

[Blog](#)

How to Train a YOLOv11 Object Detection Model on a Custom Dataset

How to Train a YOLOv11 Model with Custom Data

The Roboflow logo, which consists of the word "roboflow" in a lowercase, sans-serif font.

James Gallagher

Published Oct 8, 2024 • 7 min read

Launched on September 27, 2024, YOLOv11 (referred to by the model author Ultralytics as YOLO11) is a computer vision model that you can use for a wide variety of tasks, from object detection to segmentation to classification.

According to Ultralytics, “YOLO11m achieves a higher mean Average Precision (mAP) score on the COCO dataset while using 22% fewer parameters than YOLOv8m.” With fewer parameters, the model can run faster, thereby making the model more attractive for use in real-time computer vision applications.

YOLOv11: How to Train for Object Detection on a Custom Dataset | Step...



In this guide, we are going to walk through how to train a YOLOv11 object detection model with a custom dataset. We will:

1. Create a custom dataset with labeled images
2. Export the dataset for use in model training
3. Train the model using the a Colab training notebook
4. Run inference with the model

Here is an example of predictions from a model trained to identify shipping containers:



We have a [YOLOv11 Colab notebook](#) for you to use as you follow this tutorial.

Without further ado, let's get started!

Step #1: Create a Roboflow Project

To get started, we need to prepare a labeled dataset in the format required by YOLOv11. We can do so with Roboflow.

Create a free [Roboflow account](#). Once you have created an account, click "Create New Project" on the Roboflow dashboard. You will be taken to a page where you can configure your project.

Set a name for your project. Choose the "Object Detection" dataset type:

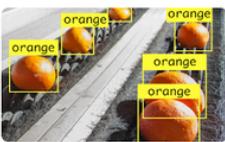
Let's create your project.

Roboflow Universe Projects > [New Public Project](#)

Project Name

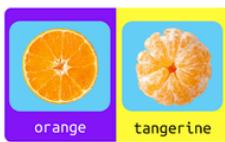
 License [?](#)
[CC BY 4.0](#)

Annotation Group [?](#)
E.g., 'dogs' or 'cars' or 'words'.

Project Type

Object Detection

Identify objects and their positions with bounding boxes.



Classification

Assign labels to the entire image.

Classification Type
 Single-Label
 Multi-Label

Best For

- Filtering
- Content Moderation



Instance Segmentation

Detect multiple objects and their actual shape.



Keypoint Detection

Identify keypoints ("skeletons") on subjects.

Show More

Cancel

Create Public Project

Then, click "Create Project" to create your project.

Step #2: Upload and Annotate Images

Next, you need to upload data for use in your project. You can upload labeled data to review or convert to the YOLO PyTorch TXT format, and/or raw images to annotate in your project.

In this guide, we will train a model that detects shipping containers. We have an [open shipping container dataset](#) on Roboflow Universe that you can use. Or, you can use your own data.

Drag and drop your raw or annotated images into the Upload box:

↑ Upload

Batch Name: Tags: [?](#)

Search or add tags for images...

**Drag and drop to upload, or:**

Supported Formats

**Images**

in .jpg, .png, .bmp, .webp

**Annotations**

in 26 formats ↗

**Videos**

in .mov, .mp4

*Max size of 20MB and 16,384 pixels per dimension.

When you drag in data, the data will be processed in your browser. Click "Save and Continue" to upload your data into the Roboflow platform.

↑ Upload

[Save and Continue →](#)Batch Name: Tags: [?](#)

Search or add tags for images...

[All Images 198](#)

Annotated 123

Not Annotated 75

Drag and drop images, annotations, and videos.

in .jpg, .png, .bmp, .webp in 26 formats ↗ in .mov, .mp4

*Max size of 20MB and 16,384 pixels per dimension.



Tips-and-tricks-on-how-to-park-a-yard-jockey-_mp4-85.jpg



Tips-and-tricks-on-how-to-park-a-yard-jockey-_mp4-232.jpg



Tips-and-tricks-on-how-to-park-a-yard-jockey-_mp4-16.jpg



Tips-and-tricks-on-how-to-park-a-yard-jockey-_mp4-208.jpg



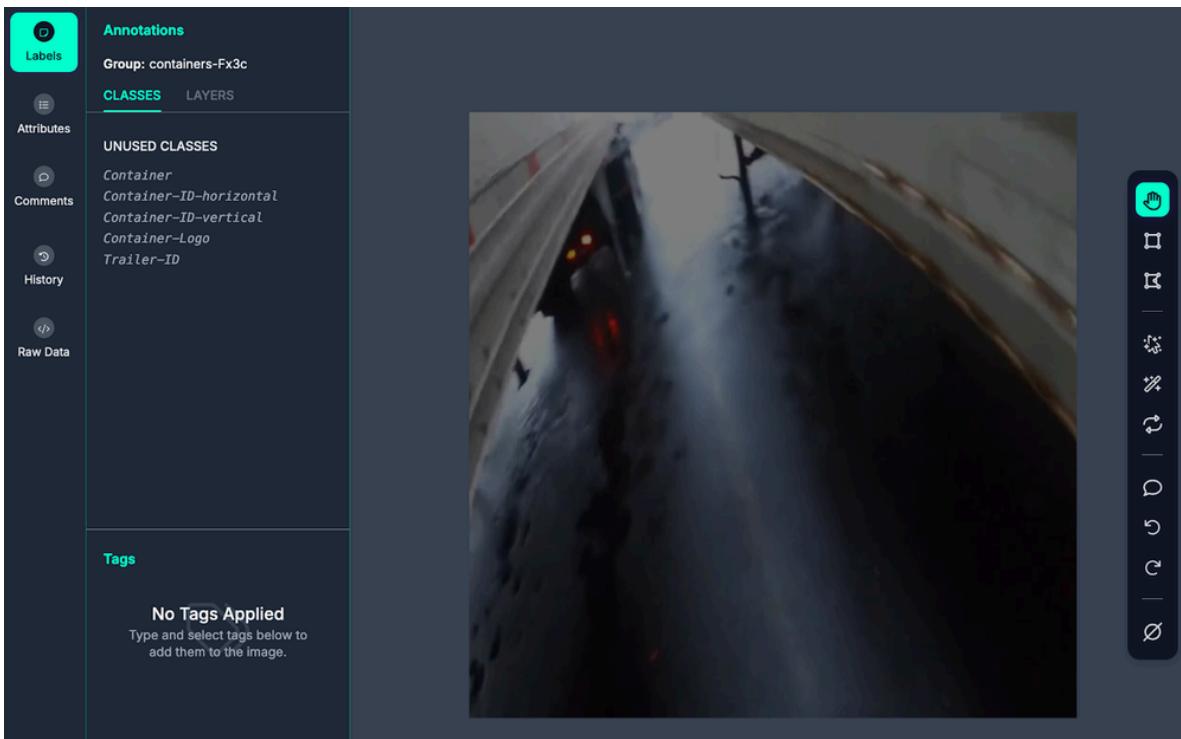
Tips-and-tricks-on-how-to-park-a-yard-jockey-_mp4-225.jpg



Tips-and-tricks-on-how-to-park-a-yard-jockey-_mp4-227.jpg

With your data in Roboflow, you can annotate images.

To annotate an image, click "Annotate" in the left sidebar. Click on an image to start annotating. You will be taken to the [Roboflow Annotate](#) interface in which you can label your data:

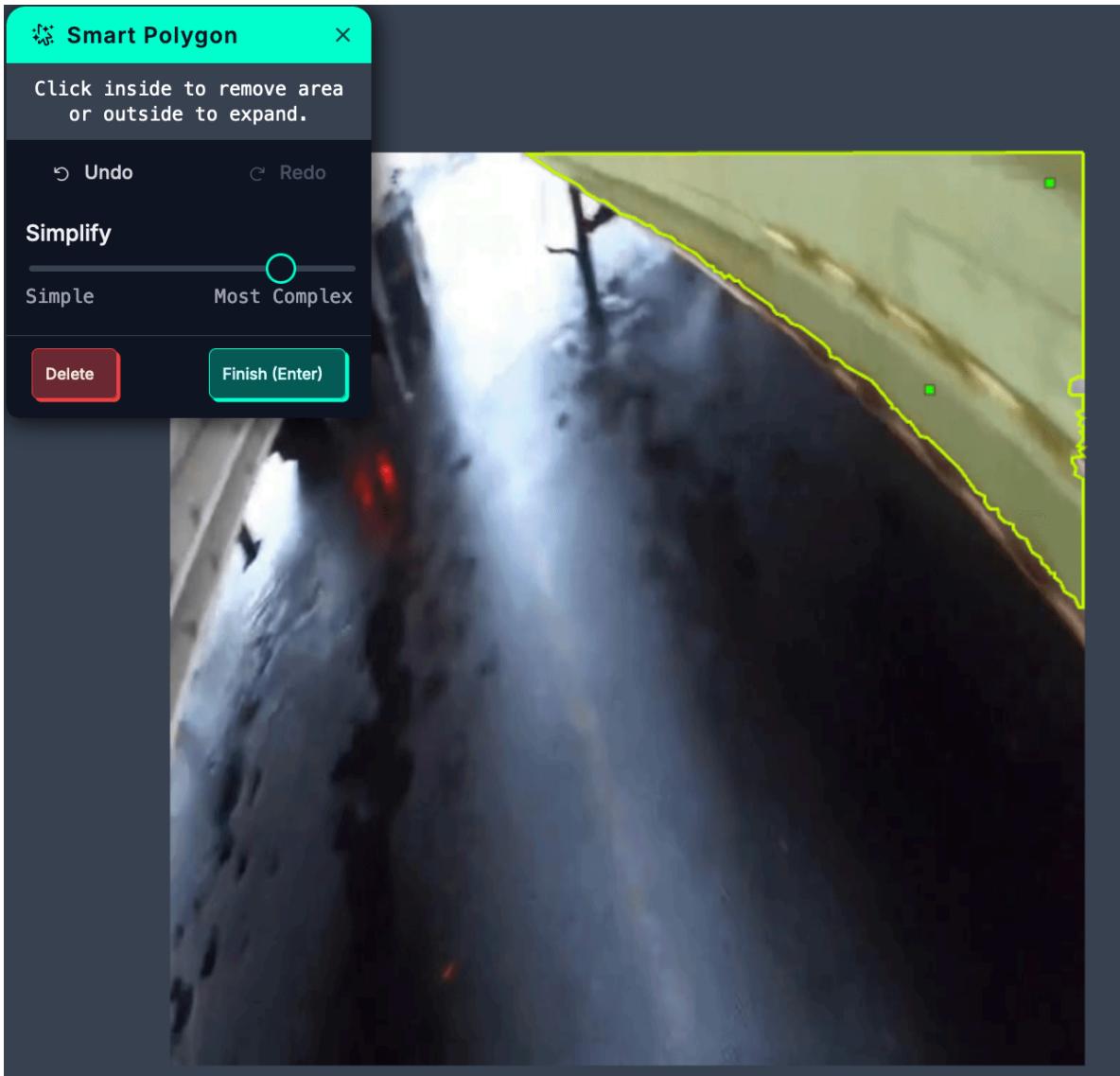


To draw bounding boxes, choose the bounding box labeling tool in the right sidebar.

With that said, we recommend using the [Roboflow Smart Polygon](#) tool to speed up the labeling process. Smart Polygon lets you use the [SAM-2](#) foundation model to label objects with a single click. Smart Polygons are automatically converted to an object detection format when your data is exported.

To enable [Label Assist](#), click on the magic wand icon in the right sidebar in the annotation interface. Choose the Enhanced labeling option.

You can then point and click on any region that you want to include in an annotation:



To expand a label, click outside the region that the tool has selected. To refine a polygon, click inside the annotated region.

When you have labeled an object, click "Finish" to save the annotation. You will then be asked to choose a class for the annotation, after which point you can label the next object in the image.

Step #3: Generate a Dataset Version

Once you have labeled your dataset, you are ready to generate a dataset version. A dataset version is a snapshot of a dataset, frozen in time. You can apply [preprocessing and augmentation](#) datasets to versions, allowing you to experiment with preprocessing and augmentation without changing the underlying dataset.

To generate a dataset version, click "Generate" in the left sidebar.

Then, you can select the preprocessing and augmentation steps that you want to apply.

The screenshot shows the Roboflow interface for configuring a dataset. It includes sections for Source Images, Train/Test Split, and Preprocessing, along with a summary table and a Continue button.

Source Images	Images: 156
	Classes: 1
	Unannotated: 156

Train/Test Split	Training Set: 114 images
	Validation Set: 33 images
	Testing Set: 9 images

Preprocessing

② What can preprocessing do?

Decrease training time and increase performance by applying image transformations to all images in this dataset.

Auto-Orient	Edit	x
Resize Stretch to 640×640	Edit	x
+ Add Preprocessing Step		

Continue

For the first version of your project, we recommend leaving the preprocessing steps as their default, and applying no augmentations. This will allow you to create a baseline against which you can compare model versions.

Once you have configured your dataset version, click “Generate” at the bottom of the page. Your dataset version will be generated.

Step #4: Export Data

With a dataset version ready, you can export it for use in training a YOLOv11 model.

Click “Download Dataset”:

v4 2024-09-30 6:19pm

Generated on Sep 30, 2024

[Download Dataset](#)[Edit](#) :

This version doesn't have a model.

Train an optimized, state of the art model with Roboflow or upload a custom trained model to use features like Label Assist and Model Evaluation and deployment options like our auto-scaling API and edge device support.

[Custom Train and Upload](#)[Train with Roboflow](#)

Then choose the YOLOv11 PyTorch TXT format:

Download

[X](#)

Format

YOLOv11

[▼](#)TXT annotations and YAML config used with [YOLOv11](#). Download zip to computer Show download code[Cancel](#)[Continue](#)

Choose the "Show download code option".

A code snippet will appear that you can use to export your dataset.

Step #5: Train a YOLOv11 Model

With the export code snippet ready, you can start training a YOLOv11 model. We have prepared a notebook that you can use to train a model.

To get started, [open the Roboflow YOLOv11 model training notebook](#) in Google Colab. Below, we will walk through the main steps.

First, we need to install a few dependencies:

- [Ultralytics](#), which we will use for training and inference;
- [supervision](#), which we will use to post-process model predictions, and;
- [Roboflow](#), which we will use to download our dataset.

```
%pip install ultralytics supervision roboflow
import ultralytics
ultralytics.checks()
```

Then, update the dataset download section to use the dataset export code snippet:

```
!mkdir {HOME}/datasets
%cd {HOME}/datasets

from google.colab import userdata
from roboflow import Roboflow

ROBOFLOW_API_KEY = userdata.get('ROBOFLOW_API_KEY')
rf = Roboflow(api_key=ROBOFLOW_API_KEY)

workspace = rf.workspace("your-workspace")
project = workspace.project("your-dataset")
version = project.version(VERSION)
dataset = version.download("yolov8")
```

Next, we need to make a few changes to the dataset files to work with the YOLOv11 training routine:

```
!sed -i '$d' {dataset.location}/data.yaml # Delete the last line
!sed -i '$d' {dataset.location}/data.yaml # Delete the second-to-last line
!sed -i '$d' {dataset.location}/data.yaml # Delete the third-to-last line

!echo 'test: ../test/images' >> {dataset.location}/data.yaml
!echo 'train: ../train/images' >> {dataset.location}/data.yaml
!echo 'val: ../valid/images' >> {dataset.location}/data.yaml
```

We can then train our model using the following command:

```
%cd {HOME}
```

```
!yolo task=detect mode=train model=yolo11s.pt data={dataset.location}/data.yaml epochs=10 imgsze=640 plots=
```

Here, we train a YOLO11s model. If you want to train a model of a different size, replace yolo11s with the ID of the base model weights to use.

We will train for 10 epochs to test the training. For a model you want to use in production you may want to train for 50-100 epochs.

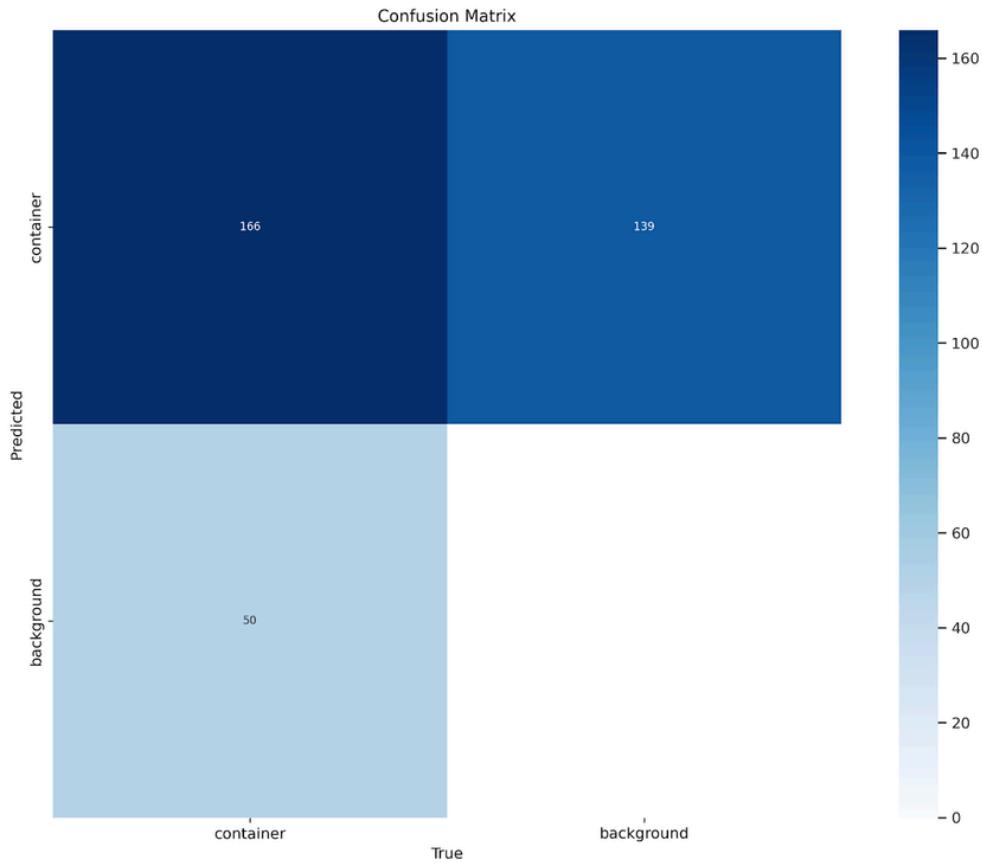
```
%cd {HOME}
!yolo task=detect mode=train model=yolo11s.pt data={dataset.location}/data.yaml epochs=10 imgsze=640 plots=True
/content
Downloading https://github.com/ultralytics/assets/releases/download/v8.3.0/yolo11s.pt to 'yolo11s.pt'...
100% 18.4M/18.4M [00:00<00:00, 62.8MB/s]
Ultralytics 8.3.1 ➜ Python-3.10.12 torch-2.4.1+cu121 CUDA:0 (NVIDIA A100-SXM4-40GB, 40514MiB)
engine/trainer: task=detect, mode=train, model=yolo11s.pt, data=/content/datasets/Shipping-Containers-4/data.yaml,
Downloading https://ultralytics.com/assets/Arial.ttf to '/root/.config/Ultralytics/Arial.ttf'...
100% 755k/755k [00:00<00:00, 151MB/s]
Overriding model.yaml nc=80 with nc=1

      from    n      params   module           arguments
  0        -1    1       928  ultralytics.nn.modules.conv.Conv      [3, 32, 3, 2]
  1        -1    1     18560  ultralytics.nn.modules.conv.Conv      [32, 64, 3, 2]
  2        -1    1     26080  ultralytics.nn.modules.block.C3k2      [64, 128, 1, False, 0.25]
  3        -1    1    147712  ultralytics.nn.modules.conv.Conv      [128, 128, 3, 2]
  4        -1    1    103360  ultralytics.nn.modules.block.C3k2      [128, 256, 1, False, 0.25]
  5        -1    1    590336  ultralytics.nn.modules.conv.Conv      [256, 256, 3, 2]
  6        -1    1    346112  ultralytics.nn.modules.block.C3k2      [256, 256, 1, True]
```

You can view the confusion matrix for your model using the following code:

```
from IPython.display import Image as IPyImage
IPyImage(filename=f'{HOME}/runs/detect/train/confusion_matrix.png', width=600)
```

Here is the [confusion matrix](#) for our model:



You can then run inference with the Ultralytics yolo command line interface:

```
!yolo task=detect mode=predict model={HOME}/runs/detect/train/weights/best.pt conf=0.25 source={dataset.lo
```

This will run inference on images in our ``test/images`` test set in our dataset. This command will return all predictions with a confidence greater than 0.25 (25%).

We can then visualize the model results using the following code:

```
import glob
import os
from IPython.display import Image as IPyImage, display

latest_folder = max(glob.glob('/content/runs/detect/predict*'), key=os.path.getmtime)
for img in glob.glob(f'{latest_folder}/*.jpg')[3:]:
    display(IPyImage(filename=img, width=600))
    print("\n")
```

Here is an example of a result from our model:



Deployment

Congratulations! You have successfully trained a YOLOv11 model. Let's finish up by deploying it to Roboflow, and retrieving the results with [inference](#).

Note: Your [API key](#) should be for your project! If you used our workspace thus far, deployment won't work until you create your own project.

```
project.version(dataset.version).deploy(model_type="yolov11", model_path=f"{HOME}/runs/detec
```

This might take a few minutes to complete. Once done, you'll be able to access the model from any device:

```
pip install inference
```

```
import cv2
import supervision as sv
from inference import get_model

model_id = project.id.split("/")[1] + "/" + dataset.version
model = get_model(model_id, "<ROBOFLOW_API_KEY>")

image = cv2.imread("<PATH_TO_IMAGE>")
result = model.infer(image)
detections = sv.Detections.from_inference(result)
```

Conclusion

YOLOv11 is a computer vision model developed by Ultralytics, the creators of [YOLOv5](#) and [YOLOv8](#). You can train models for object detection, segmentation, classification, and other tasks with YOLOv11.

In this guide, we walked through how to train a YOLOv11 model. We prepared a dataset in Roboflow, then exported the dataset in the YOLOv11 PyTorch TXT format for use in training a model. We trained our model in a Google Colab environment using the YOLO11n weights, evaluated it on images from our model test set, and deployed it to Roboflow.

The full code is available as a [notebook](#). To explore more, check out [Roboflow Notebooks](#), our open source repository of computer vision notebooks.

Cite this Post

Use the following entry to cite this post in your research:

[James Gallagher](#). (Oct 8, 2024). *How to Train a YOLOv11 Object Detection Model on a Custom Dataset*. Roboflow Blog: <https://blog.roboflow.com/yolov11-how-to-train-custom-data/>

Discuss this Post

If you have any questions about this blog post, start a discussion on the [Roboflow Forum](#).

Written by

James Gallagher

James is a technical writer at Roboflow, with experience writing documentation on how to train and use state-of-the-art computer vision models.

[View more posts](#)

Topics

[COMPUTER VISION](#)[OBJECT DETECTION](#)

More About Computer Vision

[View All Computer Vision Posts](#)

Computer Vision Use Cases in Sports

Nov 5, 2024 • 7 min read

Computer Vision Use Cases in Robotics

Oct 31, 2024 • 7 min read

Computer Vision Manufacturing Use Cases: A Guide

Oct 31, 2024 • 7 min read

Computer Vision Logistics Use Cases: A Guide

Oct 30, 2024 • 7 min read

How to Become a Computer Vision Engineer

Oct 30, 2024 • 8 min read

What is Feature Matching?

Oct 30, 2024 • 16 min read

[COMPUTER VISION](#)

[OBJECT DETECTION](#)

Products	Solutions	Developers	Pricing	Docs	Blog	🔍	Sign In	Get Started
UNIVERSE	NOTEBOOKS	DOCUMENTATION	MANUFACTURING	YOLOV8		ABOUT US		
Annotate	Autodistill	User Forum	Oil and Gas	YOLOv8		Careers		
Train	Supervision	Blog	Retail	YOLOv5		HIRING		
Workflows	Inference	Learn Computer Vision	Safety and Security	PaliGemma		Press		
Deploy	Roboflow 100	Weekly Product Webinar	Transportation	CLIP		Contact		
Pricing	Open Source	Convert Annotation Formats	Explore All Industries	Multimodal Models		Status		
Talk to Sales		Computer Vision Models		Explore All Models				



[Terms of Service](#) [Enterprise Terms](#) [Privacy Policy](#) [Sitemap](#)

© 2024 Roboflow, Inc. All rights reserved.