# Chapter 8

## Telling Things Apart: Image Segmentation

The task of image classification teaches a model to recognize whether a particular object exists in an image. Yet in many real-world applications, the question is not only what objects are present but also where they are located. This is the domain of image segmentation, a dense prediction task where every pixel in an image is assigned to a specific class. Segmentation powers technologies such as self-driving cars, medical image analysis, and content-based image retrieval.

Segmentation is often divided into two categories. Semantic segmentation assigns the same label to all objects of a given class—for example, all people in a frame are marked with a single "person" label. Instance segmentation goes further, separating individual objects even if they belong to the same class. Figure 1 contrasts these two approaches. In this discussion, the focus remains on semantic segmentation.


*Figure 1 Semantic segmentation versus Instance segmentation*

### Understanding the Data

The dataset chosen for this journey is PASCAL VOC 2012, a benchmark collection of urban and domestic scenes. Each entry consists of a standard input image paired with an annotated image where every pixel is assigned a color from a predefined palette. These colors map to object categories such as aeroplane, bicycle, cat, dining table, or background. Table 1 lists all 22 categories, while Figure 2 displays representative samples for each. Figure 3 shows a detailed example containing both a chair and a dog, each marked with a unique color.

*Table 1 Different classes and their respective labels in the PASCAL VOC 2012 data set*

| Class | Assigned Label | Class | Assigned Label |
|---|---|---|---|
| Background | 0 | Dining Table | 11 |
| Aeroplane | 1 | Dog | 12 |
| Bicycle | 2 | Horse | 13 |
| Bird | 3 | Motorbike | 14 |
| Boat | 4 | Person | 15 |
| Bottle | 5 | Potted plant | 16 |
| Bus | 6 | Sheep | 17 |
| Car | 7 | Sofa | 18 |
| Cat | 8 | Train | 19 |
| Chair | 9 | Tv/monitor | 20 |
| Cow | 10 | Boundaries/unknown object | 255 |

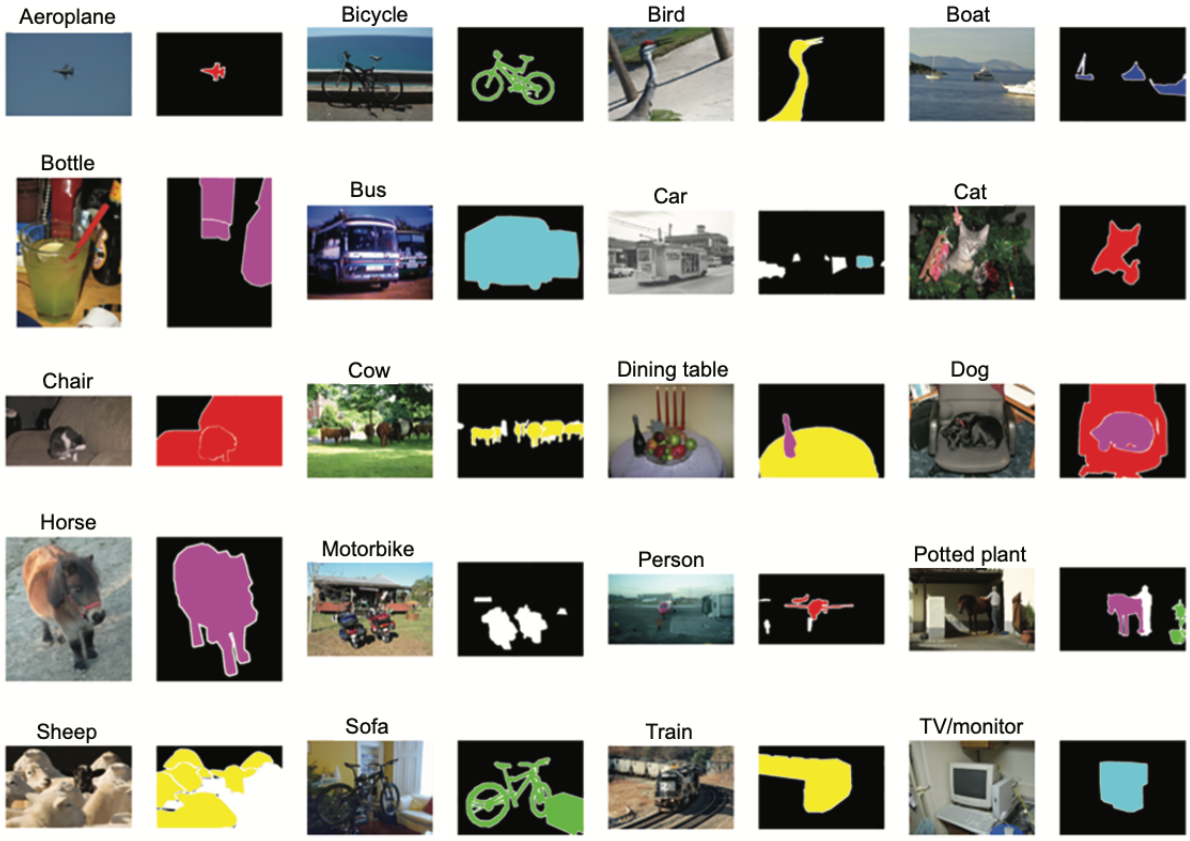


*Figure 2 Samples from the PASCAL VOC 2012 data set*

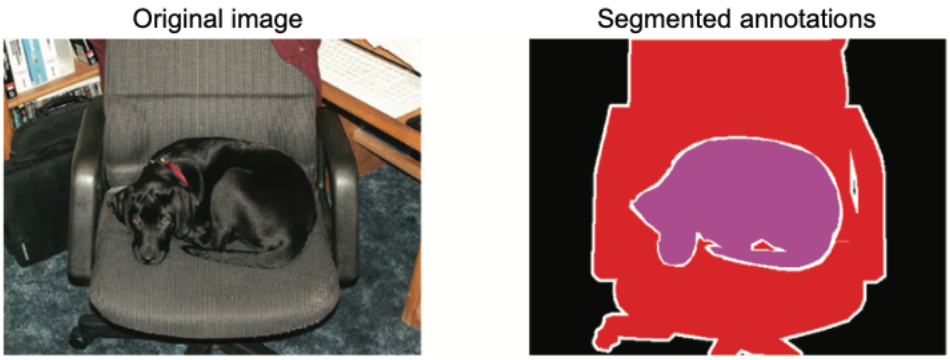


*Figure 3 An original input image in image segmentation and the corresponding target annotated images*

Unlike classification datasets, segmentation targets are not stored as ordinary images. Instead, they are palettized images: arrays of indices that reference a color palette. Special functions are required to reconstruct target images before feeding them to the model.

**Building the Data Pipeline**

A reliable data pipeline is essential for large-scale training. Using TensorFlow's tf.data API, the process begins by generating filenames for train, validation, and test sets. Each pair of original and segmented images is then loaded. Input images are read normally, while targets must be loaded as palettized arrays to preserve class indices.

Because dataset images have varying dimensions, resizing is mandatory. Resizing uses bilinear interpolation for input images but nearest-neighbor interpolation for targets, ensuring segmentation masks remain integer-based (Figure 4). Additional augmentation—such as random flips, hue changes, brightness, and contrast adjustments—enriches the training set.

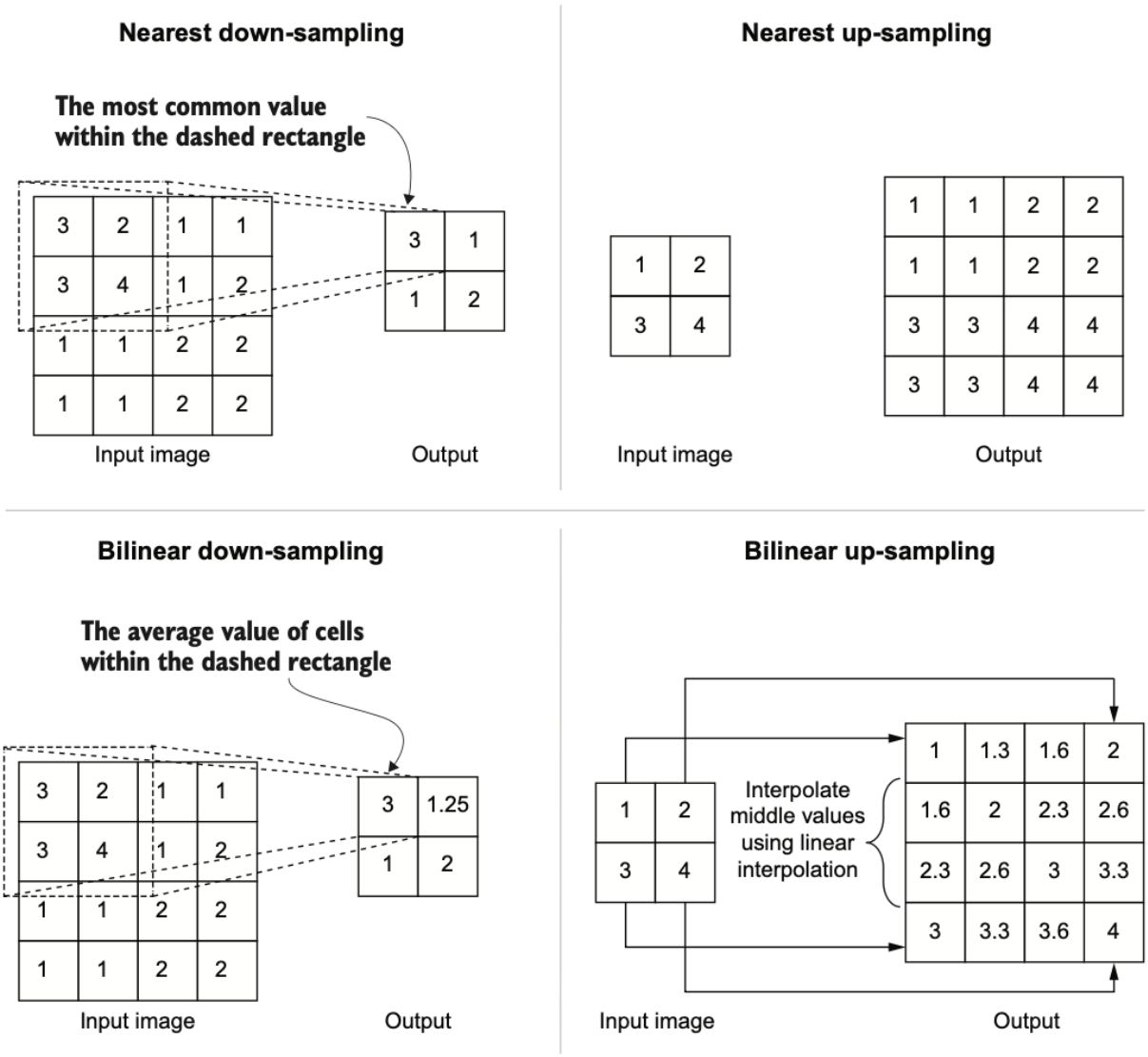


*Figure 4 Nearest interpolation and bilinear interpolation for both up-sampling and down-sampling tasks*

To streamline training, further steps include shuffling, batching, and repeating data across epochs. Finally, unnecessary singleton dimensions in the target tensors are removed with tf.squeeze(). The complete pipeline also integrates caching and prefetching. Caching stores data in memory after the first epoch, while prefetching overlaps data loading with training to avoid idle time. Figure 5 highlights the performance gains of prefetching over sequential execution. The outcome is three separate pipelines for training, validation, and testing, each with distinct characteristics.

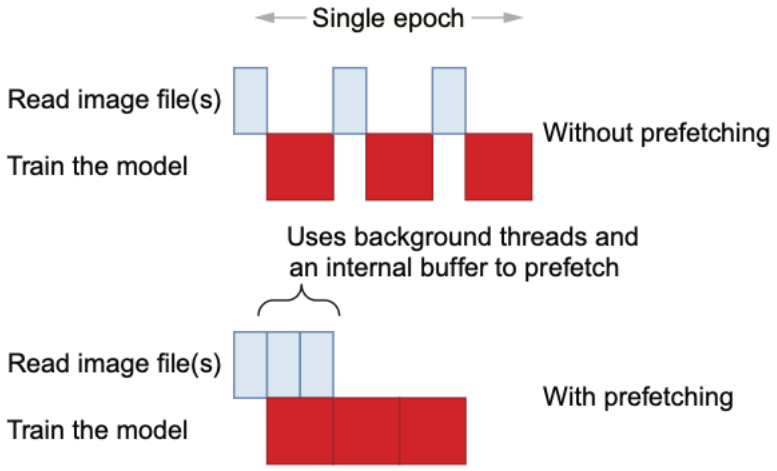


*Figure 5 Sequential execution versus pre-fetching-based execution in model training*

**DeepLab v3: Segmenting with Pretrained Networks**

With the data pipeline complete, attention turns to the segmentation model. Among state-of-the-art designs, DeepLab v3 stands out. It combines a ResNet-50 backbone pretrained on ImageNet with architectural innovations like atrous convolution and a pyramidal aggregation module. Unlike encoder-decoder models such as U-Net, DeepLab v3 avoids a heavy decoder stage and instead relies on atrous spatial pyramid pooling (ASPP) to capture multi-scale context.

Atrous convolution, or dilated convolution, increases the receptive field of convolution layers by inserting "holes" between kernel elements. This expands spatial coverage without increasing the number of parameters. Figure 6 shows how dilation rates enlarge the receptive field while maintaining efficiency. This technique directly addresses the common problem of segmentation networks producing overly small outputs due to repeated striding and pooling.

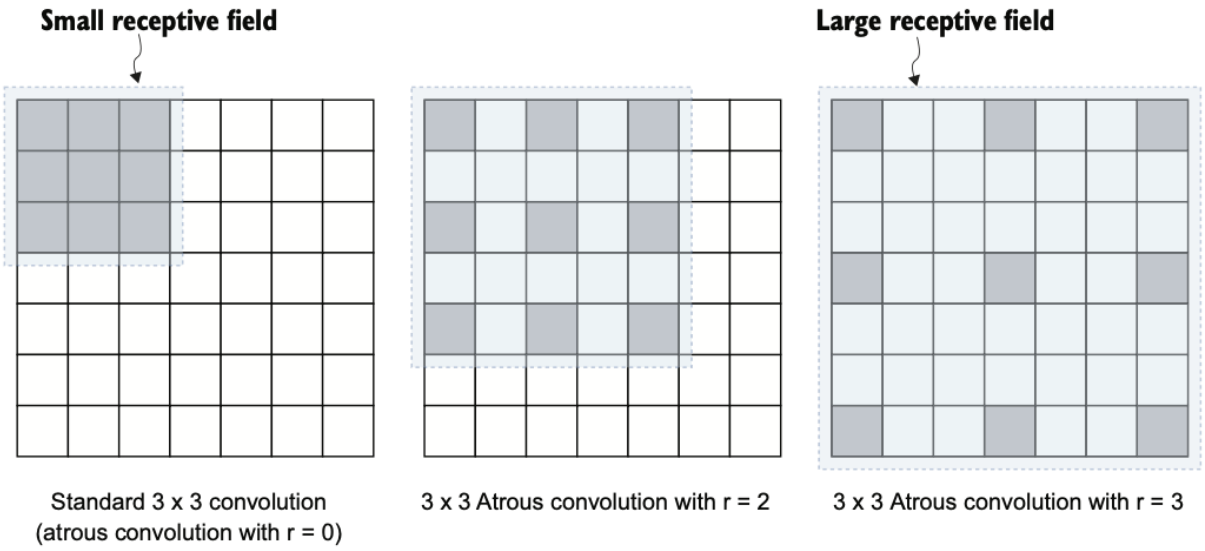


*Figure 6 Atrous convolution compared to standard convolution*