



Facial Recognition on Android Using Google Cloud AutoML

Andrew Kelly - Google Developer Expert for Android

@andrew_ke11y

<https://medium.com/devnibbles>

Steps

1. Detect a face
2. Classify the face
3. ...
4. Profit



Actual Steps

1. Checkout base Android app - make sure it runs.
2. Setup Firebase and Google Cloud project
3. Upload photos to Google Cloud AutoML
4. Add face detection to Android app
5. Train model in Google Cloud AutoML
6. Use Google Cloud AutoML REST or SDK from the Android app
7. ...
8. Profit

Running the Android App

GitHub Project

- We have a GitHub starter repository that's done some of the boring work for you.
 - Requesting Permissions
 - Basic Architecture
- <https://github.com/apkelly/DroidCon-Boston-2019>
- Or point your git client to the repository on the USB stick
- Let's open Android Studio and explore this starter project together.
- When setup successfully you should see a camera preview inside the app.

Setup Firebase

[Go to docs](#)

Welcome to Firebase!

Tools from Google for developing great apps, engaging with your users and earning more through mobile ads.

[Learn more](#) [Documentation](#) [Support](#)

Recent projects

[Add project](#)[Explore a demo project](#)

DroidCon Boston 2019

droidcon-boston-2019



DevNibbles

devnibbles



<https://console.firebaseio.google.com/>

Add a project

Project name

Tip: Projects span apps across platforms [\(?\)](#)

Project ID [\(?\)](#)
droidcon-boston-2019 

Analytics location [\(?\)](#)

Cloud Firestore location [\(?\)](#)

i Cloud Functions are not yet available in this location. If you deploy Cloud Firestore and Cloud Functions to different locations, traffic between them will be billed and latency will increase. If you might use these products together, we suggest selecting a different Cloud Firestore location.

Use the default settings for sharing Google Analytics for Firebase data

- Share your Analytics data with all Firebase features
- Share your Analytics data with Google to improve Google Products and Services
- Share your Analytics data with Google to enable technical support
- Share your Analytics data with Google to enable Benchmarking
- Share your Analytics data with Google Account Specialists

I accept the [controller-controller terms](#). This is required when sharing Analytics data to improve Google Products and Services. [Learn more](#)

[Cancel](#) [Continue](#)

Customise data sharing for your new project

Data you collect, process and store using Google Analytics ('Google Analytics data') is secure and kept confidential. This data is used to maintain and protect the Google Analytics service, to perform system critical operations and in rare exceptions for legal reasons as described in our [privacy policy](#).

The data sharing options give you more control over sharing your Google Analytics data. [Learn more](#)

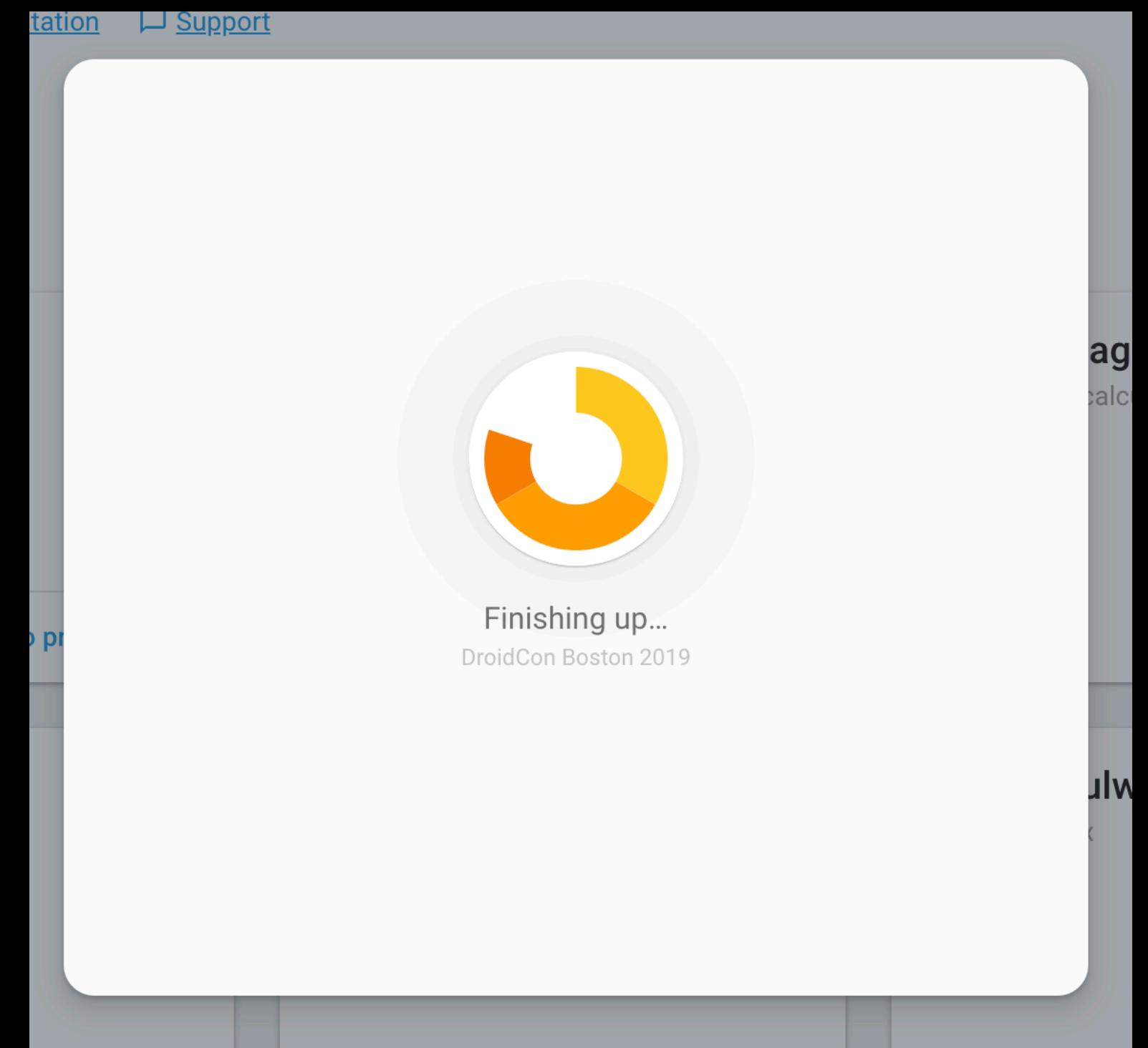
Share your Analytics data with all Firebase features
Share Analytics data with all Firebase features, including Crashlytics, Predictions, A/B Testing, Remote Config, Cloud Messaging and In-App Messaging. If you disable this option, Firebase will not be able to use your Analytics data for these features. [Learn more](#)

Google products and services
Share Google Analytics data with Google to help improve Google's products and services. If you disable this option, data can still flow to other Google products that are explicitly linked to your Google Analytics property.
 I accept the [controller-controller terms](#). [Learn more](#)

Benchmarking
Contribute anonymous data to an aggregate data set to enable features like benchmarking and publications that can help you understand data trends. All identifiable information about your website is removed and combined with other anonymous data before it is shared with others.

Technical support
Let Google technical support representatives access your Google Analytics data

[Create project](#)



The screenshot shows the Firebase Project Overview page for the project "DroidCon Boston 2019". The left sidebar lists various services: Authentication, Database, Storage, Hosting, Functions, and ML Kit. The main area displays the "Develop" section, which includes tabs for Authentication, Database, Storage, Hosting, Functions, and ML Kit. A dropdown menu is open over the "Project settings" tab, showing options like "Project settings", "Users and permissions", and "Cloud messaging".

The screenshot shows the "Your apps" section of the Firebase Project Overview. It displays the message "There are no apps in your project" and "Select a platform to get started". Below this, there are three circular icons: iOS, Android, and a browser icon with code symbols (</>).

The screenshot shows the "Add Firebase to your Android app" dialog. It is a step-by-step guide for registering an app. Step 1, "Register app", is active, showing the "Android package name" field containing "com.droidcon.boston2019.facialrecognition". Step 2, "Download config file", Step 3, "Add Firebase SDK", and Step 4, "Run your app to verify installation", are shown below. A note at the bottom states: "Required for Dynamic Links, Invites and Google Sign-In or phone-number support in Auth. Edit SHA-1s in Settings."

Package name is important - must match android app

Setup Google Cloud

Finish setting up your Google Cloud project

! Let's grant AutoML Vision access to your project. Due to recent changes on our side, you may notice we also require roles/serviceusage.serviceUsageAdmin. This allows us to check your project is setup ahead of time correctly.

You'll only have to do these steps once for your project.

1. Enable billing

You'll need to enable billing for your Google Cloud project to create custom models.

[GO TO BILLING](#)**2. Enable the required APIs and modify permissions**

Clicking "Set up now" will also create a bucket on Google Cloud Storage to store your models' images. You can also do this process manually.

[SET UP NOW](#)[MANUAL SETUP ▾](#)[CHECK AGAIN](#)[SELECT DIFFERENT PROJECT](#)

Upload Images to Google Cloud AutoML

Create Dataset

The screenshot shows the AutoML Vision BETA interface. At the top, there is a navigation bar with the AutoML Vision logo and the word "BETA". To the right of the logo is a "NEW DATASET" button with a plus sign icon. On the left side of the main content area, there is a sidebar with three icons: a list icon labeled "Datasets", a question mark icon, and another list icon. The main content area has a large circular placeholder image with horizontal stripes, centered on the page. Below this image, the text "You have no datasets" is displayed in bold. Underneath that, a smaller text says "Click "New Dataset" above". At the bottom of the page, there is a section titled "Don't need a custom solution?". It contains a link to "Google's Vision API" and a brief description stating that it can handle generalized image labeling, face, logo and landmark detection, and more.

AutoML Vision BETA

+ NEW DATASET

Datasets

You have no datasets

Click "New Dataset" above

Don't need a custom solution?

Google's [Vision API](#) can handle generalized image labeling; face, logo and landmark detection; and more.

<https://cloud.google.com/automl/ui/vision/>



Create dataset



Dataset name
devnibbles_faces



Import images

To build a custom model, you first need to import a set of images to train it. Generally the more images the better. Each image should be categorized with a label (labels are essential for telling the model how to identify an image).

Processed images will be stored on Cloud Storage.



Data preparation tip

Each label should have at least 100 images for best results. To help put together the best dataset for your use case, [read our data guidelines](#)

 Upload images from your computer ?

Supports JPG, PNG, ZIP.

2018-05-01:10-35-40.jpg, 2018-05-01:15-42-22.jpg, 2018-05-01:17-54-17.jpg, 2018-05-0...



[SELECT FILES](#)

 Select a CSV file on Cloud Storage ?

The [CSV file](#) should be a list of paths to your images on GCS and their labels, if available.

gs://devnibbles-vcm/

 Import images later

In the next step, you can add images and label them

Classification type

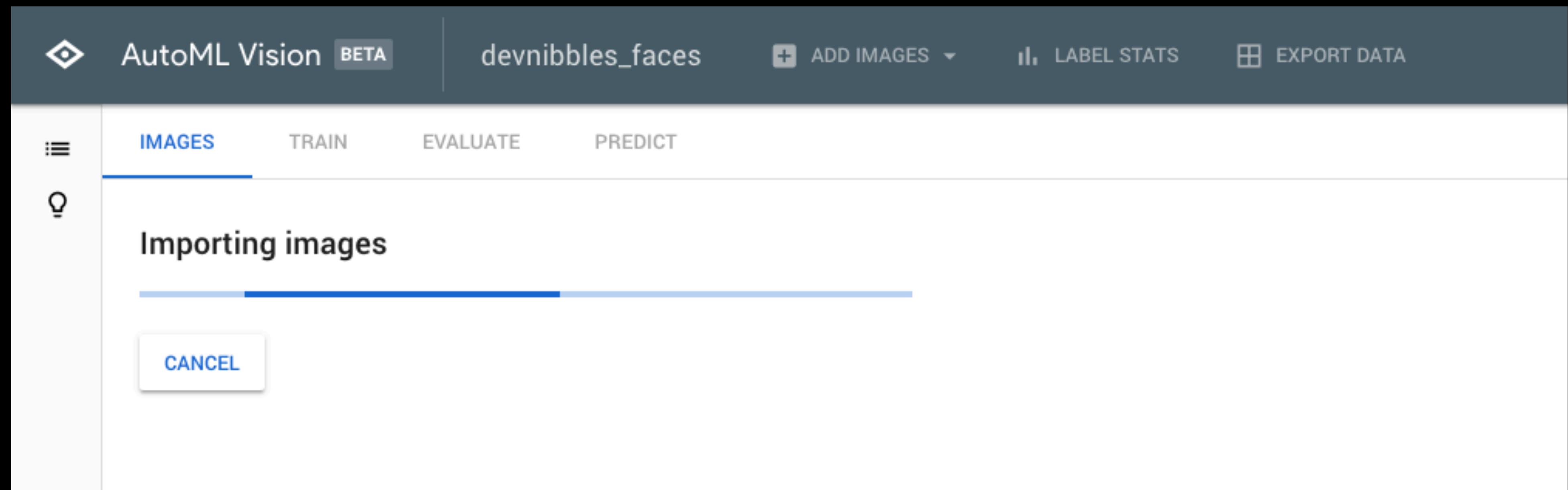
 Enable multi-label classification

If you have images that may require multiple labels, enable this setting now. Typically requires more training images per label to get good model results.

[CREATE DATASET](#)

[CANCEL](#)

If you have a large dataset, I'd recommend using the Cloud Storage option with a CSV of labels.



No need to watch this screen, go get a cup of tea, you'll be sent an email once the import is finished.

Detect Faces in Android App

MLKitActivity.kt

```
import com.droidcon.boston2019.facialrecognition.detect.mlkit.FaceDetector
import com.droidcon.boston2019.facialrecognition.detect.mlkit.FaceGraphic
import com.droidcon.boston2019.facialrecognition.detect.mlkit.FrameMetadata
import com.droidcon.boston2019.facialrecognition.detect.mlkit.MLCameraSource
import com.google.firebase.ml.vision.face.FirebaseVisionFace
import java.nio.ByteBuffer

    override fun createCameraSource() {
        mCameraSource = MLCameraSource(this, mGraphicOverlay)
        mCameraSource?.apply {
            setFrameDetector(
                FaceDetector(object : FaceDetector.DetectorCallback {
                    override fun onSuccess(
                        frameData: ByteBuffer,
                        results: List<FirebaseVisionFace>,
                        frameMetadata: FrameMetadata) {

                        if (results.isEmpty()) {
                            // No faces in frame, so clear frame of any previous faces.
                            mGraphicOverlay.clear()
                        } else {
                            // We have faces
                            results.forEach { face ->
                                val existingFace = mGraphicOverlay
                                    .find(face.trackingId) as FaceGraphic?

                                if (existingFace == null) {
                                    // A new face has been detected.
                                    val faceGraphic = FaceGraphic(
                                        face.trackingId,
                                        mGraphicOverlay
                                    )

                                    mGraphicOverlay.add(faceGraphic)
                                } else {
                                    // We have an existing face, update its position.
                                    existingFace.updateFace(face)
                                }
                            }
                            mGraphicOverlay.postInvalidate()
                        }
                    }
                })
            )
        }
    }

    override fun onFailure(exception: Exception) {
        exception.printStackTrace()
    }
}
```

Enter this code and then
run the app again.

Train Facial Recognition Model



AutoML Vision

BETA

devnibbles_faces

+ ADD IMAGES

LABEL STATS

EXPORT DATA

IMAGES TRAIN EVALUATE PREDICT

	All images	86
Labeled	0	
Unlabeled	86	
Type to filter... :		
Andy	0	
Cameron	0	
Joel	0	
Julien	0	
Ruth	0	
Add label		

4 selected

Label

Type to filter...

Select all images

Andy

Cameron

Joel

Julien

Ruth

Unlabel

Create labels for each of the faces in your dataset using the “Add label” button, you can then apply labels on groups of images.



IMAGES

TRAIN

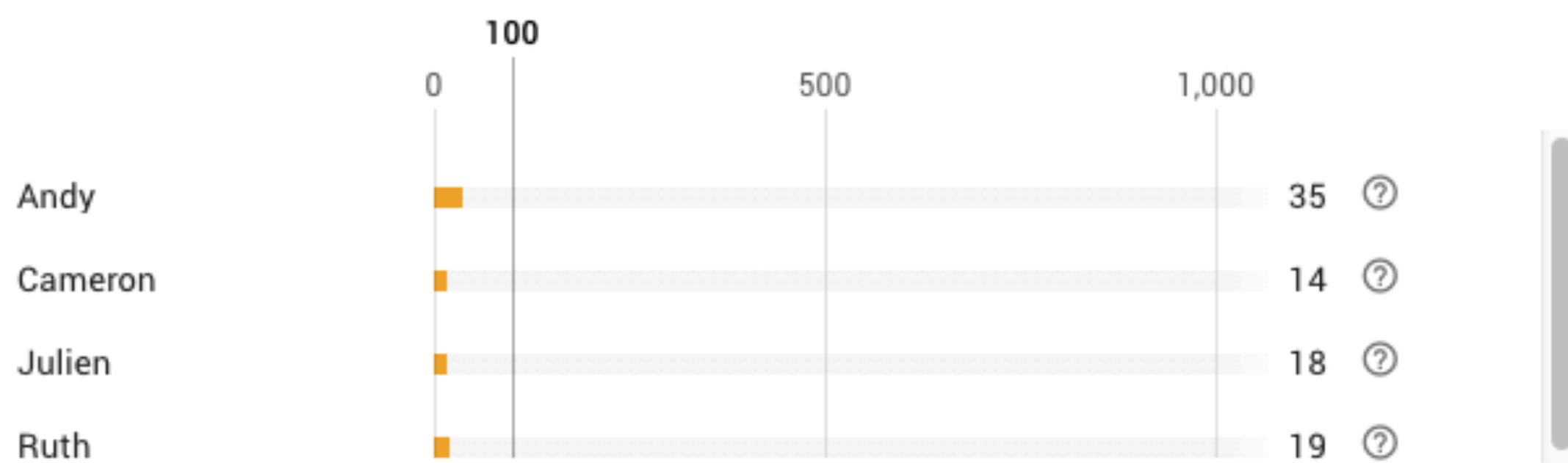
EVALUATE

PREDICT



Try labeling more images before training

Each label should have at least 100 images assigned. Fewer images often result in inaccurate precision and recall scores. [Learn more](#) ↗



Your images will be automatically split into [training and test sets](#) ↗, so you can evaluate your model's performance. Unlabeled images will not be used.

Training images	69
Validation images	8
Test images	9

[START TRAINING](#)

If your dataset is small, you'll be warned that you should import and label more images.

Train new model

Model name

devnibbles_faces_v20190121012135

The accuracy of your model generally depends on how long you allow it to train, and the quality of your dataset. To train your model for more than one compute hour, your dataset needs at least 1000 labeled images.

Training budget

1 compute hour (free*)



Data summary

86 labeled images, 4 labels

* Your first compute hour is free, for up to 10 models each month. [Pricing guide](#)

CANCEL

START TRAINING

Train your model, small datasets should easily complete within the free hour, you'll need to enable Billing on your account though.



Training vision classification model

Training can take 15 minutes to several hours or more, depending on the compute hours assigned. In the meantime, you can close this window. You will be emailed once training completes.

[CANCEL](#)

Go and get your second cup of tea, you'll receive an email once training is complete.



IMAGES

TRAIN

EVALUATE

PREDICT



Model

devnibbles_faces_v20190121012135

Created

1 compute hour

Analyzed

86 images

4 labels, 9 test images

Avg precision ⓘ

1.0

Precision ⓘ

100.0%

Recall ⓘ

100.0%

Precision and recall are based on a score threshold of 0.5

Type to filter labels...

All labels

Andy

Cameron

Julien

Ruth

All labels

Score threshold ⓘ

 0.50

Total images

86

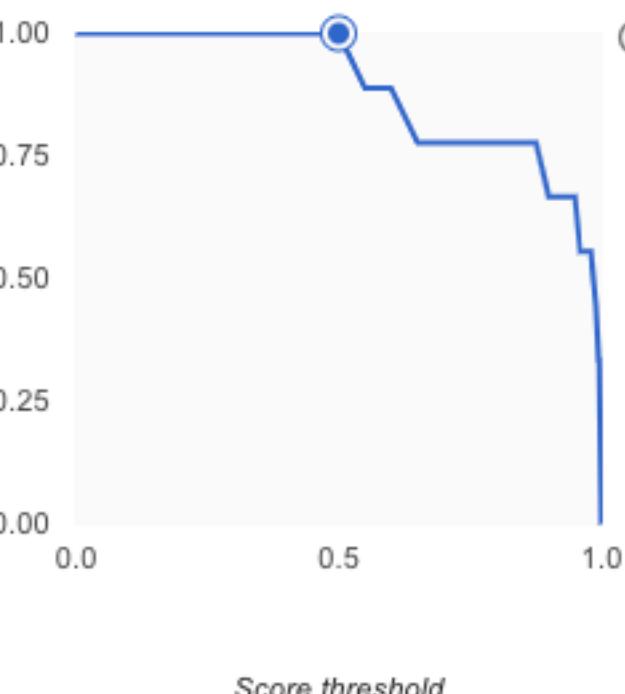
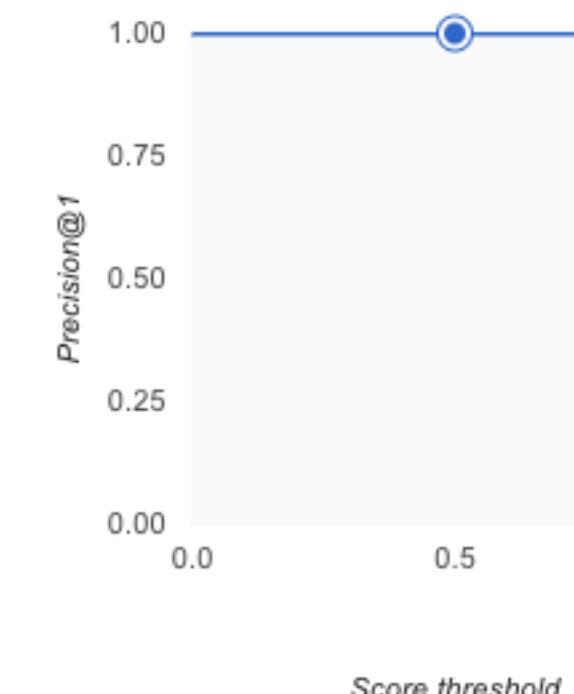
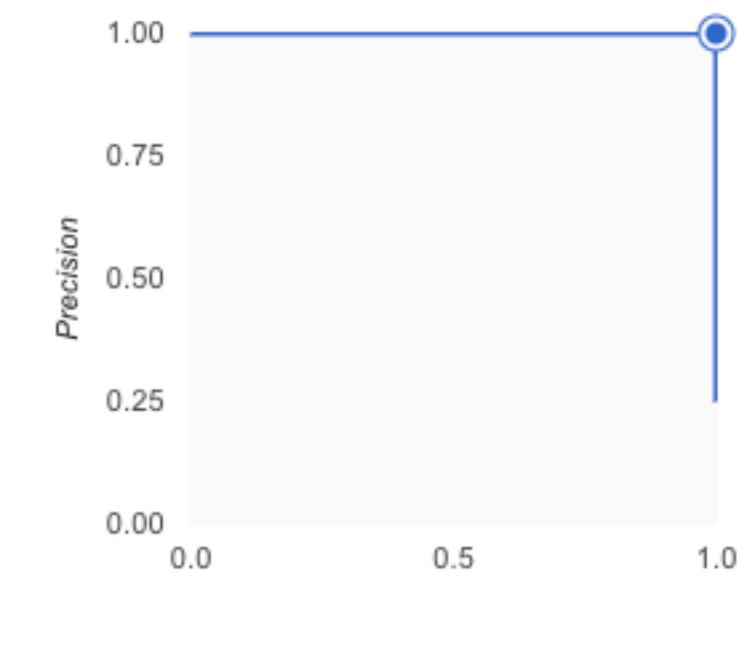
Precision ⓘ

100.0%

Recall ⓘ

100.0%

Use the slider to see which score threshold works best for your model on the precision-recall tradeoff curve. [Learn more about these metrics and graphs](#)



Confusion matrix

This table shows how often the model classified each label correctly (in blue), and which labels were misclassified (in grey).

True label

Andy

Cameron

Julien

Ruth

		Predicted label			
		Andy	Cameron	Julien	Ruth
True label	Andy	100.0%	-	-	-
	Cameron	-	100.0%	-	-
Julien	-	-	100.0%	-	-
Ruth	-	-	-	100.0%	-



IMAGES

TRAIN

EVALUATE

PREDICT



Model

devnibbles_faces_v20190121012135



Test your model on new images

If your model will be used to make predictions on people, test your model on images that capture the diversity of your userbase. [Learn more](#)



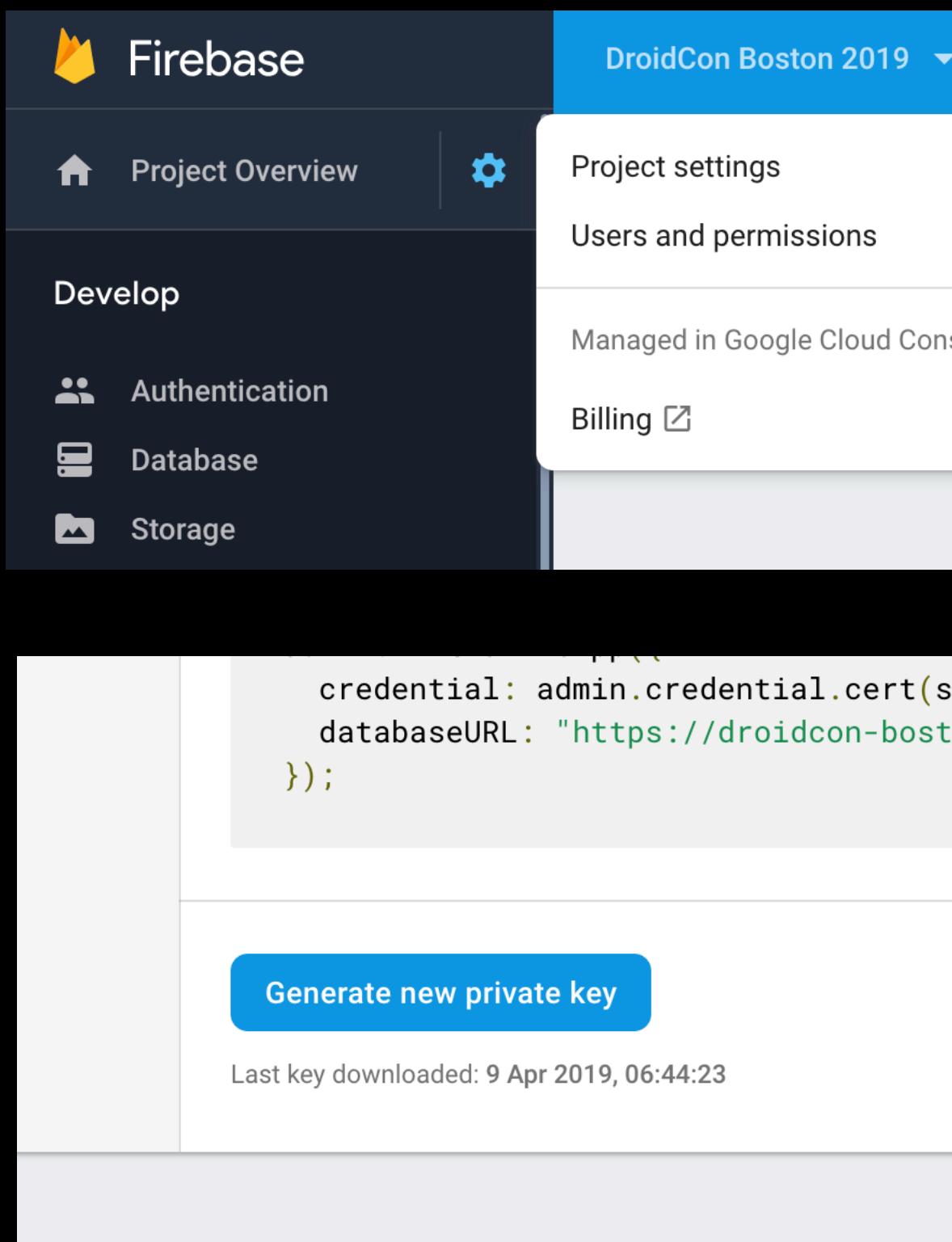
Predictions

Only top 4 labels are shown.

Andy	<div style="width: 80%;"></div>	0.803
Julien	<div style="width: 13.5%;"></div>	0.135
Cameron	<div style="width: 3.1%;"></div>	0.031
Ruth	<div style="width: 3.1%;"></div>	0.031

Your model is trained and ready to use, let's give it a quick test through the web interface.

Classify Faces in Android App



```
private const val PROJECT = "droidcon-boston-2019"
private const val LOCATION = "us-central1"
private const val MODEL = "ICN163646692449999978"
private const val SERVICE_ACCOUNT_JSON = "${"\n" +
    "  \"type\": \"service_account\", \n" +
    "  \"project_id\": \"droidcon-boston-2019\", \n" +
    "  \"private_key_id\": \"7d0c363f95d2af4939999993f0ff9f3c615a36d\", \n" +
    "  \"private_key\": \"\", \n" +
    "  \"client_email\": \"firebase-adminsdk@droidcon-boston-2019.firebaseio.gserviceaccount.com\", \n" +
    "  \"client_id\": \"11419990381871398450\", \n" +
    "  \"auth_uri\": \"https://accounts.google.com/o/oauth2/auth\", \n" +
    "  \"token_uri\": \"https://oauth2.googleapis.com/token\", \n" +
    "  \"auth_provider_x509_cert_url\": \"https://www.googleapis.com/oauth2/v1/certs\", \n" +
    "  \"client_x509_cert_url\": \"https://www.googleapis.com/robot/v1/metadata/boston-2019.firebaseio.gserviceaccount.com\"}\n"}
```

REST or SDK



```

private fun classifyUsingRetrofit(faceId: Int, imageBytes: ByteArray) {
    launch(errorHandler) {
        // Show loading indicator while we wait for the request.
        mResult.value = LoadingResource(FaceClassification(faceId, "", 0.0))

        // Build the body of our request, essentially the image to be classified.
        val body = CloudAutoMLModel(
            Payload(
                MlImage(
                    String(
                        Base64.encodeBase64(imageBytes)
                    )
                )
            )
        )

        // Define the authentication credentials and make the API request
        val response = getRESTService().classify(
            "Bearer ${accessToken?.tokenValue}",
            PROJECT, LOCATION, MODEL, body
        ).await()

        if (response.payload?.isNotEmpty() == true) {
            // We have a prediction!
            var predictedName: String? = null
            var predictedConfidence: Double? = null

            response.payload.forEach { entry ->
                if (entry.displayName != null) {
                    predictedName = entry.displayName
                    predictedConfidence = entry.classification?.score
                }
            }

            if (predictedName != null && predictedConfidence != null) {
                // We had an actual name returned
                mResult.postValue(
                    SuccessResource(
                        FaceClassification(
                            faceId,
                            predictedName!!,
                            predictedConfidence!!
                        )
                    )
                )
            } else {
                // No name was returned, this is an unknown face.
                mResult.postValue(ErrorResource(null))
            }
        } else {
            // There were no payloads returned, possible error or unknown face.
            mResult.postValue(ErrorResource(null))
        }
    }
}

```

```

private fun classifyUsingCloudSDK(faceId: Int, imageBytes: ByteArray) {
    launch(errorHandler) {
        // Show loading indicator while we wait for the request.
        mResult.value = LoadingResource(FaceClassification(faceId, "", 0.0))

        withContext(Dispatchers.IO) {
            // Define the authentication credentials
            val settings = PredictionServiceSettings.newBuilder()
                .setCredentialsProvider(FixedCredentialsProvider.create(mServiceCredentials)).build()

            val predictionServiceClient = PredictionServiceClient.create(settings)
            predictionServiceClient.use { client ->
                // Build the body of our request, essentially the image to be classified.
                val name = ModelName.of(PROJECT, LOCATION, MODEL)
                val image = Image.newBuilder().setImageBytes(ByteString.copyFrom(imageBytes)).build()
                val payload = ExamplePayload.newBuilder().setImage(image).build()
                val params = HashMap<String, String>()

                // Make the API request.
                val response = client.predict(name, payload, params)

                if (response.payloadCount > 0) {
                    // We have a prediction!
                    var predictedName: String? = null
                    var predictedConfidence: Double? = null

                    response.getPayload(0).allFields.entries.forEach { entry ->
                        if (entry.key.jsonName == "displayName") {
                            predictedName = entry.value as String
                        } else if (entry.key.jsonName == "classification") {
                            val classification = entry.value as ClassificationProto.ClassificationAnnotation
                            predictedConfidence= classification.score.toDouble()
                        }
                    }

                    if (predictedName != null && predictedConfidence != null) {
                        // We had an actual name returned
                        mResult.postValue(
                            SuccessResource(
                                FaceClassification(
                                    faceId,
                                    predictedName!!,
                                    predictedConfidence!!
                                )
                            )
                        )
                    } else {
                        // No name was returned, this is an unknown face.
                        mResult.postValue(ErrorResource(null))
                    }
                } else {
                    // There were no payloads returned, possible error or unknown face.
                    mResult.postValue(ErrorResource(null))
                }
            }
        }
    }
}

```

07:55

95%

DroidCon Boston ML-Kit





Thank You

Andrew Kelly - Google Developer Expert for Android

@andrew_ke11y

<https://medium.com/devnibbles>