Object Detection with Deep Learning

In this section, we will review Object Detection using CNNs. We will focus on the output of the CNN and assume a sliding window is used. In the next section, we will discuss more effective methods that do not rely on sliding windows, but the output of the CNN is similar in any case. We will discuss the prediction step as well as give an overview of training.

This section will provide an overview, as there are many details in object detection depending on the application. In addition, evaluating Object Detectors is complex, so we will leave some references for further reading.

Object Detection Prediction

We can use a sliding window and a CNN to detect an object. We classify the image as a background or part of another class. We typically use the popular CNN Architectures pre-trained on ImageNet as shown here:

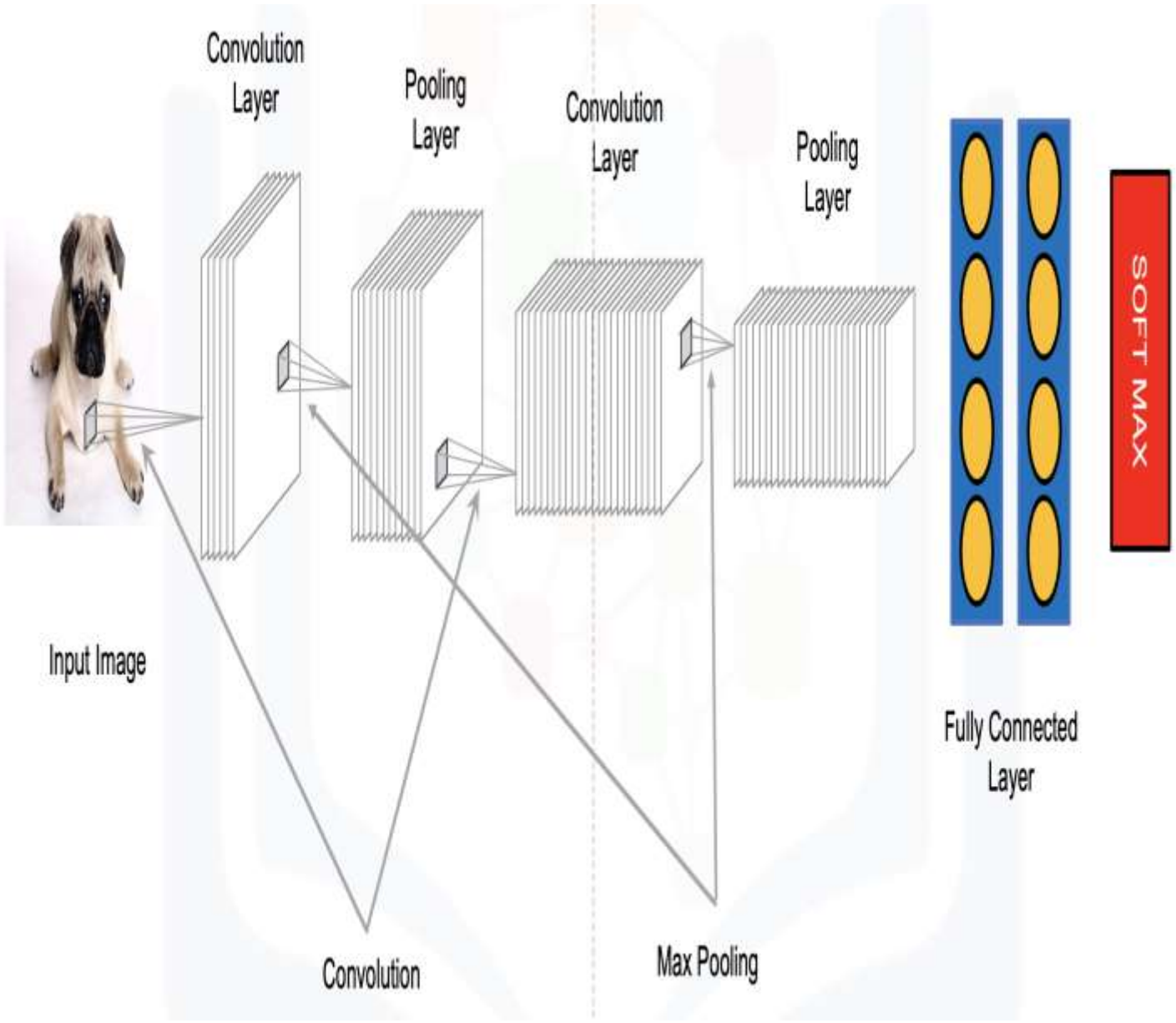

Figure 1: CNN Architectures with SoftMax on the output layer. Where the number of neurons in the output softmax layer is equal to the number of classes. In many CNNs used for object detection, we add four neurons to the output layer to predict the bounding box, as shown in the following figure.

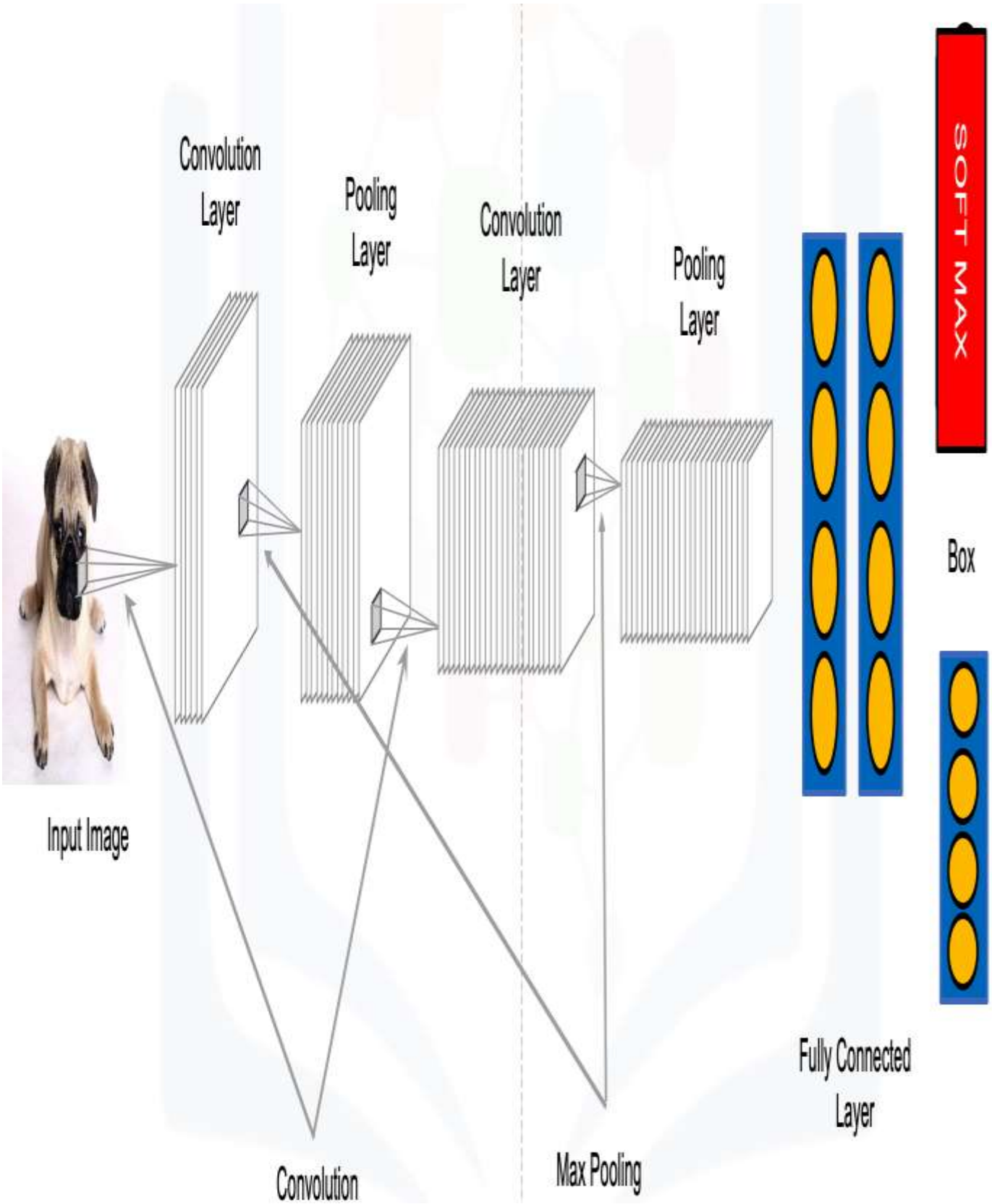

Figure 2: CNN Architectures with the SoftMax and box prediction on the output layer.

Each neuron outputs a different component of the box. This relationship is shown in Figure 3, with the corresponding bounding box around the object. We colour y hat and x hat to distinguish between the object class prediction y and x. To avoid confusion, unless explicitly referring to the coordinates system, we will use the term "box hat" to represent the output of these neurons.

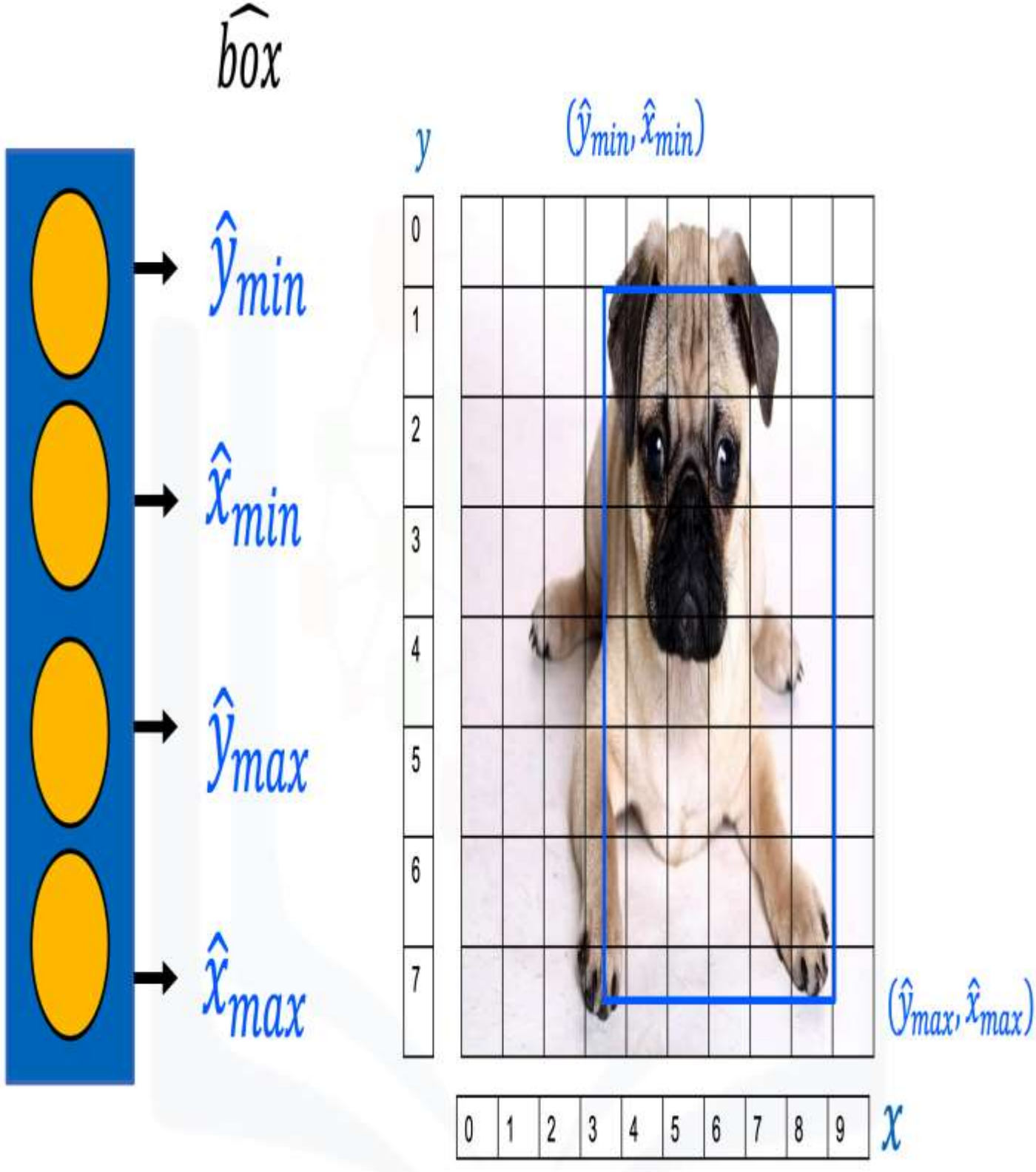
$\widehat{box}$
$\hat{y}_{min}$
$\hat{x}_{min}$
$\hat{y}_{max}$
$\hat{x}_{max}$
y
$(\hat{y}_{min}, \hat{x}_{min})$
0
1
2
3
4
5
6
7
$(\hat{y}_{max}, \hat{x}_{max})$
0 1 2 3 4 5 6 7 8 9
x

Figure 3. Relationship with the corresponding bounding box and neurons, we use oversized pixel indexes for clarity.

Unlike classification, the output values of the neuron take on real numbers. Some possible values are shown in figure 4.

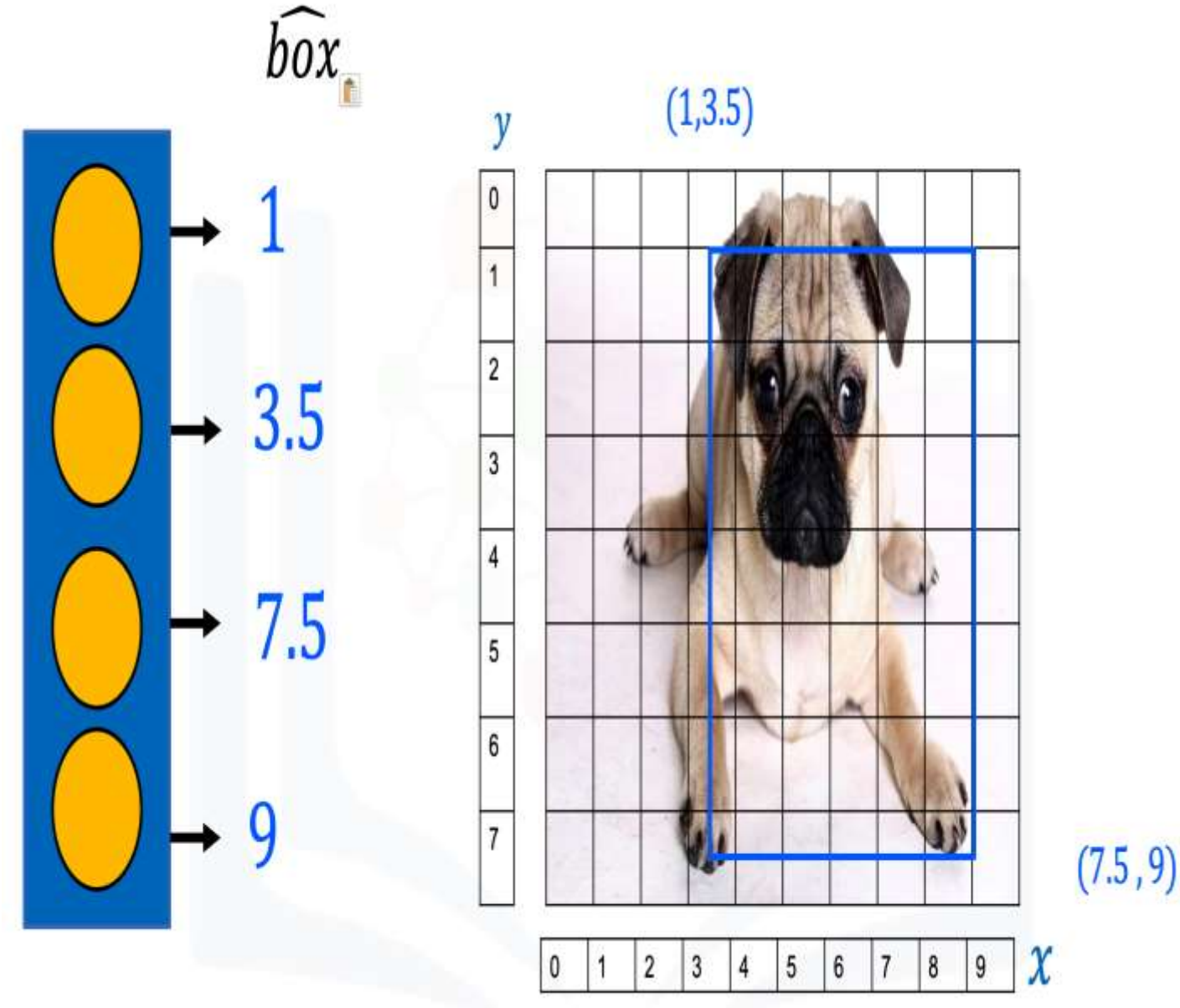

Figure 4. Real numbers output values of box neurons

To predict the class of the bounding box, we use the softmax layers as shown in Figure 6. We have an output for each class, in this case: dog, cat, bird, and background. We can use the probability or the output of the activation.

dog	z_0
cat	z_1
bird	z_2
background	z_3

Softmax

$\widehat{box}$

Figure 5. softmax layers used to predict the class of bounding box for three classes

Consider the example in Figure 6 - we have the bounding box in red. To find the class of the bounding box, we use the output of the softmax layer. Examining the probabilistic output of the softmax layer, we have four outputs: 0.7 for "dog", 0.1 for "cat", 0.05 for "bird", and 0.15 for background. Hence, we select "dog" as the classification, since the softmax has the highest output for that class.

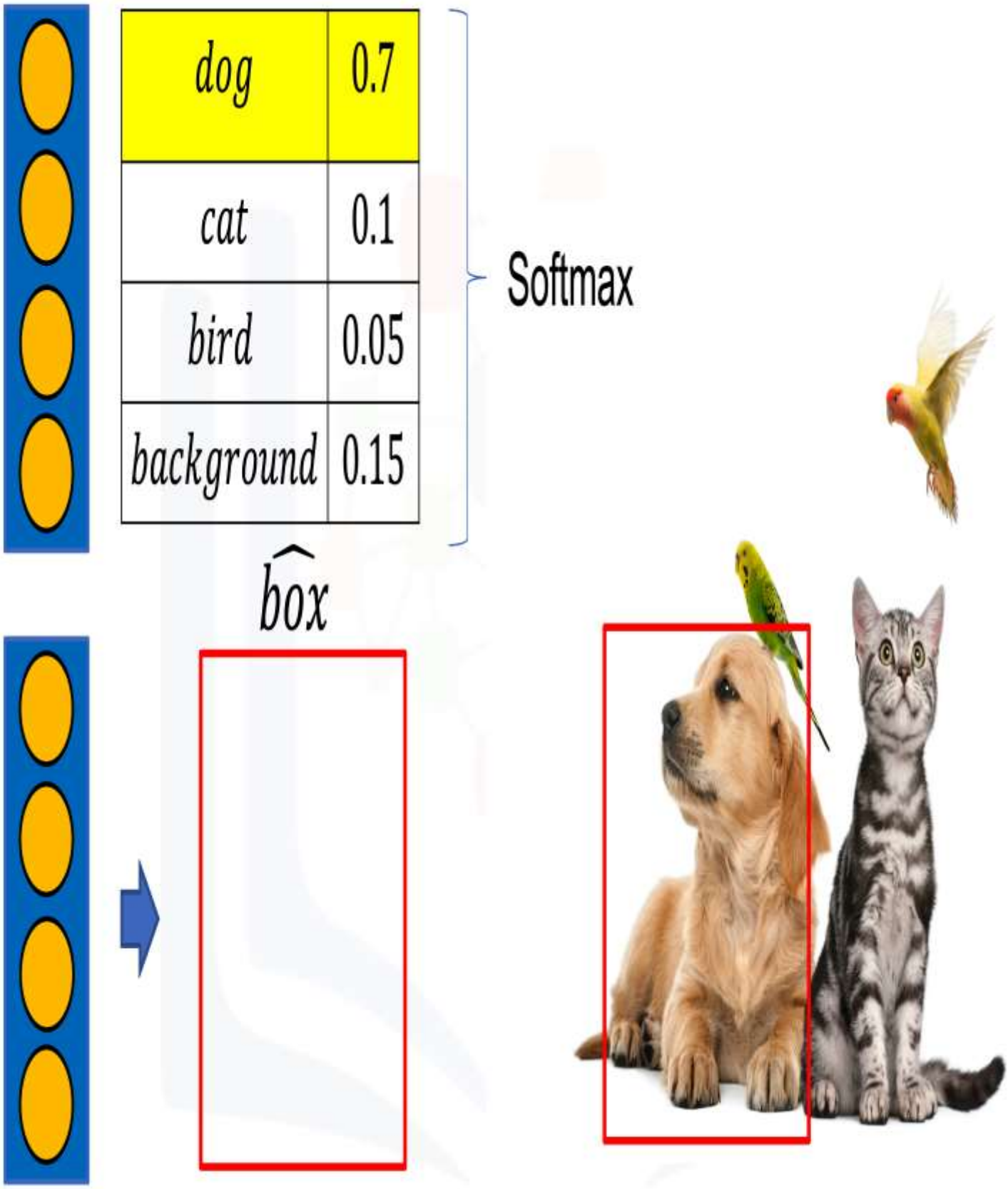
dog
0.7
cat
0.1
bird
0.05
background
0.15
Softmax
box

Figure 6. example of softmax layers used to predict the class of bounding box for three classes

Training for Object Detection

Training in Object Detection has two objectives: we have to determine the learnable parameters for the box and we have to determine the bounding boxes class. In order to determine the learnable parameters for the bounding box, we use the L2 or squared loss. This is used to find the difference between real value predictions. The L2 Loss Function calculates squared differences between the actual box value and the predicted box, as shown in figure 7, where we have the box and the L2 Loss for each coordinate of the box.

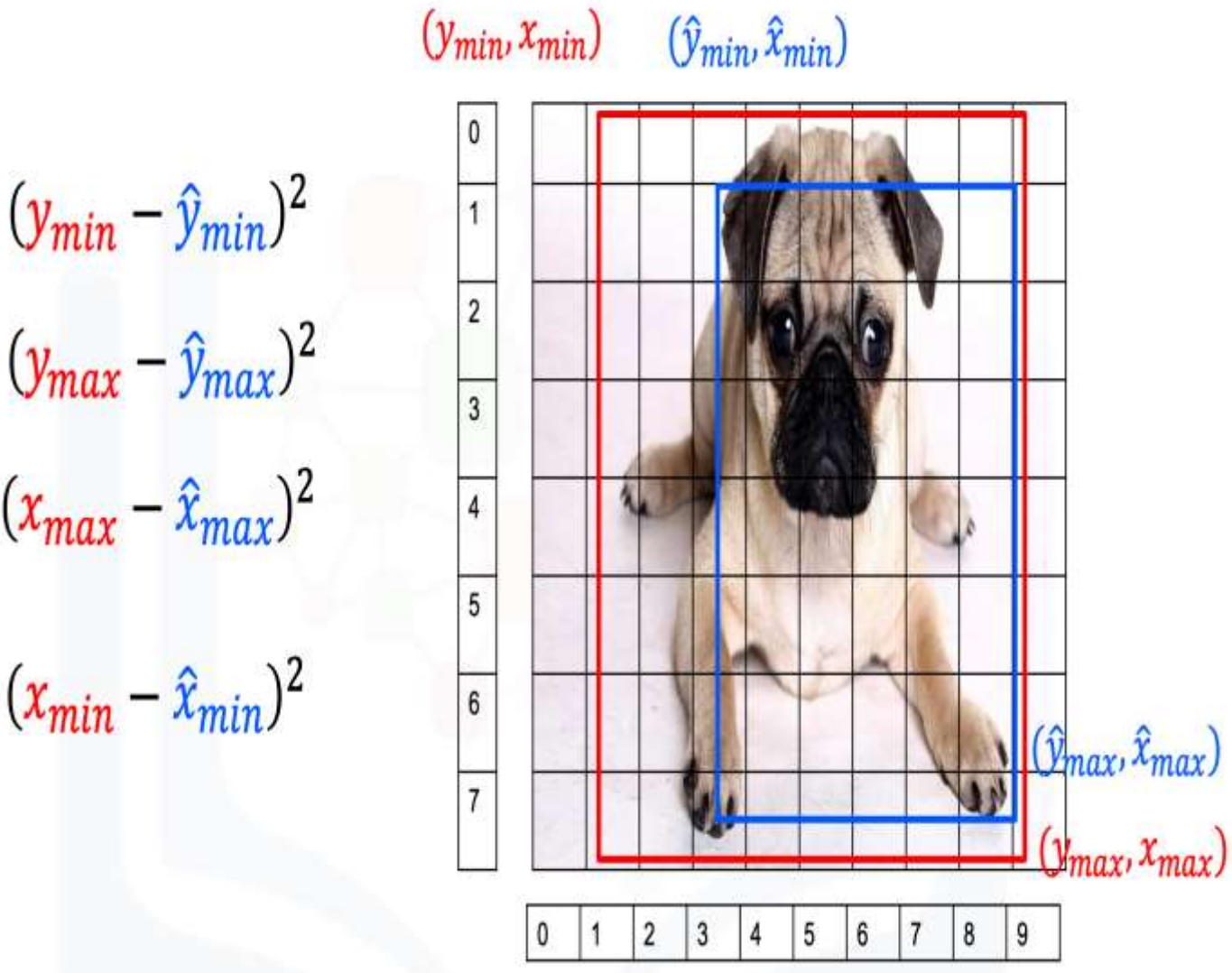

Figure 7. the L2 Loss Function calculates squared differences between the actual box value and the predicted box.

The loss for each box is given by:

$$\left\|box - \widehat{box}\right\|^2 = (y_{min} - \hat{y}_{min})^2 + (y_{max} - \hat{y}_{max})^2 + (x_{max} - \hat{x}_{max})^2 + (x_{min} - \hat{x}_{min})^2$$

Finally, to determine the classification, we combine the L2 cost with the cross-entropy loss in a Weighted Sum. This is called the Multitask Loss, which we use to determine the cost. We then use this to determine all the parameters using gradient descent to minimize the cost.

Multitask Loss = L2 Loss + Cross entropy

Types of Object Detection

Sliding window techniques are slow. Fortunatly, there are two major types of object detection that speed up the process. Region-based object detection breaks up the image into regions and performs a prediction, while Single-Stage uses the entire image.

Region-Based Convolutional Neural Network (R-CNN) are usually more accurate but slower; they include R-CNN, fast RCNN and Faster RCNN.

Single-Stage methods are faster but less accurate and include techniques like Single Shot Detection (SSD) and You Only Look Once (YOLO) .

In the following two labs, you will use Faster RCNN for prediction. You will train an SSD model, even though SSD is considerably faster than other methods, it will still take a long time to train. Therefore we will train most of the model for you, and you will train the model for the last few iterations.

References and Further Reading

1. Jaccard Index
2. Evolution Of Object Detection Networks
3. Girshick, Ross. "Fast R-CNN." *Proceedings of the IEEE international conference on computer vision*. 2015.
4. Ren, Shaoqing, et al. "Faster R-CNN: Towards real-time object detection with region proposal networks." 2015