

[Cloud AutoML Vision](#)

# Edge device model quickstart

This quickstart walks you through the process of:

- Copying a set of images into Google Cloud Storage.
- Creating a CSV listing the images and their labels.
- Using AutoML Vision to create your dataset, train a custom AutoML Vision Edge model, and make a prediction.
- Exporting and deploying your AutoML Vision Edge model.

## Before you begin

### Set up your project

1. [Sign in](https://accounts.google.com/Login) (https://accounts.google.com/Login) to your Google Account.

If you don't already have one, [sign up for a new account](https://accounts.google.com/SignUp) (https://accounts.google.com/SignUp).

2. In the Cloud Console, on the project selector page, select or create a Google Cloud project.

★ **Note:** If you don't plan to keep the resources that you create in this procedure, create a project instead of selecting an existing project. After you finish these steps, you can delete the project, removing all resources associated with the project.

[GO TO THE PROJECT SELECTOR PAGE](https://console.cloud.google.com/projectselect) (HTTPS://CONSOLE.CLOUD.GOOGLE.COM/PROJECTSELECT)

3. Make sure that billing is enabled for your Google Cloud project. [Learn how to confirm billing is enabled for your project](https://cloud.google.com/billing/docs/how-to/modify-project) (https://cloud.google.com/billing/docs/how-to/modify-project).
4. Enable the AutoML and Cloud Storage APIs.

[ENABLE THE APIS](https://console.cloud.google.com/flows/enableapi?APIID=STORAGE) (HTTPS://CONSOLE.CLOUD.GOOGLE.COM/FLOWS/ENABLEAPI?APIID=STORAGE-)

5. [Install the gccloud command line tool](https://cloud.google.com/sdk/downloads#interactive) (https://cloud.google.com/sdk/downloads#interactive).

- Follow the instructions to [create a service account and download a key file](https://cloud.google.com/iam/docs/creating-managing-service-accounts#creating_a_service_account) (https://cloud.google.com/iam/docs/creating-managing-service-accounts#creating\_a\_service\_account) for that account.

★ Service accounts are the only authentication option available with the AutoML API.

- Set the `GOOGLE_APPLICATION_CREDENTIALS` environment variable to the path to the service account key file that you downloaded when you created the service account.

```
export GOOGLE_APPLICATION_CREDENTIALS=key-file
```



- Set the `PROJECT_ID` environment variable to your [Project ID](https://cloud.google.com/resource-manager/docs/creating-managing-projects#identifying_projects) (https://cloud.google.com/resource-manager/docs/creating-managing-projects#identifying\_projects)

```
export PROJECT_ID=your-project-id
```



The AutoML API calls and resource names include your Project ID in them. The `PROJECT_ID` environment variable provides a convenient way to specify the ID.

- If you are an owner for your project, add your service account to the **AutoML Editor** IAM role, replacing **service-account-name** with the name of your new service account. For example, `service-account1@myproject.iam.gserviceaccount.com`.

```
gcloud auth login
gcloud projects add-iam-policy-binding $PROJECT_ID \
  --member="serviceAccount:service-account-name" \
  --role="roles/automl.editor"
```



- Otherwise (if you are not a project owner), ask a project owner to add both your user ID and your service account to the **AutoML Editor** IAM role.

## Create a Cloud Storage bucket

Use Cloud Shell, a browser-based Linux command line connected to your Cloud Console project, to create your Cloud Storage bucket:

1. Open Cloud Shell (<http://console.cloud.google.com/?cloudshell=true>).
2. Create a Google Cloud Storage bucket. The bucket name must be in the format: ***project-id-vcm***. The following command creates a storage bucket in the **us-central1** region named ***project-id-vcm***. For a complete list of available regions, see the Bucket Locations page ([https://cloud.google.com/storage/docs/locations#available\\_locations](https://cloud.google.com/storage/docs/locations#available_locations)).

```
gsutil mb -p project-id -c regional -l us-central1 gs://project-id-vcm/
```

Recommended file structure for your Cloud Storage files:

```
gs://project-id-vcm/dataset-name/documents/document-name.txt
```

## Copy the sample images into your bucket

Next, copy the flower dataset used in [this Tensorflow blog post](#)

(<https://cloud.google.com/blog/big-data/2016/12/how-to-classify-images-with-tensorflow-using-google-cloud-machine-learning-and-cloud-dataflow>)

. The images are stored in a public Cloud Storage bucket, so you can copy them directly from there to your own bucket.

1. In your Cloud Shell session, enter:

```
gsutil -m cp -R gs://cloud-ml-data/img/flower_photos/ gs://${BUCKET}/img/
```

The file copying takes about 20 minutes to complete.

## Create the CSV file

The sample dataset contains a CSV file with all of the image locations and the labels for each image. You'll use that to create your own CSV file:

1. Update the CSV file to point to the files in your own bucket:

```
gsutil cat gs://${BUCKET}/img/flower_photos/all_data.csv | sed "s:cloud-ml-data
```

2. Copy the CSV file into your bucket:

```
gsutil cp all_data.csv gs://{BUCKET}/csv/
```

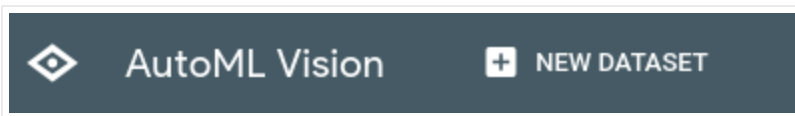


## Create your dataset

Visit the [AutoML Vision UI](https://console.cloud.google.com/vision) (<https://console.cloud.google.com/vision>) to begin the process of creating your dataset and training your model.

When prompted, make sure to select the project that you used for your Cloud Storage bucket.

1. From the AutoML Vision page, click **New Dataset**:



2. Specify a name for this dataset. Click the + sign to continue.

A screenshot of the 'Create dataset' form in the AutoML Vision UI. The form has a title 'Create dataset' and a text input field labeled 'Dataset name' with a question mark icon to its right. A blue horizontal line is visible at the bottom of the form.

3. Specify the Cloud Storage URI of your CSV file. For this quickstart, the CSV file is at `gs://your-project-123-vcv/csv/all_data.csv`. Make sure to replace `your-project-123` with your specific project ID.
4. Click **Create Dataset**. The import process takes a few minutes. When it completes, you are taken to the next page which has details on all of the images identified for your dataset, both labeled and unlabeled images. You can filter images by label by selecting a label under **Filter labels**. If you are using the flower dataset, you will see a warning alert which will notify you of repeated images or images with multiple labels (if multi-label is not enabled).

	IMAGES	TRAIN	EVALUATE	PREDICT
	All images	0	Filter...	
	Labeled	0		
	Unlabeled	0		
	Filter labels			
You have no labels				
<a href="#">ADD LABEL</a>				


- You can add additional images and update labels for new and existing images after you have imported a CSV file.

## Train your model


- Once your dataset has been created and processed, select the **Train** tab to initiate model training.

IMAGES	TRAIN	EVALUATE	PREDICT
All images	3667	Type to filter images...	
Labeled	3667		
Unlabeled	0		
Type to filter...			
daisy	633		
dandelion	898		
roses	641		
sunflowers	697		
tulips	798		
<a href="#">Add label</a>			


Select all images



roses

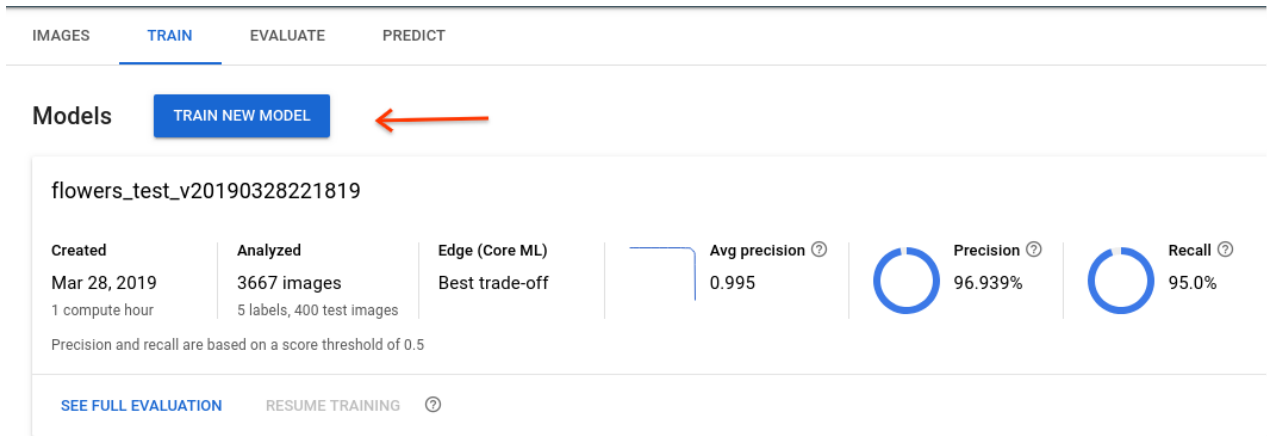


roses



roses

## 2. Select **TRAIN NEW MODEL** to continue.



The screenshot shows the Google Cloud AutoML Vision interface. At the top, there are four tabs: IMAGES, TRAIN, EVALUATE, and PREDICT. The TRAIN tab is selected. Below the tabs, there is a 'Models' section with a blue button labeled 'TRAIN NEW MODEL' and a red arrow pointing to it. Below the button, there is a card for a model named 'flowers\_test\_v20190328221819'. The card displays the following information:

Created	Analyzed	Edge (Core ML)	Avg precision	Precision	Recall
Mar 28, 2019 1 compute hour	3667 images 5 labels, 400 test images	Best trade-off	0.995	96.939%	95.0%

Precision and recall are based on a score threshold of 0.5

At the bottom of the card, there are two buttons: 'SEE FULL EVALUATION' and 'RESUME TRAINING' with a help icon.

This will open a pop-up window with training options.

- From the training pop-up window, select **1. "Edge"** from the **Model type**. Then select model optimized for **2. "Best trade-off"** and your **3.** node hour budget.

## Train new model

### Model name

flowers\_test\_v20190402214836

### Model type

Cloud-hosted

Host your model on Google Cloud for online predictions.

Edge  1.

Download your model for offline/mobile use. Typically has lower accuracy than Cloud-hosted models.

Format model for Core ML (iOS / macOS)

2.  


### Optimize model for:

#### Lowest latency

Latency: 22 msec

Size: 557 KB

Accuracy: Typically lower

#### Best trade-off

Latency: 65 msec

Size: 3.1 MB

Accuracy: Best trade-off

#### Higher accuracy

Latency: 105 msec

Size: 5.6 MB

Accuracy: Typically higher

### Show latency estimates for

Google Pixel 1 

Please note that prediction latency estimates are for guidance only. Actual latency will depend on your target device and environment setup.


### Set a node hour budget

Your model's accuracy generally depends on how long you allow it to train, and the quality of your dataset. Your model automatically stops training when it stops improving. You pay only for the node hours used.

5 node hours (recommended) 



 3.

Models are based on [state-of-the-art research](#)  at Google. Your model will be available as a TF Lite package.

CANCEL

START TRAINING

#### 4. Select **"Start training"** to begin model training.

Training is initiated for your model, and should take about an hour. The training might stop earlier than the node hour you selected. The service will email you once training has completed, or if any errors occur.

Once training is complete, you can refer to evaluation metrics, as well as test and use the model.

Select the **Evaluate** tab to get more details on [F1, Precision, and Recall](https://cloud.google.com/vision/automl/docs/evaluate)

(<https://cloud.google.com/vision/automl/docs/evaluate>) scores.

Select an individual label under **Filter labels** to get details on true positives, false negatives and false positives.

## Make a Prediction

Select the **Predict** tab for instructions on sending an image to your model for a prediction. You can also refer to [Annotating images](https://cloud.google.com/vision/automl/docs/predict) (<https://cloud.google.com/vision/automl/docs/predict>) for examples.

## Export and Deploy the Edge model

The final step in using an AutoML Vision Edge model is to export (optimize and download) and deploy (use) your model.

There are multiple ways you can export and deploy your models to use for prediction on Edge devices.

**Model and export options:** [Tensorflow Lite](https://cloud.google.com/vision/automl/docs/export-edge#tensorflow_lite_models)

([https://cloud.google.com/vision/automl/docs/export-edge#tensorflow\\_lite\\_models](https://cloud.google.com/vision/automl/docs/export-edge#tensorflow_lite_models)) is optimized for mobile phones and ARM (mobile device operating system) devices. It can further be compiled to run on [Edge TPU](https://cloud.google.com/vision/automl/docs/export-edge#export_to_edge_tpu) ([https://cloud.google.com/vision/automl/docs/export-edge#export\\_to\\_edge\\_tpu](https://cloud.google.com/vision/automl/docs/export-edge#export_to_edge_tpu)). Another export option - [Tensorflow SavedModel](https://cloud.google.com/vision/automl/docs/export-edge#deployment_to_a_container)

([https://cloud.google.com/vision/automl/docs/export-edge#deployment\\_to\\_a\\_container](https://cloud.google.com/vision/automl/docs/export-edge#deployment_to_a_container)) - can be used in a [Docker container](https://cloud.google.com/vision/automl/docs/containers-gcs-tutorial) (<https://cloud.google.com/vision/automl/docs/containers-gcs-tutorial>) for serving. Lastly,



a [Core ML](https://cloud.google.com/vision/automl/docs/export-edge#core_ml_models) ([https://cloud.google.com/vision/automl/docs/export-edge#core\\_ml\\_models](https://cloud.google.com/vision/automl/docs/export-edge#core_ml_models)) model is specially optimized for iOS apps.

In this quickstart you will use Tensorflow Lite (TF Lite) as an example. TF Lite models are both easy to use and have a wide set of use cases.

1. Under **Use your Edge model**, select the **TFLite** tab.

#### Use your Edge model

TF LITE   CONTAINER   EDGE DEVICES   GOOGLE CLOUD

1. Download a sample camera app from the TF Lite documentation site: [Android](#) [iOS](#)

2. Export your model as a TF Lite package.

Destination folder on Cloud Storage

[gs://\[redacted\]-vcm/models/edge/ICN4182253562634949954/](gs://[redacted]-vcm/models/edge/ICN4182253562634949954/)

EXPORT

After your model finishes exporting, you can download it from Cloud Storage. Follow the sample app's instructions to learn how to implement your model on your device.

2. Select **Export** to export a TF Lite package into your Cloud Storage storage bucket. The export process typically takes several minutes.
3. After the export is completed, select the Cloud Storage link, and it will direct you to the Google Cloud Storage destination folder in the Google Cloud Platform Console.

In the Google Cloud Storage destination location you will find a folder named with timestamp and model format, under which you can find the following files:

- a tflite file (`model.tflite`),
- a dictionary file (`dict.txt`)
- a metadata file (`tflite_metadata.json`)

## What's Next

With these files, you can follow tutorials to deploy on [Android devices](https://cloud.google.com/vision/automl/docs/tflite-android-tutorial) (<https://cloud.google.com/vision/automl/docs/tflite-android-tutorial>), [iOS devices](https://cloud.google.com/vision/automl/docs/tflite-ios-tutorial) (<https://cloud.google.com/vision/automl/docs/tflite-ios-tutorial>), [Raspberry Pi 3](https://cloud.google.com/vision/automl/docs/tflite-raspberry-pi-3)

([https://www.tensorflow.org/lite/guide/build\\_rpi](https://www.tensorflow.org/lite/guide/build_rpi)), or the [Web](https://cloud.google.com/vision/automl/docs/tensorflow-js-tutorial)  
(<https://cloud.google.com/vision/automl/docs/tensorflow-js-tutorial>).

## Other model use options

- You can also export the model as TensorFlow SavedModel and use it with a Docker container in the **Container** tab. See the [container tutorial](https://cloud.google.com/vision/automl/docs/containers-gcs-tutorial) (<https://cloud.google.com/vision/automl/docs/containers-gcs-tutorial>) on how to export to a container.
- You can export the model for running on Edge TPU in the **Edge devices** tab. Then follow Coral's official documentation about how to [run an inference on the Edge TPU](https://coral.ai/docs/edgetpu/tflite-python/) (<https://coral.ai/docs/edgetpu/tflite-python/>).
- You can check  **Format model for Core ML (iOS / macOS)** before training the model for training a CoreML supported model. After training, you can export the model in **CoreML** tab, and follow the [CoreML tutorial](https://cloud.google.com/vision/automl/docs/tflite-coreml-ios-tutorial) (<https://cloud.google.com/vision/automl/docs/tflite-coreml-ios-tutorial>).

---

*Except as otherwise noted, the content of this page is licensed under the [Creative Commons Attribution 4.0 License](https://creativecommons.org/licenses/by/4.0/) (<https://creativecommons.org/licenses/by/4.0/>), and code samples are licensed under the [Apache 2.0 License](https://www.apache.org/licenses/LICENSE-2.0) (<https://www.apache.org/licenses/LICENSE-2.0>). For details, see our [Site Policies](https://developers.google.com/terms/site-policies) (<https://developers.google.com/terms/site-policies>). Java is a registered trademark of Oracle and/or its affiliates.*

*Last updated December 26, 2019.*