Cloud AutoML Vision Object Detection

Exporting Edge models

After you have created (trained) a model, you can export your custom model.

After exporting your model you can then deploy the model to a device.

You can export an image classification model in either generic **Tensorflow Lite** (#tf-lite) format, general **TensorFlow** format, or **TensorFlow.js for Web** (#web) format to a Google Cloud Storage location using the ExportModel API (https://cloud.google.com/automl/docs/reference/rest/v1/projects.locations.models/export).

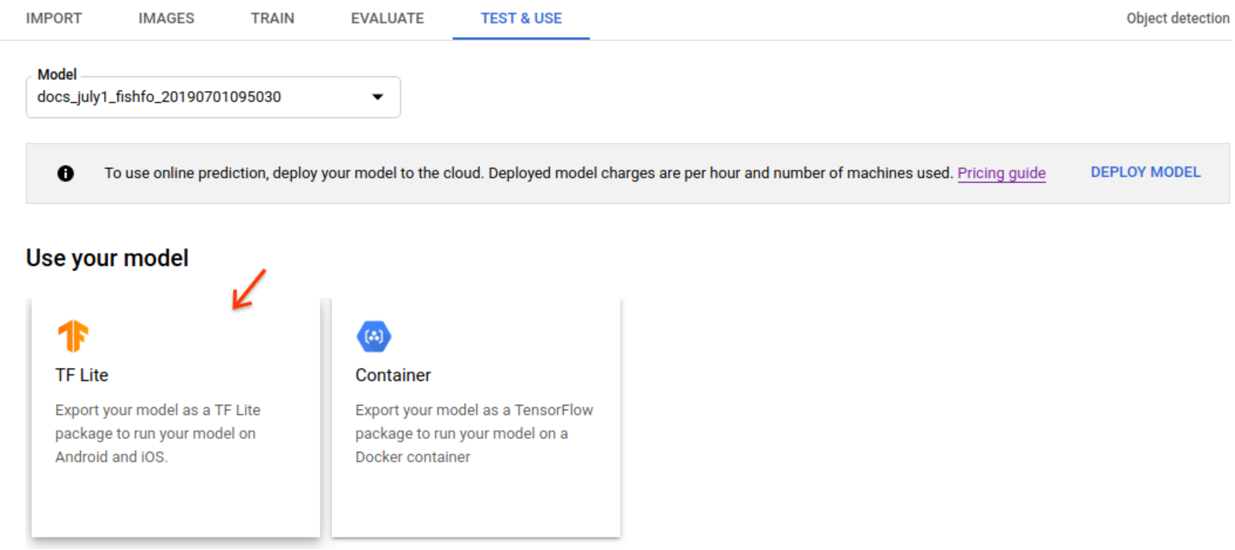
Export to devices

TensorFlow Lite models

WEB UI REST & CMD LINE

1. Open the Cloud AutoML Vision Object Detection UI (https://console.cloud.google.com/vision) and click the lightbulb icon in the left navigation bar to display the available models.

   To view the models for a different project, select the project from the drop-down list in the upper right of the title bar.
2. Select the row for the model you want to use to label your images.
3. Select the **Test & use** tab.
4. In the **Use your model** section select the **TF Lite** card. This opens a side "**Use your on-device model**" window.



Model
docs_july1_fishfo_20190701095030

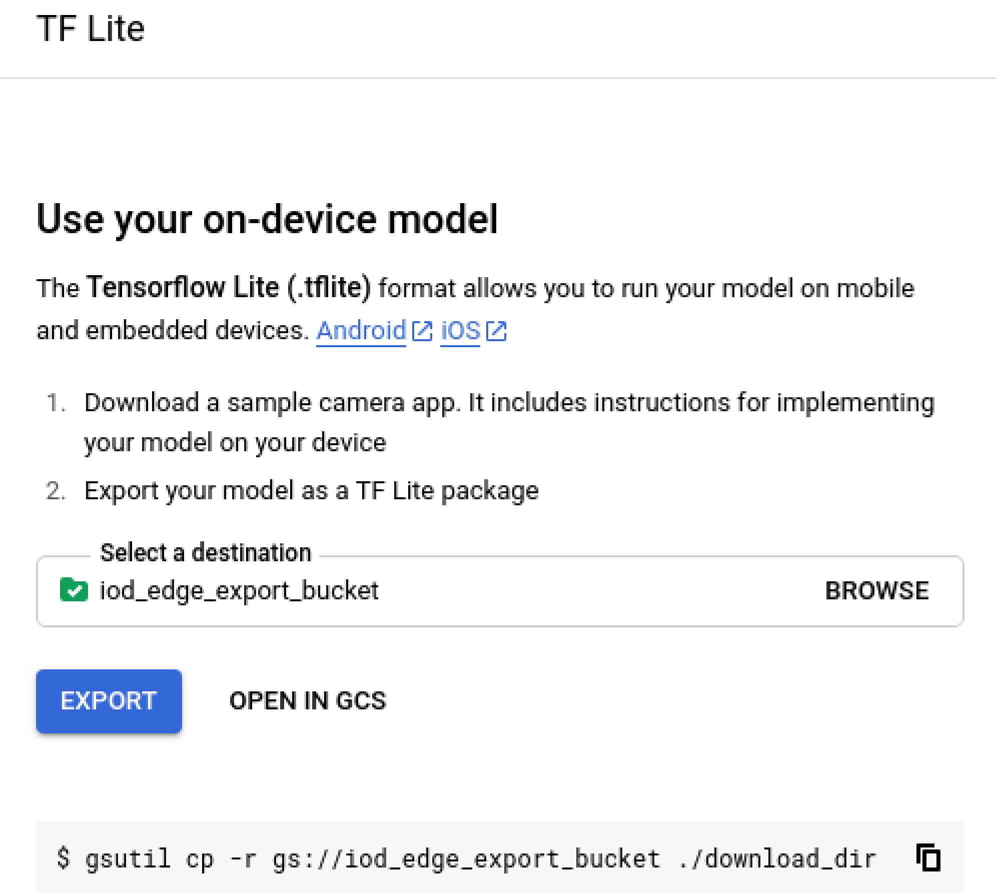
To use online prediction, deploy your model to the cloud. Deployed model charges are per hour and number of machines used. Pricing guide

Use your model

TF Lite
Export your model as a TF Lite package to run your model on Android and iOS.

Container
Export your model as a TensorFlow package to run your model on a Docker container

5. In the side window, specify the output Google Cloud Storage location. After choosing the storage location for your model output, select **Export** to begin the model export operation.



TF Lite

Use your on-device model

The **Tensorflow Lite (.tflite)** format allows you to run your model on mobile and embedded devices. Android iOS

1. Download a sample camera app. It includes instructions for implementing your model on your device
2. Export your model as a TF Lite package

Select a destination
iod_edge_export_bucket

```
$ gsutil cp -r gs://iod_edge_export_bucket ./download_dir
```

6. After exporting you can select the "**Open in Google Cloud Storage**" option in the same window to go directly to the export directory in Google Cloud Storage.

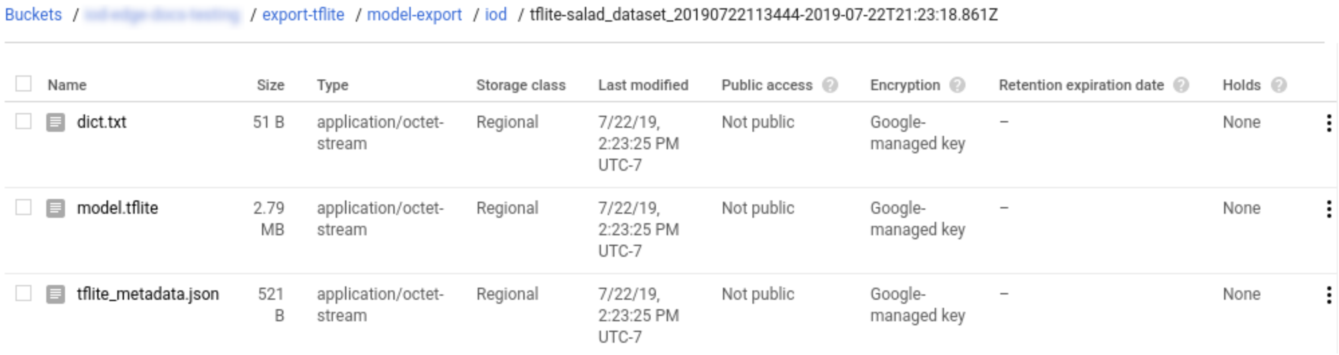
As a result you will see a folder structure in the directory you provided (***cloud-storage-bucket***/[***directory***]). The created folder structure will have the following general format (timestamp in ISO-8601 format):

- `*cloud-storage-bucket*/model-export/iod/*model-type-dataset-name*-YYYY-MM-DDThh:mm:ss.sssZ`

For example:

- `*cloud-storage-bucket*/model-export/iod/tf-saved-model-*dataset-name*-2019-07-22T21:25:35.135Z`
- `*cloud-storage-bucket*/model-export/iod/tflite-*dataset-name*-2019-07-22T21:23:18.861Z`

The folder contains a TensorFlow Lite model named `model.tflite`, a label file named `dict.txt`, and a `tflite_metadata.json` file.



Buckets / [redacted] / export-tflite / model-export / iod / tflite-salad_dataset_20190722113444-2019-07-22T21:23:18.861Z

□	Name	Size	Type	Storage class	Last modified	Public access	Encryption	Retention expiration date	Holds	
□	dict.txt	51 B	application/octet-stream	Regional	7/22/19, 2:23:25 PM UTC-7	Not public	Google-managed key	–	None	⋮
□	model.tflite	2.79 MB	application/octet-stream	Regional	7/22/19, 2:23:25 PM UTC-7	Not public	Google-managed key	–	None	⋮
□	tflite_metadata.json	521 B	application/octet-stream	Regional	7/22/19, 2:23:25 PM UTC-7	Not public	Google-managed key	–	None	⋮

`dict.txt`

Each line in the label file `dict.txt` represents a label of the predictions returned by the TensorFlow Lite model, in the same order they were requested. For example, the `dict.txt` for the salad dataset is as follows:

```
background
Baked Goods
Salad
Cheese
Seafood
Tomato
```

`tflite_metadata.json`

A `tflite_metadata.json` file looks similar to below:

```
{
    "inferenceType": "QUANTIZED_UINT8",
    "inputShape": [
        1,   // This represents batch size
        512,  // This represents image width
        512,  // This represents image height
        3  // This represents inputChannels
    ],
    "inputTensor": "normalized_input_image_tensor",
    "maxDetections": 20,  // This represents max number of boxes.
    "outputTensorRepresentation": [
        "bounding_boxes",
        "class_labels",
        "class_confidences",
        "num_of_boxes"
    ],
    "outputTensors": [
        "TFLite_Detection_PostProcess",
        "TFLite_Detection_PostProcess:1",
        "TFLite_Detection_PostProcess:2",
        "TFLite_Detection_PostProcess:3"
    ]
}
```

Using the exported model

After exporting your model to a Google Cloud Storage bucket you can deploy your AutoML Vision Edge model on Android devices (https://www.tensorflow.org/lite/models/object_detection/overview), iOS devices (https://cloud.google.com/vision/automl/object-detection/docs/tflite-ios-tutorial), or Raspberry Pi 3 (https://www.tensorflow.org/lite/guide/build_rpi).

Export to a container

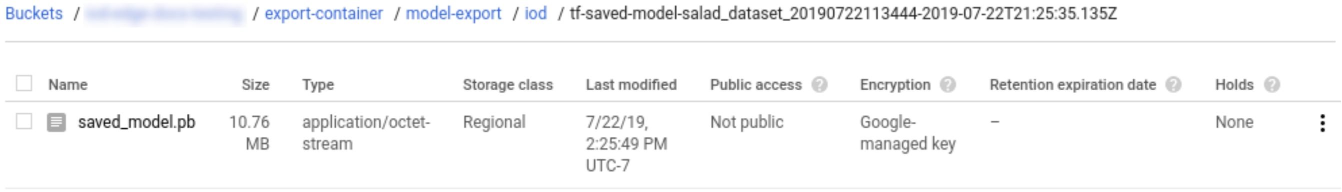
As a result you will see a folder structure in the directory you provided (***cloud-storage-bucket***/[***directory***]). The created folder structure will have the following general format (timestamp in ISO-8601 format):

- `*cloud-storage-bucket*/model-export/iod/*model-type-dataset-name*-YYYY-MM-DDThh:mm:ss.sssZ`

For example:

- `*cloud-storage-bucket*/model-export/iod/tf-saved-model-*dataset-name*-2019-07-22T21:25:35.135Z`
- `*cloud-storage-bucket*/model-export/iod/tflite-*dataset-name*-2019-07-22T21:23:18.861Z`

The folder contains a TensorFlow model named `saved_model.pb`.



Buckets / [redacted] / export-container / model-export / iod / tf-saved-model-salad_dataset_20190722113444-2019-07-22T21:25:35.135Z

□ Name	Size	Type	Storage class	Last modified	Public access	Encryption	Retention expiration date	Holds
□ saved_model.pb	10.76 MB	application/octet-stream	Regional	7/22/19, 2:25:49 PM UTC-7	Not public	Google-managed key	–	None

Using the exported model

After exporting your model to a Google Cloud Storage bucket you can use your exported model to make predictions in a Docker image. See the Containers tutorial (https://cloud.google.com/vision/automl/object-detection/docs/containers-gcs-tutorial) for instructions on deployment to a container.

Export for Web

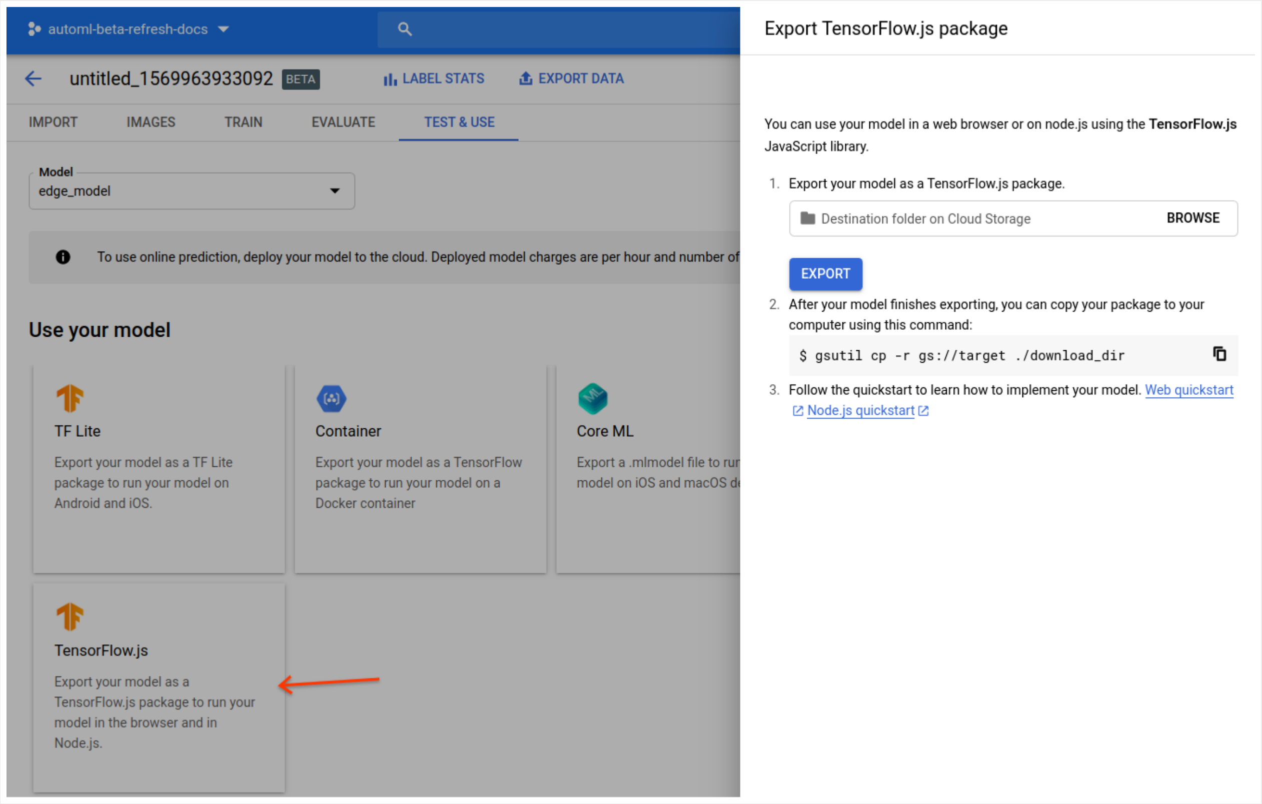
Web UI

Note: Starting September 2019 we will start migrating AutoML Vision users to a new user interface that may affect the steps in this operation. This migration will occur in an on-going basis. See the "**Integrated UI**" tab for instructions using the updated interface.

1. Open the AutoML Vision UI (https://console.cloud.google.com/vision) and select the lightbulb icon in the side navigation bar to display the available models.

To view the models for a different project, select the project from the drop-down list in the upper right of the title bar.

2. Select the row for the model you want to use to label your images.
3. Select the **Test & Use** tab just below the title bar.
4. In the **Use your model** section select the **Tensorflow.js** option. After selecting the **Tensorflow.js** option, select **Export** to export your Web-ready TensorFlow.js model.



To use online prediction, deploy your model to the cloud. Deployed model charges are per hour and number of

TF Lite: Export your model as a TF Lite package to run your model on Android and iOS.

Container: Export your model as a TensorFlow package to run your model on a Docker container

Core ML: Export a .mlmodel file to rur model on iOS and macOS d

TensorFlow.js: Export your model as a TensorFlow.js package to run your model in the browser and in Node.js.

Export TensorFlow.js package

You can use your model in a web browser or on node.js using the **TensorFlow.js** JavaScript library.

1. Export your model as a TensorFlow.js package.

   Destination folder on Cloud Storage

2. After your model finishes exporting, you can copy your package to your computer using this command:

```
$ gsutil cp -r gs://target ./download_dir
```

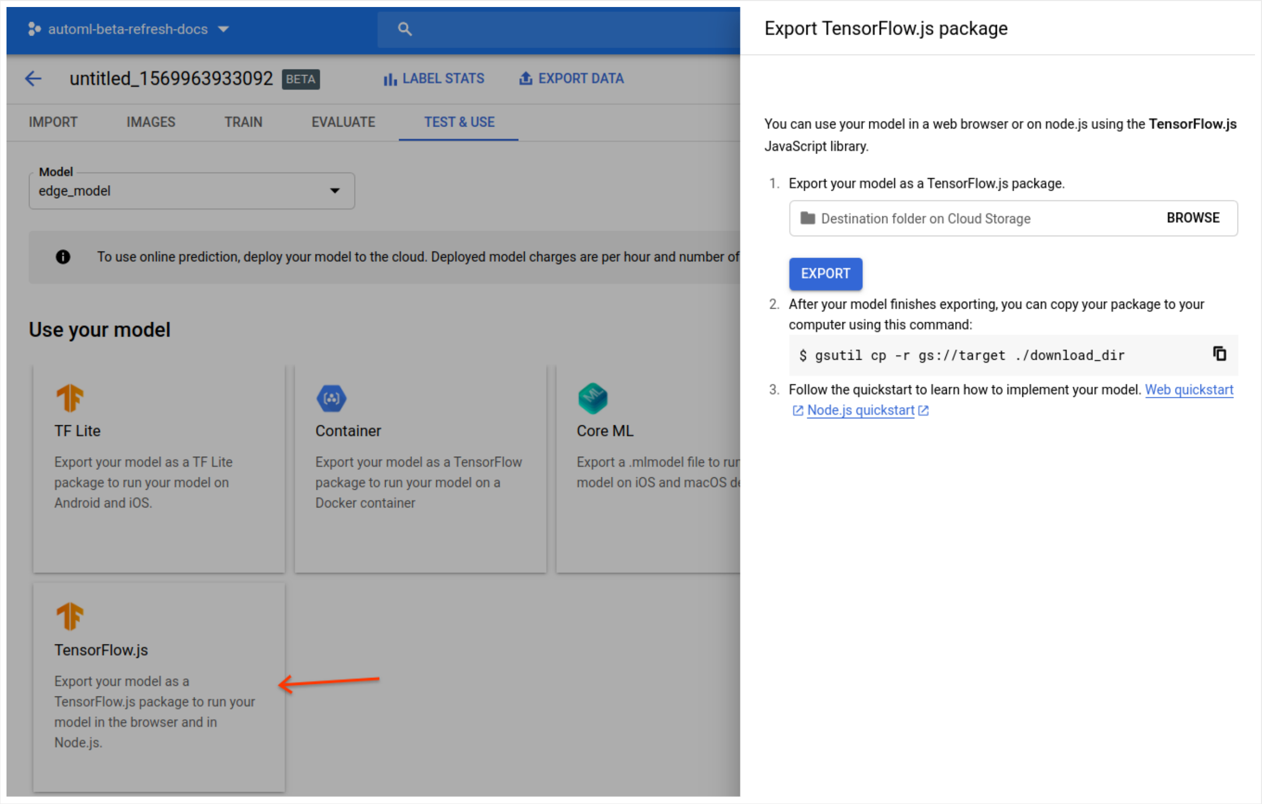
3. Follow the quickstart to learn how to implement your model. Web quickstart Node.js quickstart

Integrated UI

1. Open the Vision Dashboard (https://console.cloud.google.com/vision/dashboard) and select the lightbulb icon in the side navigation bar to display the available models.

   To view the models for a different project, select the project from the drop-down list in the upper right of the title bar.

2. Select the row for the model you want to use to label your images.
3. Select the **Test & Use** tab just below the title bar.
4. In the **Use your model** section select the **Tensorflow.js** option. After selecting the **Tensorflow.js** option, select **Export** to export your Web-ready TensorFlow.js model.



To use online prediction, deploy your model to the cloud. Deployed model charges are per hour and number of

Use your model

TF Lite

Export your model as a TF Lite package to run your model on Android and iOS.

Container

Export your model as a TensorFlow package to run your model on a Docker container

Core ML

Export a .mlmodel file to rur model on iOS and macOS d

TensorFlow.js

Export your model as a TensorFlow.js package to run your model in the browser and in Node.js.

Export TensorFlow.js package

You can use your model in a web browser or on node.js using the **TensorFlow.js** JavaScript library.

1. Export your model as a TensorFlow.js package.

   Destination folder on Cloud Storage BROWSE

   EXPORT

2. After your model finishes exporting, you can copy your package to your computer using this command:

   ```
   $ gsutil cp -r gs://target ./download_dir
   ```

3. Follow the quickstart to learn how to implement your model. Web quickstart Node.js quickstart

REST & CMD LINE

Before using any of the request data below, make the following replacements:

- ***project-id***: your GCP project ID.
- ***model-id***: the ID of your model, from the response when you created the model. The ID is the last element of the name of your model. For example:
  - model name: `projects/`***`project-id`***`/locations/`***`location-id`***`/models/IOD4412217016962778756`
  - model id: `IOD4412217016962778756`
- ***output-storage-bucket***: a Google Cloud Storage bucket/directory to save output files to, expressed in the following form: `gs://bucket/directory/`. The requesting user must have write permission to the bucket.

HTTP method and URL:

```
POST https://automl.googleapis.com/v1/projects/project-id/locations/us-central1/mode
```

Request JSON body:

```
{
  "outputConfig": {
    "modelFormat": "tf_js",
    "gcsDestination": {
      "outputUriPrefix": "output-storage-bucket/"
    },
  }
}
```

To send your request, choose one of these options:

curl

Note: Ensure you have set the `GOOGLE_APPLICATION_CREDENTIALS` (https://cloud.google.com/docs/authentication/production) environment variable to your service account private key file path.

Save the request body in a file called `request.json`, and execute the following command:

```
curl -X POST \
-H "Authorization: Bearer "$(gcloud auth application-default print-access-token) \
-H "Content-Type: application/json; charset=utf-8" \
-d @request.json \
https://automl.googleapis.com/v1/projects/project-id/locations/us-central1/models/mo
```

PowerShell

Note: Ensure you have set the `GOOGLE_APPLICATION_CREDENTIALS` (https://cloud.google.com/docs/authentication/production) environment variable to your service account private key file path.

Save the request body in a file called `request.json`, and execute the following command:

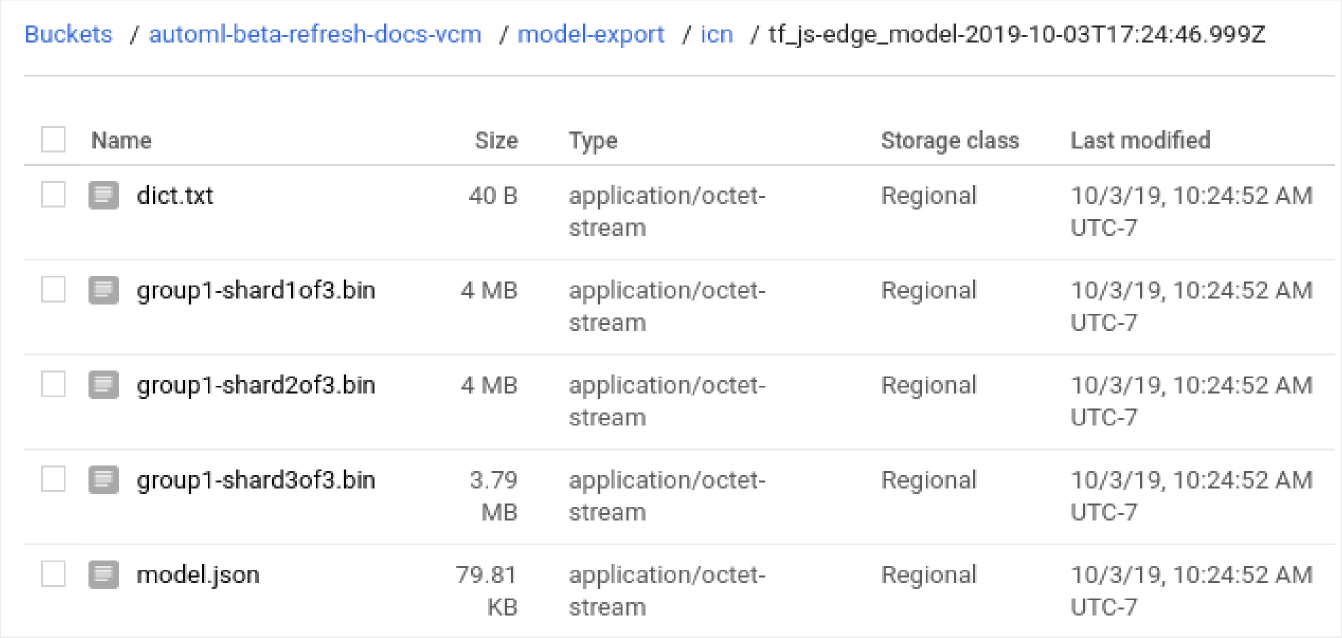
```
$cred = gcloud auth application-default print-access-token
$headers = @{ "Authorization" = "Bearer $cred" }

Invoke-WebRequest `
  -Method POST `
  -Headers $headers `
  -ContentType: "application/json; charset=utf-8" `
  -InFile request.json `
  -Uri "https://automl.googleapis.com/v1/projects/project-id/locations/us-central1/m
```

You should receive a JSON response similar to the following:

```
{
  "name": "projects/project-id/locations/us-central1/operations/operation-id",
  "metadata": {
    "@type": "type.googleapis.com/google.cloud.automl.v1.OperationMetadata",
    "createTime": "2019-07-22T21:23:21.643041Z",
    "updateTime": "2019-07-22T21:23:21.643041Z",
    "exportModelDetails": {
      "outputInfo": {
        "gcsOutputDirectory": "output-storage-bucket/model-export/icn/tf_js-dataset-
      }
    }
  }
}
```

As a result you will see a folder in the directory you provided (${***USER_GCS_PATH***}). The created folder will be named according to timestamp in the format `/model-export/icn/tf_js-dataset-name-YYYY-MM-DDThh:mm:ss.sssZ` (for example, `tf_js-edge_model-2019-10-03T17:24:46.999Z`).

The folder contains binary files (`.bin`), a label file named `dict.txt`, and a `model.json` file.



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

`dict.txt`

Each line in the label file `dict.txt` represents a label of the predictions returned by the TensorFlow.js model, in the same order they were requested. For example, the `dict.txt` for the flowers dataset is as follows:

```
daisy
dandelion
roses
sunflowers
tulips
```

`model.json` (shortened for clarity)

```
{
  "format": "graph-model",
  "generatedBy": "1.14.0",
  "convertedBy": "TensorFlow.js Converter v1.2.9",
  "modelTopology": {
    "node": [
      {
        "name": "image",
        "op": "Placeholder",
        "attr": {
          "shape": {
            "shape": {
              "dim": [
                {
                  "size": "1"
                },
                {
                  "size": "224"
                },
                {
                  "size": "224"
                },
                {
                  "size": "3"
                }
              ]
            }
          },
          "dtype": {
            "type": "DT_FLOAT"
          }
        }
      },
      ...
      {
        "name": "mnas_v4_a/cell_3/op_0/expand_0/tf_layer/kernel",
        "op": "Const",
        "attr": {
          "dtype": {
            "type": "DT_FLOAT"
          },
          "value": {
            "tensor": {
              "dtype": "DT_FLOAT",
              "tensorShape": {
                "dim": [
                  {
                    "size": "1"
                  },
                  {
                    "size": "1"
                  },
                  {
                    "size": "24"
                  },
                  {
                    "size": "72"
                  }
                ]
              }
            }
          }
        }
      },
      {
        "name": "mnas_v4_a_1/feature_network/cell_14/op_0/project_0/Conv2D_bn_offset
        "op": "Const",
        "attr": {
          "value": {
            "tensor": {
              "dtype": "DT_FLOAT",
              "tensorShape": {
                "dim": [
                  {
                    "size": "192"
                  }
                ]
              }
            }
          },
          "dtype": {
            "type": "DT_FLOAT"
          }
        }
      },
      ...
      {
        "name": "mnas_v4_a_1/feature_network/stem/bn/FusedBatchNorm",
        "op": "_FusedConv2D",
        "input": [
          "image",
          "mnas_v4_a/stem/conv/tf_layer/kernel",
          "mnas_v4_a_1/feature_network/stem/conv/Conv2D_bn_offset"
        ],
        "device": "/device:CPU:0",
        "attr": {
          "num_args": {
            "i": "1"
          },
          "explicit_paddings": {
            "list": {}
          },
          "fused_ops": {
            "list": {
              "s": [
                "Qmlhc0FkZA=="
              ]
            }
          },
          "use_cudnn_on_gpu": {
            "b": true
          },
          "padding": {
            "s": "U0FNRQ=="
          },
          "dilations": {
            "list": {
              "i": [
                "1",
                "1",
                "1",
                "1"
              ]
            }
          },
          "epsilon": {
            "f": 0
          },
          "T": {
            "type": "DT_FLOAT"
          },
          "strides": {
            "list": {
              "i": [
                "1",
                "2",
                "2",
                "1"
              ]
            }
          },
          "data_format": {
            "s": "TkhXQw=="
          }
        }
      },
      {
        "name": "mnas_v4_a_1/feature_network/lead_cell_0/op_0/depthwise_0/depthwise"
        "op": "DepthwiseConv2dNative",
        "input": [
          "mnas_v4_a_1/feature_network/stem/bn/FusedBatchNorm",
          "mnas_v4_a/lead_cell_0/op_0/depthwise_0/depthwise_kernel"
        ],
        "attr": {
          "dilations": {
            "list": {
              "i": [
                "1",
                "1",
                "1",
                "1"
              ]
            }
          },
          "strides": {
            "list": {
              "i": [
                "1",
                "1",
                "1",
                "1"
              ]
            }
          },
          "T": {
            "type": "DT_FLOAT"
          },
          "padding": {
            "s": "U0FNRQ=="
          },
          "data_format": {
            "s": "TkhXQw=="
          }
        }
      },
      ...
      {
        "name": "mnas_v4_a_1/feature_network/lead_cell_1/op_0/Relu",
        "op": "_FusedConv2D",
        "input": [
          "mnas_v4_a_1/feature_network/lead_cell_0/op_0/bn2_0/FusedBatchNorm",
          "mnas_v4_a/lead_cell_1/op_0/expand_0/tf_layer/kernel",
          "mnas_v4_a_1/feature_network/lead_cell_1/op_0/expand_0/Conv2D_bn_offset"
        ],
        "device": "/device:CPU:0",
        "attr": {
          "explicit_paddings": {
            "list": {}
          },
          "num_args": {
            "i": "1"
          },
          "fused_ops": {
            "list": {
              "s": [
                "Qmlhc0FkZA==",
                "UmVsdQ=="
              ]
            }
          },
          "use_cudnn_on_gpu": {
            "b": true
          },
          "padding": {
            "s": "U0FNRQ=="
          },
          "dilations": {
            "list": {
              "i": [
                "1",
                "1",
                "1",
                "1"
              ]
            }
          },
          "epsilon": {
            "f": 0
          },
          "T": {
            "type": "DT_FLOAT"
          },
          "strides": {
            "list": {
              "i": [
                "1",
                "1",
                "1",
                "1"
              ]
            }
          },
          "data_format": {
            "s": "TkhXQw=="
          }
        }
      },
      {
        "name": "mnas_v4_a_1/feature_network/lead_cell_1/op_0/depthwise_0/depthwise"
        "op": "DepthwiseConv2dNative",
        "input": [
          "mnas_v4_a_1/feature_network/lead_cell_1/op_0/Relu",
          "mnas_v4_a/lead_cell_1/op_0/depthwise_0/depthwise_kernel"
        ],
        "attr": {
          "strides": {
            "list": {
              "i": [
                "1",
                "2",
                "2",
                "1"
              ]
            }
          },
          "T": {
            "type": "DT_FLOAT"
          },
          "data_format": {
            "s": "TkhXQw=="
          },
          "padding": {
            "s": "U0FNRQ=="
          },
          "dilations": {
            "list": {
              "i": [
                "1",
                "1",
                "1",
                "1"
              ]
            }
          }
        }
      },
      {
        "name": "mnas_v4_a_1/feature_network/lead_cell_1/op_0/bn1_0/FusedBatchNorm",
        "op": "BiasAdd",
        "input": [
          "mnas_v4_a_1/feature_network/lead_cell_1/op_0/depthwise_0/depthwise",
          "mnas_v4_a_1/feature_network/lead_cell_1/op_0/depthwise_0/depthwise_bn_off
        ],
        "attr": {
          "T": {
            "type": "DT_FLOAT"
          },
          "data_format": {
            "s": "TkhXQw=="
          }
        }
      },
      {
        "name": "mnas_v4_a_1/feature_network/lead_cell_1/op_0/Relu_1",
        "op": "Relu",
        "input": [
          "mnas_v4_a_1/feature_network/lead_cell_1/op_0/bn1_0/FusedBatchNorm"
        ],
        "attr": {
          "T": {
            "type": "DT_FLOAT"
          }
        }
      },
      ...
      {
        "name": "mnas_v4_a_1/feature_network/feature_extractor/Mean",
        "op": "Mean",
        "input": [
          "mnas_v4_a_1/feature_network/lead_cell_17/op_0/Relu",
          "mnas_v4_a_1/feature_network/feature_extractor/Mean/reduction_indices"
        ],
        "attr": {
          "T": {
            "type": "DT_FLOAT"
          },
          "keep_dims": {
            "b": false
          },
          "Tidx": {
            "type": "DT_INT32"
          }
        }
      },
      {
        "name": "mnas_v4_a_1/output/fc/MatMul",
        "op": "MatMul",
        "input": [
          "mnas_v4_a_1/feature_network/feature_extractor/Mean",
          "mnas_v4_a/output/fc/tf_layer/kernel"
        ],
        "attr": {
          "transpose_b": {
            "b": false
          },
          "transpose_a": {
            "b": false
          },
          "T": {
            "type": "DT_FLOAT"
          }
        }
      },
      {
        "name": "mnas_v4_a_1/output/fc/BiasAdd",
        "op": "BiasAdd",
        "input": [
          "mnas_v4_a_1/output/fc/MatMul",
          "mnas_v4_a/output/fc/tf_layer/bias"
        ],
        "attr": {
          "T": {
            "type": "DT_FLOAT"
          },
          "data_format": {
            "s": "TkhXQw=="
          }
        }
      },
      {
        "name": "ExpandDims",
        "op": "ExpandDims",
        "input": [
          "mnas_v4_a_1/output/fc/BiasAdd",
          "ExpandDims/dim"
        ],
        "attr": {
          "T": {
            "type": "DT_FLOAT"
          },
          "Tdim": {
            "type": "DT_INT32"
          }
        }
      },
      {
        "name": "Squeeze",
        "op": "Squeeze",
        "input": [
          "ExpandDims"
        ],
        "attr": {
          "T": {
            "type": "DT_FLOAT"
          },
          "squeeze_dims": {
            "list": {
              "i": [
                "0"
              ]
            }
          }
        }
      },
      {
        "name": "Softmax",
        "op": "Softmax",
        "input": [
          "Squeeze"
        ],
        "attr": {
          "T": {
            "type": "DT_FLOAT"
          }
        }
      },
      {
        "name": "scores",
        "op": "Identity",
        "input": [
          "Softmax"
        ],
        "attr": {
          "T": {
            "type": "DT_FLOAT"
          }
        }
      }
    ],
    "library": {},
    "versions": {}
  },
  "weightsManifest": [
    {
      "paths": [
        "group1-shard1of3.bin",
        "group1-shard2of3.bin",
        "group1-shard3of3.bin"
      ],
      "weights": [
        {
          "name": "mnas_v4_a_1/feature_network/feature_extractor/Mean/reduction_indi
          "shape": [
            2
          ],
          "dtype": "int32"
        },
        ...
        {
          "name": "mnas_v4_a/cell_14/op_0/expand_0/tf_layer/kernel",
          "shape": [
            1,
            1,
            192,
            1152
          ],
          "dtype": "float32"
        },
        {
          "name": "mnas_v4_a_1/feature_network/cell_14/op_0/expand_0/Conv2D_bn_offse
          "shape": [
            1152
          ],
          "dtype": "float32"
        },
        {
          "name": "mnas_v4_a_1/feature_network/lead_cell_17/op_0/conv2d_0/Conv2D_bn_
          "shape": [
            1280
          ],
          "dtype": "float32"
        }
      ]
    }
  ]
}
```

Last updated January 22, 2020.