Segmenting Anything with Segment Anything and FiftyOne

Kyle Stratis

The recent explosion of generative AI has led to a new generation of promptable **zero-shot** (requiring no additional training) models in many domains, with perhaps the most exciting being Meta's Segment Anything Model (SAM). Trained on 11 million images with over 1.1 billion segmentation masks, Segment Anything can rapidly segment entire images as well as individual regions within an image with the use of a point or bounding box prompt.

When you use Segment Anything on its own, you have to write additional visualization code if you want to actually see your masks. Enter FiftyOne, an open source toolkit for building computer vision datasets visually. With FiftyOne, you can view, slice, filter, and do any number of operations on your image datasets. Powerful enough on its own, but when you introduce Segment Anything to the mix, you have an indispensable toolchain for building and inspecting segmentation datasets.

In this article, you will learn how to:

- Set up an environment with FiftyOne, SAM, and an image dataset
- Segment a single detected object in an image with Segment Anything and view it in FiftyOne
- Segment all detected objects in an image with Segment Anything and view them in FiftyOne
- Segment all detected objects in all images in a dataset with Segment Anything and view them in FiftyOne

Note: You can follow along in the code blocks below or in this Jupyter Notebook.

Setting Up

The Environment

To begin, create and activate a virtual environment using your tool of choice. In the activated environment, install FiftyOne, SAM, and a few additional tools to work with your data:

```
$ pip install fiftyone git+https://github.com/facebookresearch/segment-anything.git torch torchvision opencv-python numpy==1.24.4
```

After installing these dependencies, download the default (`vit_h`) Segment Anything model checkpoint to the same directory as your code. You can experiment with the other model sizes on offer as well, just ensure to update your code to point to the correct file.

Then, import the following packages:

```
# Standard ibrary imports
from copy import deepcopy

# External imports
import cv2
import fiftyone as fo
import fiftyone.zoo as foz
import numpy as np
from segment_anything import SamPredictor, sam_model_registry
import torch
```

Now that you've imported your dependencies,

The Model

To use the model, you will need to load it into memory from the checkpoint you downloaded and then instantiate a `SamPredictor` object. If you have a CUDA-enabled GPU, you can optionally move the loaded model to your GPU.

```
sam_checkpoint = "sam_vit_h_4b8939.pth"
model_type = "vit_h"
sam = sam_model_registry[model_type](checkpoint=sam_checkpoint)

# Only run the below line if you have a CUDA-enabled GPU
sam.to(device="cuda")

predictor = SamPredictor(sam)
```

The Dataset

Now it's time to set up your dataset. Even though assembling a dataset is normally the most tedious part of any computer vision project, FiftyOne's data zoo provides you with easy access to several major datasets.

For this tutorial, we will use FiftyOne's `quickstart` dataset, and then take a slice of the first 10 images:

```
dataset = foz.load_zoo_dataset("quickstart")
sliced_view = dataset[:10]
```

That's it. `load_zoo_dataset()` will download the dataset if you don't have it already and load it into memory, and list slice notation allows you to create `sliced_view` which is a `DatasetView`. The `DatasetView` is an object that allows you to split, reorganize, tag, and otherwise modify the samples in your dataset without actually changing the underlying data.

This dataset is especially useful because it contains ground truth bounding box annotations, which you can use as prompts to direct SAM where in the image to segment.

Segmenting and Viewing

Segmenting and Viewing a Single Object in a Single Image

In the last section, you set up your environment, your Segment Anything model, and your dataset. Now you can explore how to segment images with SAM and view the segments in FiftyOne, starting with one single sample.

First, select the first sample from the `sliced_view` dataset view:

```
sample = sliced_view.first()
```

> ⚠ **Warning**
>
> You can't access individual `Sample` objects by their index. Instead you can use methods like `.first()` and `.last()`, sample IDs, or filepaths.

To use Segment Anything, you have to load your image into memory as a NumPy array and then call `.set_image()` to generate embeddings for your image. Because we are using OpenCV in this tutorial, you will have to also change the color format from the default BGR to RGB:

```
image = cv2.imread(sample["filepath"])
image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
predictor.set_image(image)
```

With the embeddings created, you can start generating predictions. As mentioned earlier, you will be using the ground truth detection bounding boxes available in the `Sample` as bounding box prompts, but for this part of the tutorial we will start with just a single detection.

Even though you have bounding boxes available to you, they aren't in a format that SAM understands. Luckily, FiftyOne provides a utility to convert from FiftyOne's relative xywh format to the absolute xyxy VOC format.

```
h, w, _ = image.shape
detection = sample["ground_truth"]["detections"][0]
input_bbox = fo.utils.voc.VOCBoundingBox.from_detection_format(detection["bou
```

Now that you have `input_bbox`, you can call `SamPredictor`'s `predict` method to generate a single mask and score for the main image contained within the bounding box prompt:

```
mask, score, _ = predictor.predict(
    box=np.array([input_bbox.xmin, input_bbox.ymin, input_bbox.xmax, input_bb
    multimask_output=False,
)
```

After generating the mask and its score, you can load this information into the `Sample` object as a prediction in its `predictions` field. An easy way to do this is to make a deep copy of the detection, then add the mask to the `mask` field and the score to the `confidence` field.

Take a deeper look at the second line below before moving on, because a few things are going on here. FiftyOne expects segmentation masks to be 2D arrays that only cover the governing bounding box, but if you check the mask's shape, you'll see that it is a 3D array with the dimensions of the original image.

So we instead store the 2D array with `mask[0]` and only take the values inside of the bounding box with list slicing.

To save this as a detection, pass it in a list to the `Detections` constructor, add it to the `Sample` object in the `predictions` field, and save the sample:

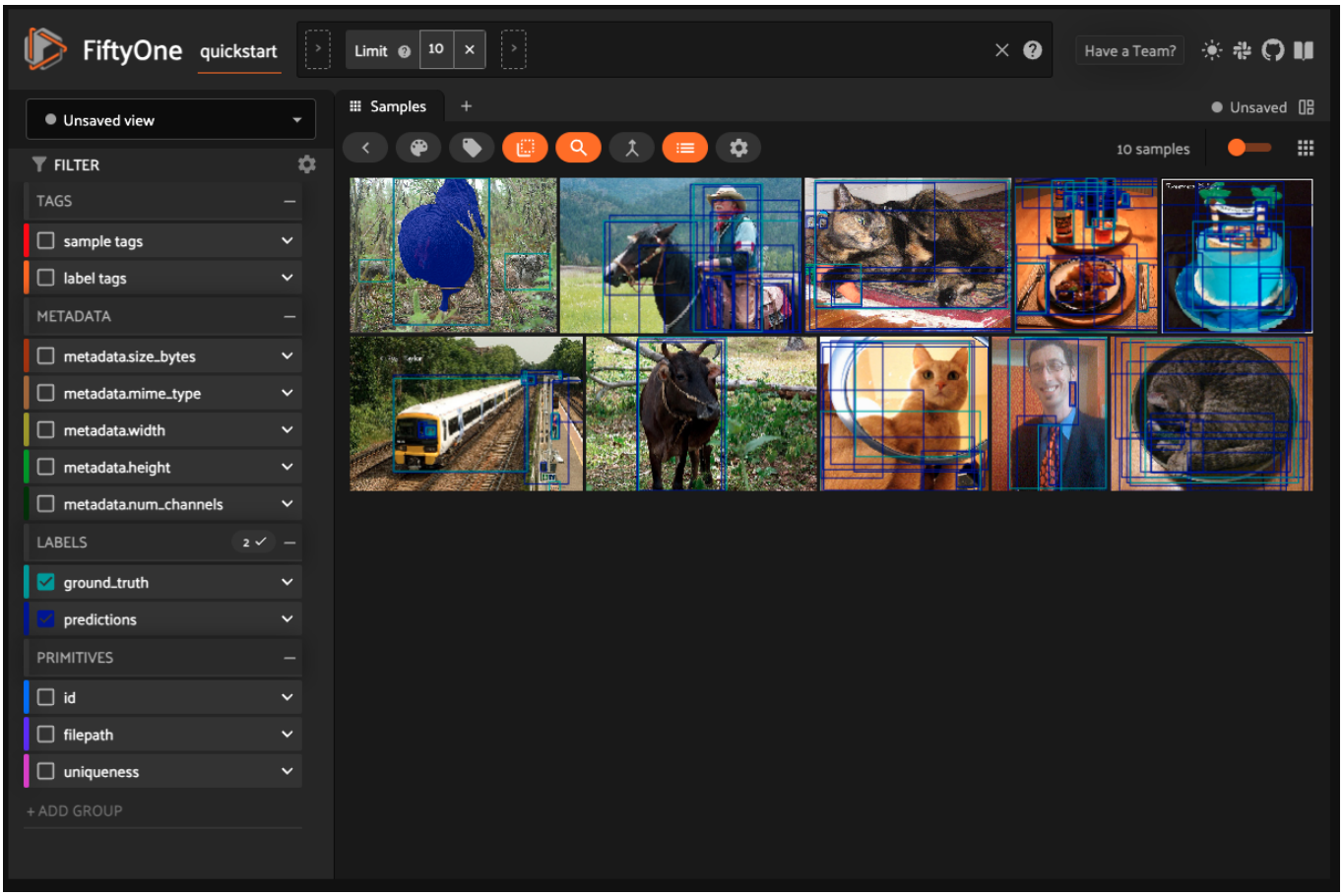
```
prediction = deepcopy(detection)
prediction["mask"] = mask[0][input_bbox.ymin:input_bbox.ymax+1, input_bbox.xm
```

```
prediction["confidence"] = score
sample["predictions"] = fo.Detections(detections=[prediction])
sample.save()
```

And with that, you've successfully generated a segmentation mask with Segment Anything and saved it along with its score to a `Sample` object representing the sample image.

With that done, you can launch the FiftyOne application and look at the segmented object:

```
session = fo.launch_app(sliced_view)
```



A screenshot of the FiftyOne app showing a 10-image dataset and the first image with a Segment Anything-generated segmentation mask

Putting this code together, you should have:

```
# Get the sample and open the image
sample = sliced_view.first()
image = cv2.imread(sample["filepath"])
image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)

# Generate embeddings
```

```
predictor.set_image(image)

# Get ground truth bounding box and convert it to VOC
h, w, _ = image.shape
detection = sample["ground_truth"]["detections"][0]
input_bbox = fo.utils.voc.VOCBoundingBox.from_detection_format(detection["bou

# Generate and save the mask to the sample and sample to the DatasetView
mask, score, _ = predictor.predict(
    box=np.array([input_bbox.xmin, input_bbox.ymin, input_bbox.xmax, input_bb
    multimask_output=False,
)
prediction = deepcopy(detection)
prediction["mask"] = mask[0][input_bbox.ymin:input_bbox.ymax+1, input_bbox.xm
prediction["confidence"] = score
sample["predictions"] = fo.Detections(detections=[prediction])
sample.save()

# Launch FiftyOne app
session = fo.launch_app(sliced_view)
```

Segmenting and Viewing All Detections in a Single Image

The hard part is done. Now you can apply the code you wrote above to all detected objects in an image by iterating through the sample's ground truth detections and storing all predictions in a list before saving them to the sample:

```
# Get the sample and open the image
sample = sliced_view.first()
image = cv2.imread(sample["filepath"])
image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)

# Generate embeddings
predictor.set_image(image)

# Iterate through all detections the sample
predictions = []
h, w, _ = image.shape
for detection in sample["ground_truth"]["detections"]:

    # Get ground truth bounding box and convert it to VOC
    input_bbox = fo.utils.voc.VOCBoundingBox.from_detection_format(detection[
```

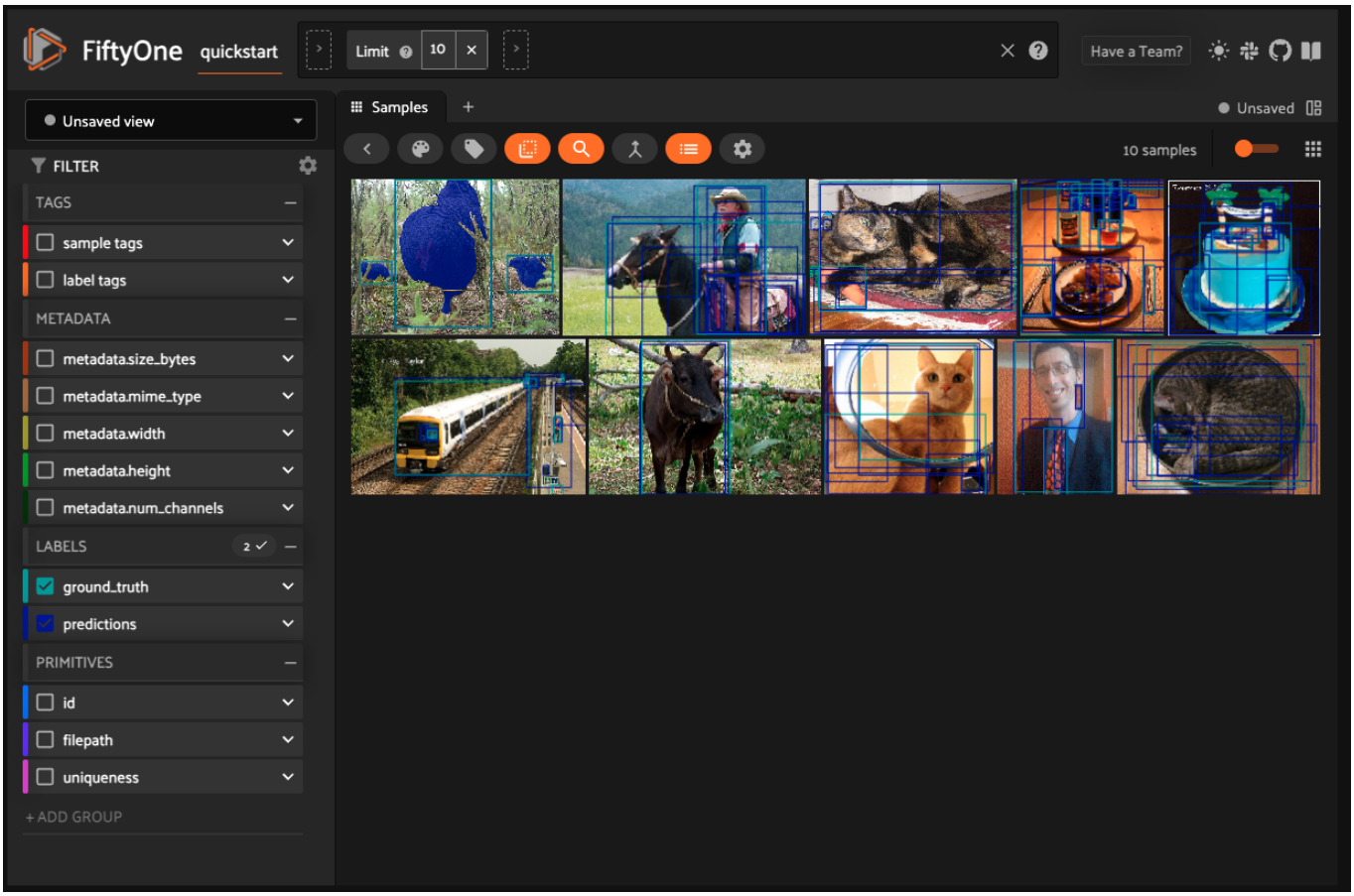
```
    # Generate and save the mask to the sample and sample to the DatasetView
    mask, score, _ = predictor.predict(
        box=np.array([input_bbox.xmin, input_bbox.ymin, input_bbox.xmax, inpu
        multimask_output=False,
    )
    prediction = deepcopy(detection)
    prediction["mask"] = mask[0][input_bbox.ymin:input_bbox.ymax+1, input_bbo
    prediction["confidence"] = score
    predictions.append(prediction)

# Create a Detections object from the predictions list and save it to the sam
sample["predictions"] = fo.Detections(detections=predictions)
sample.save()
```

While this code sample may look more complex, it is the same code you already wrote, just moved into a for loop that builds the `predictions` list.

To view the new segments, you can either fully relaunch the FiftyOne app or, more simply, refresh the session:

```
session.refresh()
```

After doing this, you will see 3 segment masks overlaid on the first picture in the dataset:



The FiftyOne interface now showing 3 segment masks in the first image of the dataset

Segmenting and Viewing All Detections in All Images in a Dataset

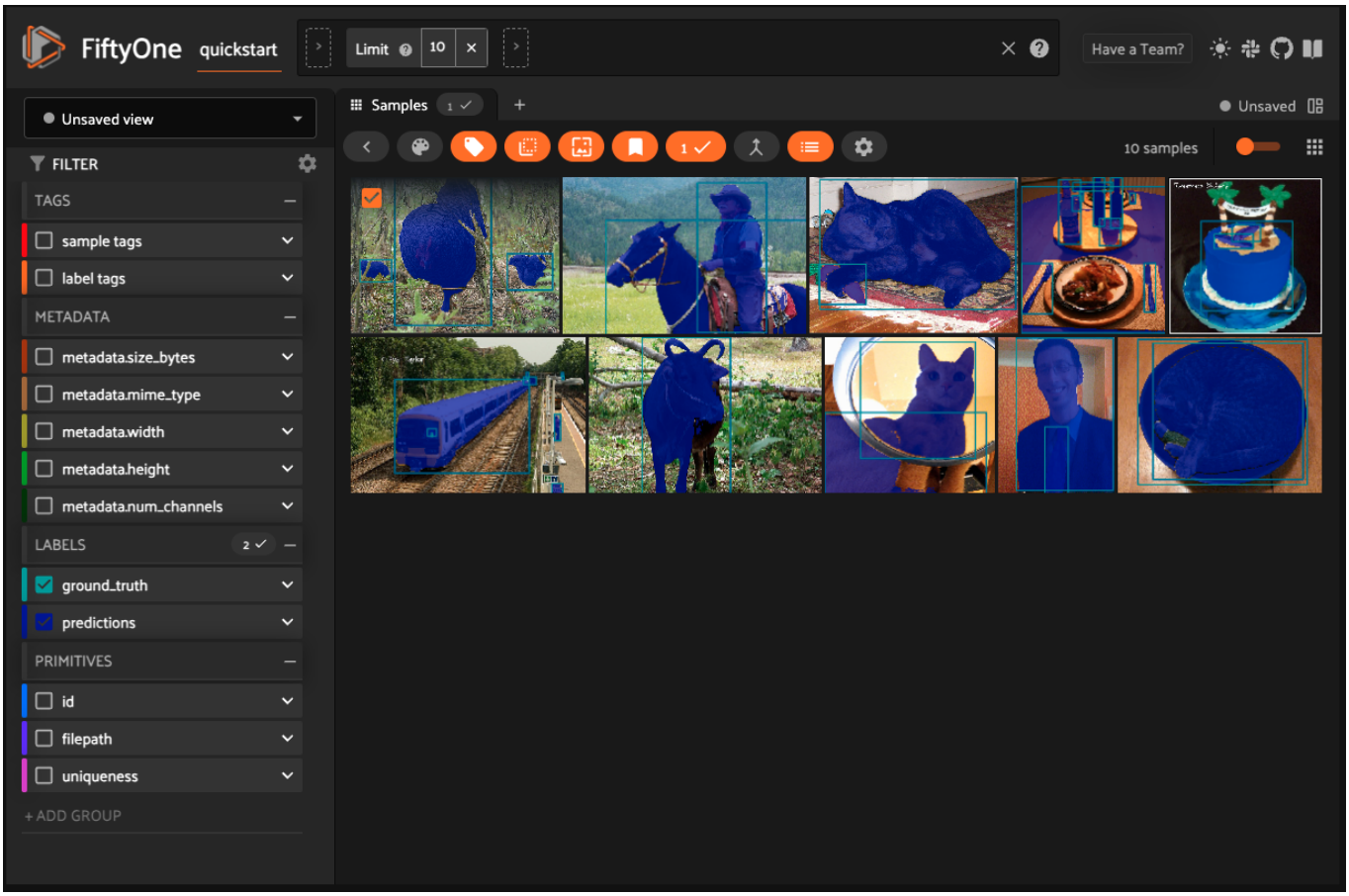
This part is even simpler than the last. The code you've already written in the previous session just needs to be wrapped in a for loop, iterating through each sample in the `sliced_view` `DatasetView`:

```
for sample in sliced_view:
    image = cv2.imread(sample["filepath"])
    image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
    predictor.set_image(image)
    predictions = []
    h, w, _ = image.shape
    for detection in sample["ground_truth"]["detections"]:
        input_bbox = fo.utils.voc.VOCBoundingBox.from_detection_format(detect
        mask, score, _ = predictor.predict(
            box=np.array([input_bbox.xmin, input_bbox.ymin, input_bbox.xmax,
            multimask_output=False,
        )
        prediction = deepcopy(detection)
```

```
        prediction["mask"] = mask[0][input_bbox.ymin:input_bbox.ymax+1, input
        prediction["confidence"] = score
        predictions.append(prediction)
    sample["predictions"] = fo.Detections(detections=predictions)
    sample.save()
```

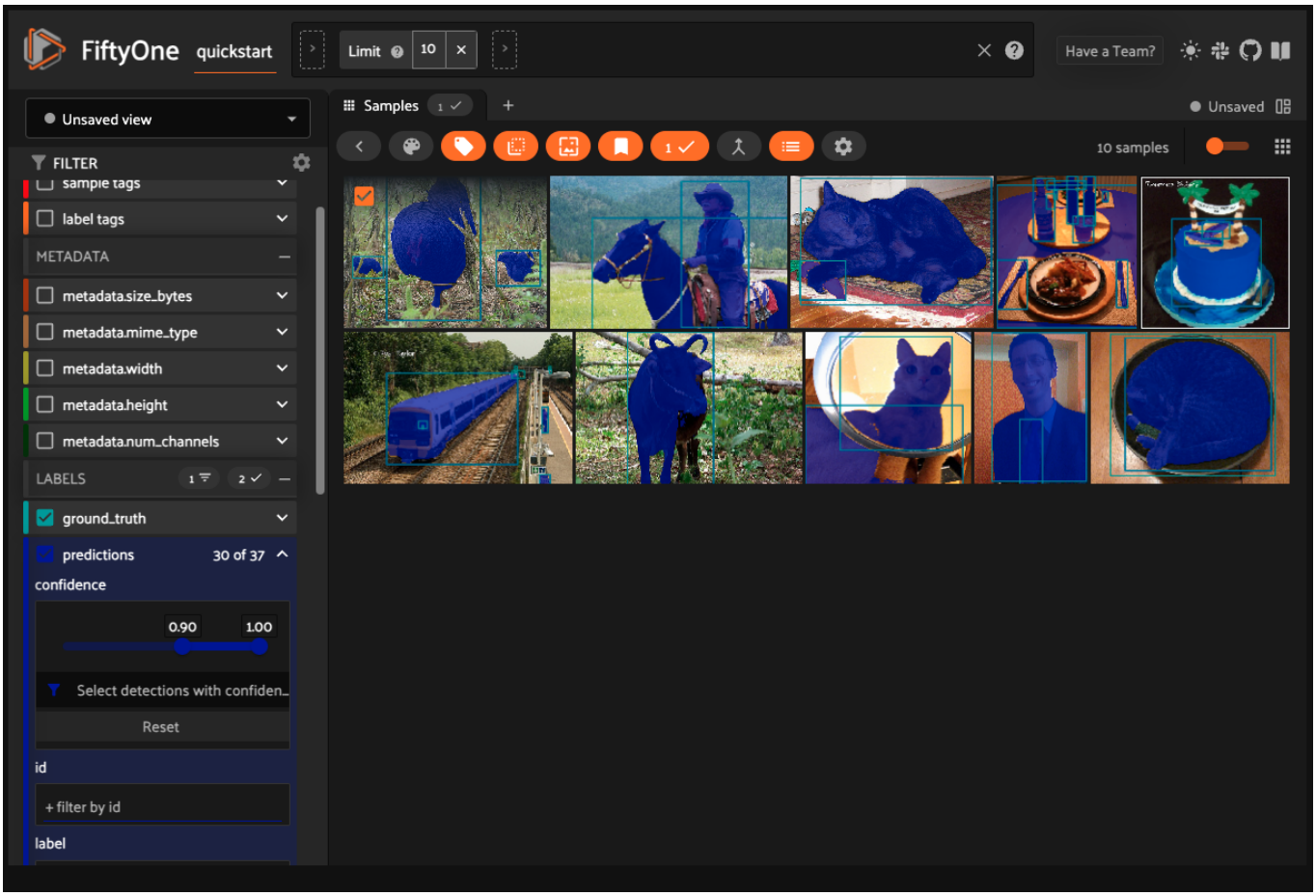
Again, just reset the FiftyOne session and you'll be able to see segments displayed on all of the images in your dataset:

```
session.refresh()
```



The FiftyOne interface now showing segments for all objects in all images in the dataset

With all images segmented, you can now do all kinds of operations on your dataset via FiftyOne. Do you remember saving the mask's confidence score in the prediction dictionary earlier in the tutorial? Now you can filter based on that score: just click "predictions" under "LABELS" and slide the slider to contain any range you want, like this setting that excludes all detections under a 0.9 confidence score:



The FiftyOne interface showing only high-confidence segmentation masks and the open predictions panel, revealing the confidence slider

Wrapping Up

Segment Anything and FiftyOne are a powerful combination for anyone building segmentation datasets for computer vision tasks. Segment Anything allows you to automatically generate whole-image and region-specific segmentation masks with no additional training while FiftyOne allows you to view, filter, and organize your datasets.

In this tutorial, you joined these two powerful tools together, learning how to:

- Set up Segment Anything to run locally from a checkpoint file
- Download pre-built datasets from FiftyOne's dataset zoo
- Segment a single detected object in an image with Segment Anything
- Segment all detected objects in an image with Segment Anything
- Segment all detected objets in all images in a dataset with Segment Anything
- Store and view segmentation masks in FiftyOne

What kinds of datasets will you be exploring now with Segment Anything and FiftyOne?