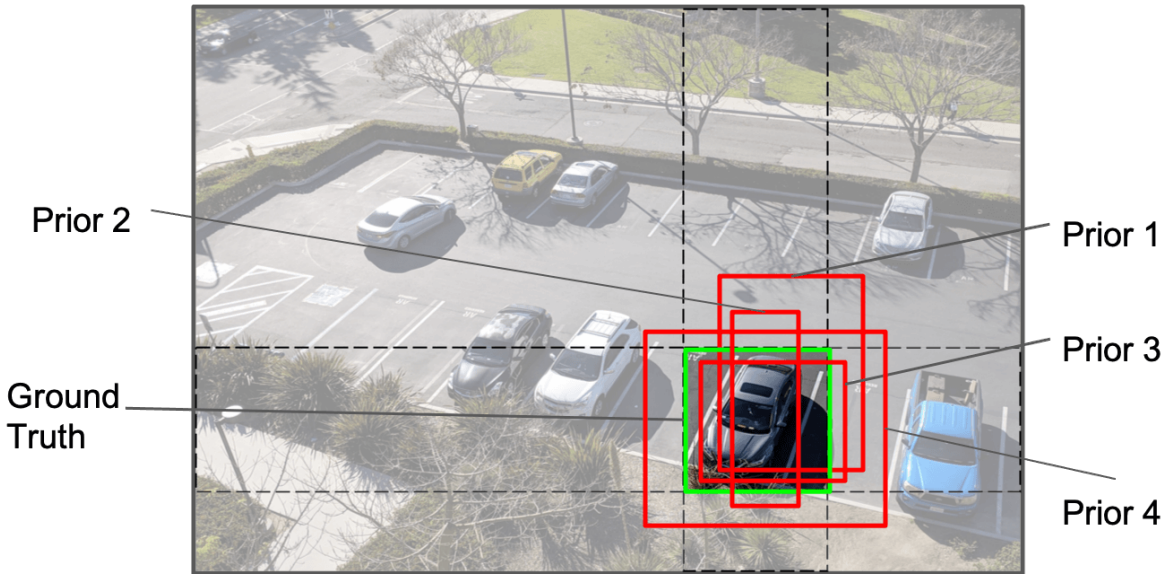

Figure 8: **Matching to simplify regression:** Example of multiple anchor boxes (prior boxes) for a potential matching ground truth.