8 Tips For Gathering and Labeling Datasets for Training Object Detection Models

Building a quality dataset is one of the most important steps in training an object detection model. The first step for training any machine learning model is to build a dataset. For object detection models, this means collecting and labeling hundreds to thousands of images of the objects you'd like to detect. Curating a dataset is often the hardest and most important part of training a model, and there are many details that need to be considered. In this guide, we introduce the process of dataset creation and share eight tips for effectively gathering and labeling images to train an object detection model. **Note: These guidelines apply equally to any type of images, including thermal (infrared) images, not just standard color photographs.**

Tip 1. Use images similar to what will be seen in the actual application

When training a computer vision model, it's best to use images that are similar to what will be seen by the camera in the end application. They should be taken from the same angle, perspective, range of distances, lighting, and environments as what the model will encounter when it is actually deployed. For best results, the images should be captured by the same camera model that will be used in the application (this includes using the same type of sensor—for example, if the application's camera is a thermal image, capture your training images with a thermal camera as well).

Using similar images helps the model learn key features relevant to that setting, such as lighting conditions, edges, depth, and other complex features. It's important to capture similar images because it helps the model learn the features from those environments and correlate them to the objects you want to detect.

When setting up the camera to capture the images, imitate the position the camera will be in for the actual application. For example, if the model will run on a surveillance camera, set the camera up so it has a similar view as the surveillance camera would. If it will run on a car-mounted camera, set the camera up so it is at a similar level and direction as it would be on the car. If the application is going to see the objects from many different perspectives and backgrounds, the camera will need to be set up in a variety of ways so that it can capture multiple views of the objects.

Tip 2. Provide examples of objects at various rotations, distances, and perspectives

To be able to detect objects, the model must know what they look like from all angles and perspectives. Also, the model will learn the approximate size of the object: if the images only show examples of the objects up close, the model won't do well at detecting them when they're far away. **It's important to provide images showing the objects at various perspectives, rotations, and distances. Using images like this will increase the robustness of the model.**

One approach to capturing objects from various angles is to methodically move the objects slightly in different directions or orientations across different images. Another approach is to capture enough random images of the objects so that all orientations are represented. **The ideal dataset will show each object at every angle, perspective, and distance that will be seen in the actual application. The more rotations, distances, and perspectives that can be captured, the more this will help the model understand the object and improve its ability to detect it.**

It's also useful to include a variety of backgrounds, settings, and other objects in the training dataset. This adds more visual data that helps the model understand what the objects of interest *don't* look like, which ultimately helps reduce false positives.

Tip 3: Start with a Large and Diverse Image Dataset (≈5,000+ Images per Class)

When training an object detection model, **the more images per class, the better.** A good rule of thumb is to aim for around **5,000 training images for each object category** as a starting point.

This volume is typically enough for the model to learn the distinguishing features of each object class. In general, increasing the number of images in the dataset will improve the model's accuracy. In fact, high-performance production models often use tens of thousands (or even hundreds of thousands) of images per class. Keep in mind that newer training pipelines – for example, the latest YOLO models – include advanced data augmentation (automatic image cropping, lighting variation, blurring, flipping, etc.) which effectively **multiplies** the dataset size. These techniques can reduce the number of original images needed, but you still want a **large, varied base dataset** to begin with for best results.

Ensure Variety in Your Images: It's not just about quantity; diversity of the training images is equally important. Each object class should be captured across a wide range of scenarios so the model learns to recognize the object under different conditions. Here are some guidelines to diversify your dataset:

- **Multiple Angles and Viewpoints:** Include images taken from different camera angles – for example, top-down views, angled views, and side/profile views. This way, the model isn't surprised by a new viewpoint in real-world footage.
- **Various Distances and Scales:** Gather images at **different distances** from the camera (close-ups, medium range, and far-away shots). This ensures the model can detect both large, near objects and small, distant ones.
- **Different Poses or Orientations:** If the objects can appear in different poses or states, cover those as well. For instance, if your class is "person," collect images of people standing, sitting, lying down, partially hidden behind other objects, or cut off at the edge of the frame. If your class is "animal," include animals in various postures (running, grazing, resting) and from different sides. Covering a range of **orientations and occlusions** will teach the model to handle nuances like partially visible targets.
- **Varying Backgrounds and Environments:** Incorporate a mix of backgrounds, locations, and lighting conditions for each class. An object should be shown in different environments (e.g. an animal in grass, in forest, on a hill; a person in open field, near buildings, in sunlight, at night, etc.). This variety helps the model distinguish the object from **distracting backgrounds** and generalize to new scenes it hasn't seen before.
- **Images with Multiple Objects (and None):** Include some images that contain **multiple objects** from your target classes in one frame (e.g. several people or animals in the same image), as well as images with **zero** target objects. Having 0–3 objects per image on average is a realistic scenario. Crucially, also add **background-only images** (images with no target objects present) to your dataset – about **10–20% of the images** can be empty backgrounds. These "no-object" images help the model learn to **ignore blank scenery** and reduce false alarms by understanding when **no object is present**. For example, if you have 10,000 labeled-object images, you might include an extra 1,000–2,000 purely empty scenes.

By following these guidelines, you'll build a **rich and balanced dataset** that closely mirrors real-world conditions. Apply the same principles of quantity and diversity to **every object class** in your project. For instance, if you have multiple classes (say people, vehicles, animals, etc.), try to collect a few thousand images for each class with similar variety (different angles, distances, contexts for each category). Even for **"distractor" classes** or common non-target objects (things that aren't targets but might appear in the camera, like rocks, trees, or other background elements),

include a reasonable number of examples with plenty of variation. This way, the model learns to tell these apart from true targets.

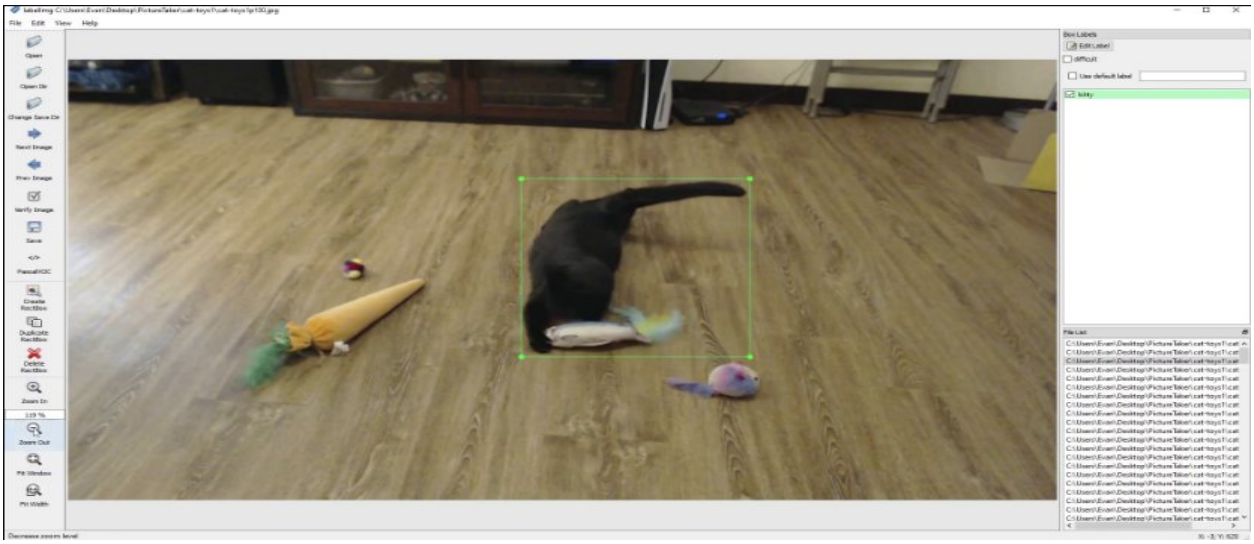
In summary, starting with *around 5,000 diverse images per class* – covering numerous poses, angles, distances, and environments – provides a solid foundation. Such a dataset, possibly boosted by augmentation techniques, will help your object detection model achieve good accuracy and robust performance across many real-world scenarios.

Tip 4. Don’t use pictures that are nearly identical

We discarded the image on the right as it doesn’t add much value to the dataset, since it is nearly identical to the one on the left. Make sure there’s some visual difference between each image used in the dataset. Having near-identical images doesn’t provide additional data to the model and thus isn’t useful for training. An identical or almost identical image will immediately become redundant, as the model will have already accounted for that exact scenario and it won’t add to its knowledge.

If a dataset was created by extracting frames from a video, it’s common to have near-identical images because the scene doesn’t change much from one frame to the next. This can also occur when taking repetitive images of a mostly unchanging environment. When preparing the training data, it’s a good idea to look through the images and remove ones that are too similar.

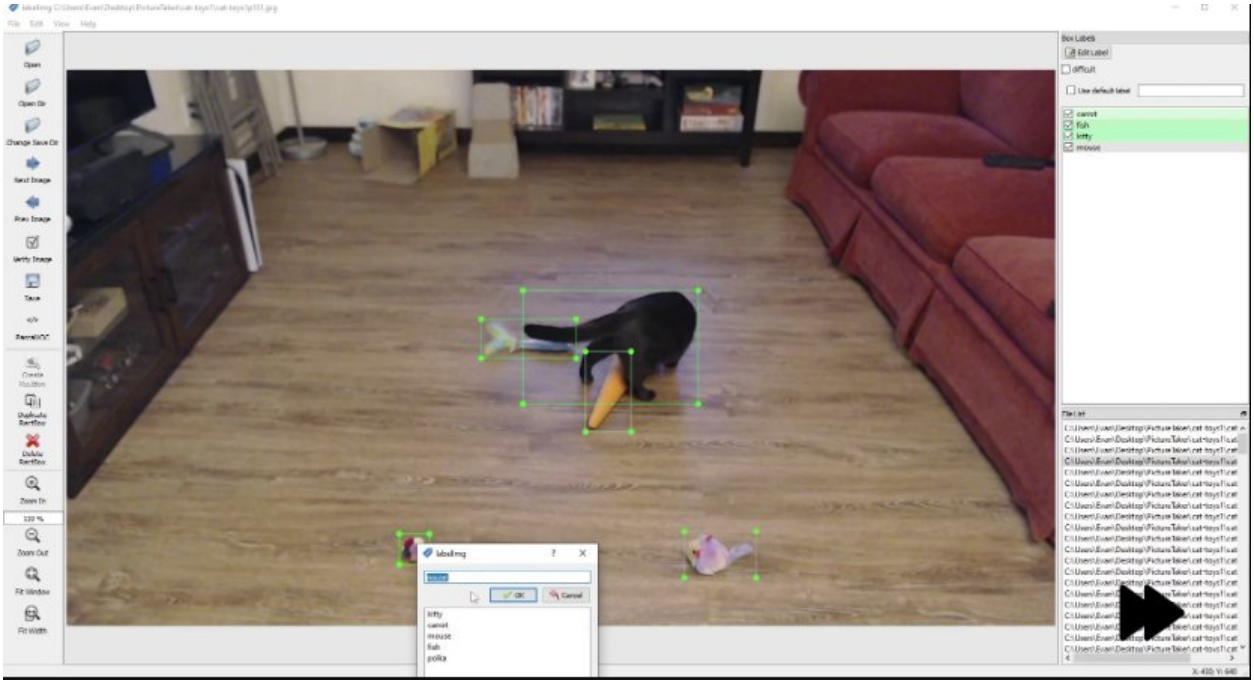
Tip 5. Labeling – Include the full object inside the bounding box



Object detection works best with objects that have clear and well-defined boundaries, like a cat. When labeling these object, be sure to include the full object inside the bounding box. The bounding box doesn't have to be perfect, but it's better to make the box slightly too big rather than risk missing parts of the object. Make use of the zoom tool to get a more precise bounding box around the object if needed.

Certain shapes of objects (such as long, thin objects placed diagonally) could lead to more background being present in the bounding box. This is fine, but make sure you get many different angles of the object and include examples with different backgrounds as well. If objects are slightly obscured, just draw a box around the visible part of the object. It's also okay for bounding boxes to overlap, so don't worry if the bounding boxes intersect each other.

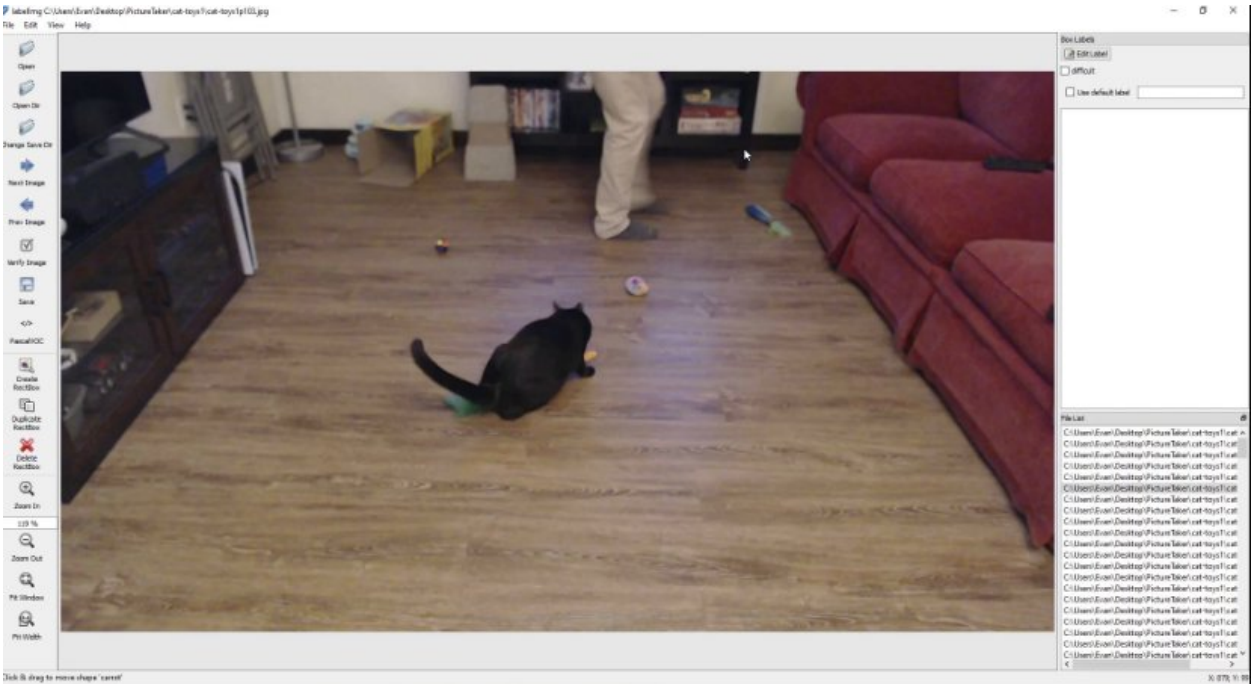
Tip 6. Labeling – Ask yourself, "where would I want the model to predict this bounding box?"



To determine where to draw the label box, it can be helpful to ask "where would I want the model to predict these bounding boxes?" Sometimes, as the person labeling the image, it can be confusing to decide where to draw an object's bounding box label. This often happens in unusual cases where an object is partially obscured or difficult to see in the image. In these cases, ask yourself where you would want the bounding box to be predicted if the model saw this image in the actual application. Wherever you would want the box to be predicted, that's where you should draw the bounding box label. Placing the bounding box around this region will help the model more accurately learn about the object, even if it is obscured.

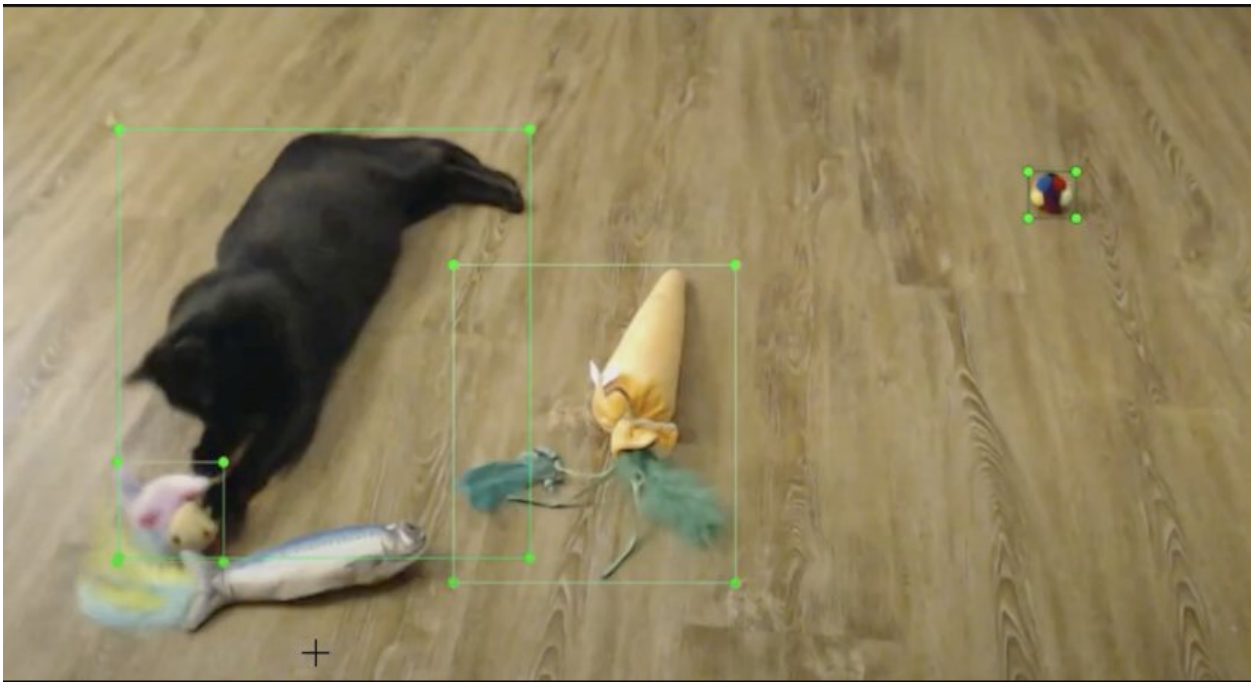
Tip 7. Labeling – Don’t use confusing or overcrowded images

It’s fine to have overlapping bounding boxes in an image; however, using overcrowded or very busy images will cause the model to have a hard time predicting bounding boxes correctly. As seen in the example above, it would be very challenging to separate and label each person in the large crowd. It’s best to use images where there are clear boundaries around each object.



If you’re confused about how to label certain images (due to a restricted view of the object, for example) it is okay to exclude that image and use a different one that has a clearer view of the object. For example, in the image above, there are a couple of ways the carrot could be labeled: both ends of the carrot could each have their own bounding box, or the full carrot could be labeled in a single bounding box. In this case, it’s acceptable to omit the image from the dataset due to it being too confusing. If there are confusing or contradictory training examples in the dataset, the model won’t be able to correlate features to objects as accurately.

Tip 8. Start with basic images first, then add more difficult images



Each toy and the cat are unobstructed and well-lit, providing an “easy” example for the model to learn from. Object detection models learn best from clear, obvious examples. It can be tempting to use “difficult” images to train the model – where objects are obscured, in dim lighting, blurry, or in other challenging visual conditions. However, it’s best to start with “easy” images where the objects are obvious, clear, and have good lighting. Train the model on the easy images first and confirm it works in these basic conditions. Then, add more difficult images (for example, images with multiple overlapping objects, dim lighting, or motion blur) and retrain the model to improve its performance in those conditions. This incremental approach makes it easier to identify areas where the model struggles and address them one by one. ***(This same principle applies to thermal image datasets as well: start with clear thermal images of your objects before introducing more challenging thermal scenes.)***