Cloud AutoML Vision Object Detection

Edge TensorFlow.js tutorial

Terminology: See the AutoML Vision Edge terminology (https://cloud.google.com/vision/automl/object-detection/docs/terminology) page for a list of terms used in this tutorial.

What you will build


```
[
  {
    "box": {
      "left": 104.34133559465408,
      "top": 22.210068069398403,
      "width": 72.48612493276596,
      "height": 56.39078188687563
    },
    "label": "Tomato",
    "score": 0.9775320291519165
  },
  {
    "box": {
      "left": 255.76230883598328,
      "top": 88.64829875528812,
      "width": 51.92631483078003,
      "height": 61.99347786605358
    },
    "label": "Tomato",
    "score": 0.9570426344871521
  },
  {
    "box": {
      "left": -0.2758800983428955,
      "top": 1.8042884767055511,
      "width": 475.52409768104553,
      "height": 253.8670040667057
    },
    "label": "Salad",
    "score": 0.8704740405082703
  }
]
```


In this tutorial you will download a TensorFlow.js Object Detection model trained and exported using AutoML Vision Edge. You will then build a web page that loads the model and makes a

prediction on an image.

Objectives

You will write JavaScript code to:

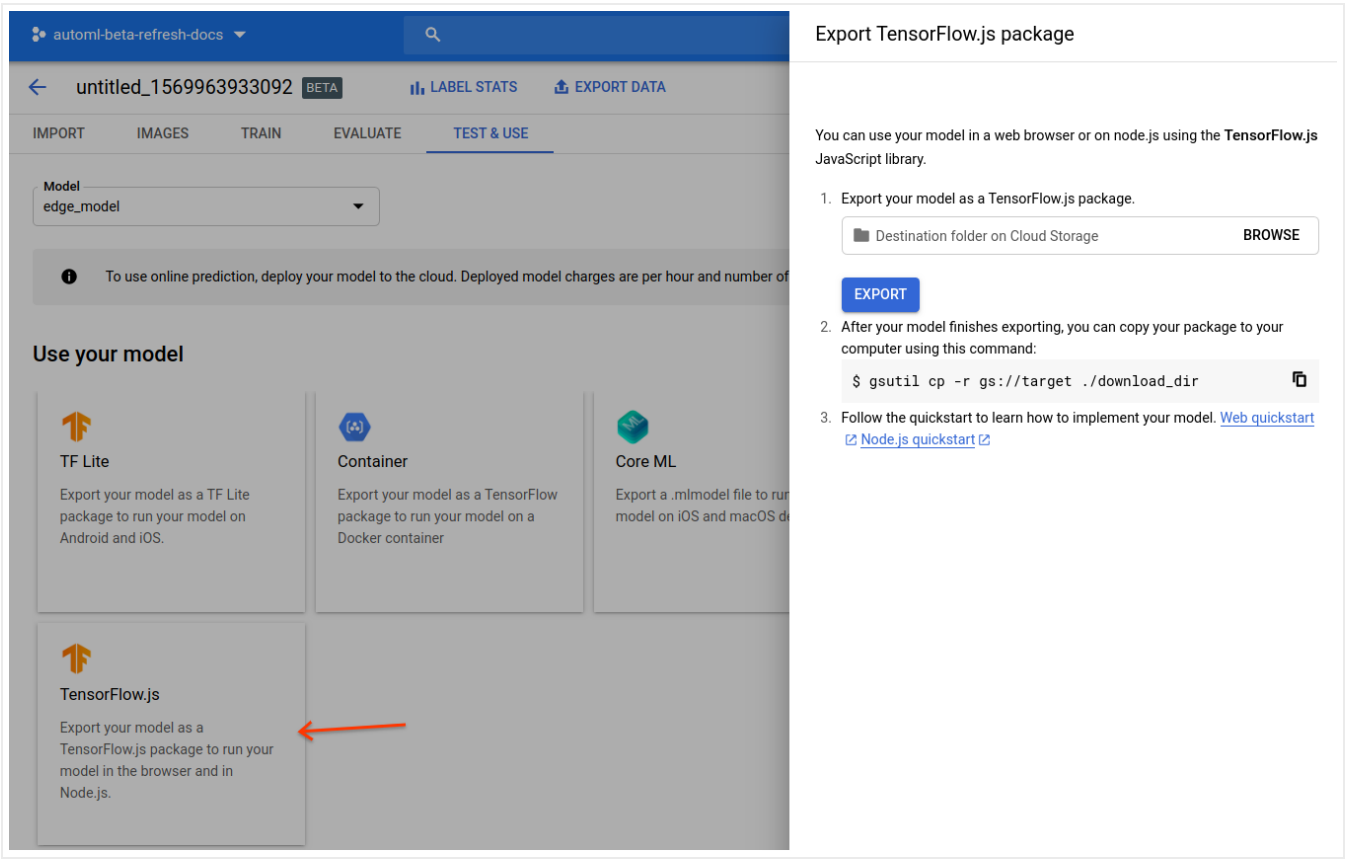
- Run a pre-trained AutoML Vision Edge Object Detection model in a web page using the TensorFlow.js library.

Before you begin

Train a model from AutoML Vision Edge

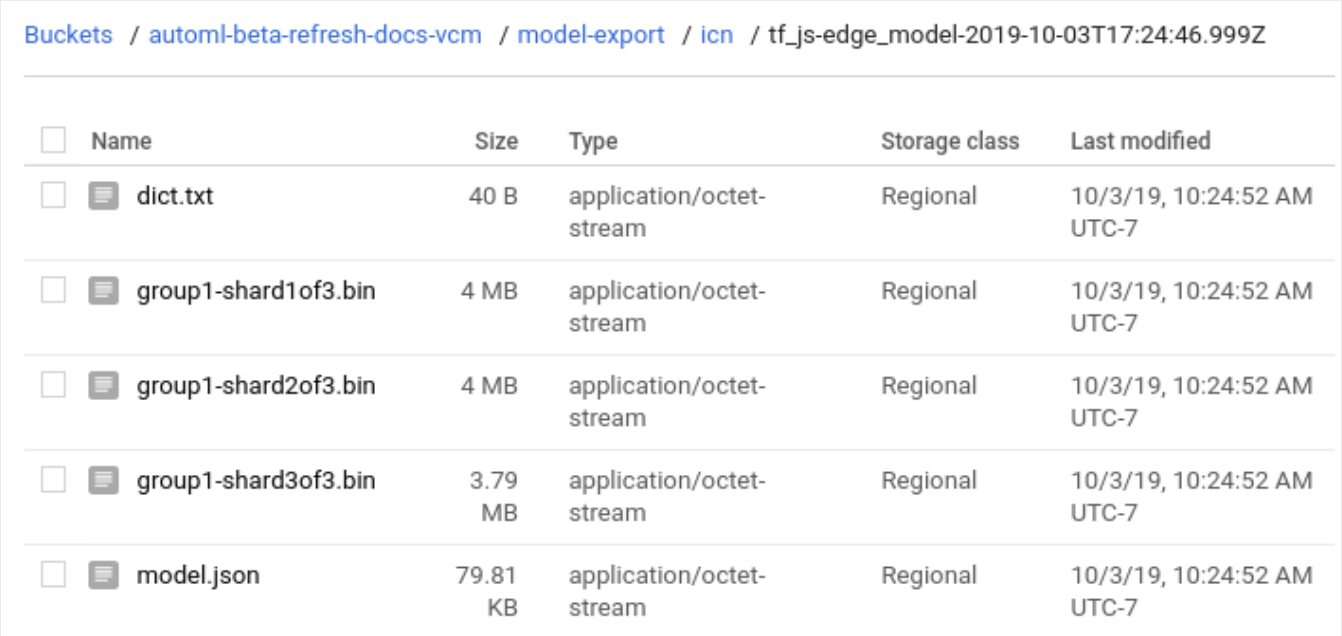
Before you can deploy a model to an Edge device you must first train and export a TensorFlow.js model from AutoML Vision Edge following the Edge device model quickstart (https://cloud.google.com/vision/automl/object-detection/docs/edge-quickstart).

In the final step, export the model to TensorFlow.js:



After completing the quickstart you should have the following exported files on Google Cloud Storage:

- a `dict.txt` file with labels
- a `model.json` file
- `*.bin` weight files



Buckets / automl-beta-refresh-docs-vcm / model-export / icn / tf_js-edge_model-2019-10-03T17:24:46.999Z

Name	Size	Type	Storage class	Last modified
dict.txt	40 B	application/octet-stream	Regional	10/3/19, 10:24:52 AM UTC-7
group1-shard1of3.bin	4 MB	application/octet-stream	Regional	10/3/19, 10:24:52 AM UTC-7
group1-shard2of3.bin	4 MB	application/octet-stream	Regional	10/3/19, 10:24:52 AM UTC-7
group1-shard3of3.bin	3.79 MB	application/octet-stream	Regional	10/3/19, 10:24:52 AM UTC-7
model.json	79.81 KB	application/octet-stream	Regional	10/3/19, 10:24:52 AM UTC-7

Download the model files

Copy the exported files from Google Cloud Storage to a local directory:

```
gsutil cp gs://${cloud-storage-bucket}/model-export/iod/${model-name}/* ${local-fold
```

Note: If you download the files via the UI instead, they are renamed with a timestamp, and you will have to modify their names to their original form.

Write a small web app

After you have your TensorFlow.js model files stored locally you are ready to write your web app:

1. Navigate to the local directory where your model files are stored if you haven't already.
2. Create an `index.html` file in that same local directory with the following contents:

   `index.html`

   CODE SAMPLES

COM/TENSORFLOW/TFJS/BLOB/MASTER/TFJS-AUTOML/CODE_SNIPPETS/OBJECT_DETECTION.HTML)

FEEDBACK (#)

```
<script src="https://unpkg.com/@tensorflow/tfjs"></script>
<script src="https://unpkg.com/@tensorflow/tfjs-automl"></script>
<img id="salad" crossorigin="anonymous" src="https://storage.googleapis.com/t
<script>
async function run() {
  const model = await tf.automl.loadObjectDetection('model.json');
  const img = document.getElementById('salad');
  const options = {score: 0.5, iou: 0.5, topk: 20};
  const predictions = await model.detect(img, options);
  console.log(predictions);
  // Show the resulting object on the page.
  const pre = document.createElement('pre');
  pre.textContent = JSON.stringify(predictions, null, 2);
  document.body.append(pre);
}
run();
</script>
```

3. (optional): Replace the `src` path attribute of the `<img>` tag to point to your own image.

★ **Note:** If you are using a js bundler with NPM, you can use ES6 imports instead of a <script> tag.

Run the web app

Start a simple HTTP server on port 8000 from the directory with the `index.html` file:

PYTHON NODE.JS

```
python -m SimpleHTTPServer 8000
```

Open http://localhost:8000 (http://localhost:8000) in the browser and you will see the prediction result on the console (**View** > **Developer** > **JavaScript console**), as well as on the page.

The result is a sorted list of prediction objects:

```
[
  {
    "box": {
      "left": 105.1,
      "top": 22.2,
      "width": 70.6,
      "height": 55.7
    },
    "label": "Tomato",
    "score": 0.972
  },
  ...
]
```

For more information regarding the API, see the AutoML NPM package documentation (https://www.npmjs.com/package/@tensorflow/tfjs-automl).

For ideas on how to plot the bounding boxes on top of your image, see this code sample (https://github.com/tensorflow/tfjs/blob/84ae6e04fd7126e1cc6184735a5fb1b14029cfd3/tfjs-automl/demo/object_detection/index.js#L36) that overlays text and boxes onto the image using SVG.

How does it work?

Now that you have your app running you can explore what the code is doing.

The first two script tags load the TensorFlow.js library and the AutoML library, which are available on NPM package manager (https://www.npmjs.com/).

tfjs-automl/code_snippets/object_detection.html (https://github.com/tensorflow/tfjs/blob/master/tfjs-automl/code_snippets/object_detection.html)

3.COM/TENSORFLOW/TFJS/BLOB/MASTER/TFJS-AUTOML/CODE_SNIPPETS/OBJECT_DETECTION.HTML)

```
<script src="https://unpkg.com/@tensorflow/tfjs"></script>
<script src="https://unpkg.com/@tensorflow/tfjs-automl"></script>
```

The AutoML NPM package provides a set of APIs to load and run models produced by AutoML Vision Edge. The package takes care of any pre-processing or post-processing needed to run

the model such as the ability to feed an image or video element, normalizing pixel values, and returning a sorted object with labels and scores.

The image tag loads a test image from a publish Google Cloud Storage path:

tfjs-automl/code_snippets/object_detection.html (https://github.com/tensorflow/tfjs/blob/master/tfjs-automl/code_snippets/object_detection.html)

```
<img id="salad" crossorigin="anonymous" src="https://storage.googleapis.com/tfjs-tes
```

You should replace the image `src` with the path to your own image when testing your model.

Next, you load the model and make a prediction with your image:

tfjs-automl/code_snippets/object_detection.html (https://github.com/tensorflow/tfjs/blob/master/tfjs-automl/code_snippets/object_detection.html)

```
const model = await tf.automl.loadObjectDetection('model.json');
const img = document.getElementById('salad');
const options = {score: 0.5, iou: 0.5, topk: 20};
const predictions = await model.detect(img, options);
```

You specify a model with `tf.automl.loadObjectDetection(url)`. This function takes an absolute or relative URL to the exported `model.json` file, which is this case is a relative path since the `index.html` file and the model files are in the same directory.

You get your predictions by calling `model.detect(img, options)`. This function runs a single image through the model and returns the predicted objects. The input for this function is an html image element, video element, or a 3D tensor (https://js.tensorflow.org/api/latest/#tensor3d).

The `model.detect(img, options)` function also takes in the optional `options` argument with the possible specifications:

- `score` - Probability score between 0 and 1. Defaults to 0.5. Boxes with score lower than this threshold will be ignored.
- `topk` - Only the topk most likely objects are returned. The actual number of objects might be less than this number.

- `iou` - Intersection over union threshold. IoU is a metric between 0 and 1 used to measure the overlap of two boxes. The predicted boxes will not overlap more than the specified threshold.

  See the Evaluating models (https://cloud.google.com/vision/automl/object-detection/docs/evaluate#intersection-over-union) page for more information on model evaluation using Intersection over union (IoU).

For more information regarding the API, see the AutoML NPM package documentation (https://www.npmjs.com/package/@tensorflow/tfjs-automl).

What's next?

You have completed a tutorial of the TensorFlow.js Object Detection web app using an Edge model. You ran the web app in a web browser and made an object detection prediction using your custom Edge model and an image that you loaded from the web. You then examined parts of the sample code to understand the underlying functionality.

Next steps:

- View a demo (https://github.com/tensorflow/tfjs/tree/master/tfjs-automl/demo/object_detection) that uses SVG (https://en.wikipedia.org/wiki/Scalable_Vector_Graphics) to overlay the predicted boxes with probabilities onto the image.
- Learn more about TensorFlow.js (https://www.tensorflow.org/js/).
- Learn more about the AutoML NPM library via the official documentation (https://www.npmjs.com/package/@tensorflow/tfjs-automl).
- Learn more about TensorFlow in general via its tutorials (https://www.tensorflow.org/tutorials).

Last updated November 20, 2019.