**Cognitive Computing Seminar Report: Google Cloud AI Image Recognition**

Nelson Shaw

Magdin Stoica

April 23, 2020

**Table of Contents**

[Cognitive Area Motivation 3](#_Toc38310799)

[Platform Motivation 3](#_Toc38310800)

[Problem Area 3](#_Toc38310801)

[Google’s Cloud AI Vision Overview 4](#_Toc38310802)

[Platform Capabilities 5](#_Toc38310803)

[Comparison with IBM’s Platform 6](#_Toc38310804)

[Indoor Localization with Image Recognition Tutorial 7](#_Toc38310805)

[References 17](#_Toc38310806)

Cognitive Area Motivation

The motivation behind selecting the cognitive area of image recognition is that recognizing images helps a cognitive computer to learn using a sense of vision. One of the key motivations for exploring image recognition is that it may be applied to a massive variety of fields including healthcare, archaeology, and engineering. Existing image recognition applications include recognizing a person, identifying brain tumors, and identifying a given breed of pets. This particular area is useful for identifying patterns in images in order to closely classify what the computer is observing. More specifically, these technologies will be helpful to identify the location of a user within an indoor space. Therefore, image recognition will be the subject of the seminar.

Platform Motivation

The motivation behind selecting the Google Cloud AI platform is that the Cloud AI platform's Vision AI service can provide image recognition services such as training on a large set of image data to accomplish the use-case specified in this seminar. The platform also works with Android and Google Firebase, which are more easily accessible than Apple's Core ML platform capabilities. The Vision AI will support the use-case of locating a user indoors as it allows for also building custom machine learning models which can be used to analyze very specific and custom images which are not a part of any pre-trained Cloud AI machine learning models. Additionally, the model can be deployed to a server for collecting test data and processing it in real time, which is ideal for locating a user indoors in real time. Finally, the platform can be integrated with other programming languages such as Python to connect to the server.

Problem Area

Essentially, the key problem in question is using image recognition technology to locate a user within an indoor environment. Typically, technologies like GPS are used to find a user outdoors. However, because of the complexity of the indoor environment, GPS is not as effective indoors. One such technique for finding a user indoors is called location fingerprinting. A given floor in a building can be setup with grid points around the floor. Fingerprinting consists of a training and test phase. In the training phase, a user stands on each grid point and collects signal data from wireless routers from that point. The signal data acts as the features of that grid point. This is done for every grid point so that a radio map is created. The test phase involves the user standing at any location (not just on a grid point) on the floor, collecting signal data. The machine learning algorithm will attempt to determine the closest grid point to the user based on signal data similarity.

For this tutorial, we will attempt to use image recognition instead of signal data in order to try to locate a user within an indoor environment. A cognitive solution can be used to solve this problem primarily because of the uncertainty of the data collected. From a computing perspective, there is uncertainty as to whether or not the user is standing at a particular location or not based on the signal data collected because of the complexity of the indoor environment affecting the signals. Consequently, there is uncertainty as to which point in the room the user is standing in, given the lighting and objects in the room and the distance between each grid point. Therefore, a cognitive solution can be developed to approach this problem.

Google’s Cloud AI Vision Overview

Google’s Cloud AI platform contains several cognitive services, including Vision AI, Video AI, multi-language processing, translation, speech-to-text, text-to-speech, Cloud Inference API, advertisement recommendations, and Cloud AutoML. Video AI allows for annotating video content with pre-trained models, as well as custom models. Multi-language processing allows for processing the context of a given section of text such as posts from social media. The translation service allows for custom real-time language translation models using AutoML. Finally, the Cloud Inference API detects correlations and makes inferences on time series data [1].

One of the main cognitive services provided by Cloud AI is the Vision AI services. Cloud AI's Vision allows for users to either use pre-trained machine learning models to assign labels to images and identify objects in an image or to create custom machine learning models using the AutoML features. The Vision service and AutoML are also capable of object identification. The additional features of Vision include being able to identify celebrities, face detection, handwritten text detection, and content moderation [2].

The AutoML platform will be necessary for solving the problem in this seminar. The AutoML platform allows users to create custom machine learning models on Google’s server that can either perform single or multi-label classification [2]. The platform outputs data such as precision and recall on the testing data and which images it labelled correctly [3]. It is also possible to use online prediction, which is not explored in the tutorial. Online prediction uploads the machine learning model to Google’s cloud server and allows users to request real-time predictions using unseen data after the model has been trained and tested [4]. For edge devices, there is AutoML Edge, which allows for exporting machine learning models onto edge devices such as mobile devices. This also means that users can export the model as a TensorFlow package so that it can be used in conjunction with Node.js [5]. The AutoML machine learning models may also be called using REST APIs and RPC APIs [6]. The image data can also be encoded in Base64 if the user does not want to specify the URI path of the image [7]. The AutoML Vision platform can also provide its services while using other programming languages. Essentially, Google’s AutoML Vision platform also provides cloud client libraries for languages such as C#, Go, Java, Node.js, Python, PHP, and Ruby [4]. The user can also deploy AutoML Edge to Android and IOS devices as well [6].

In terms of the available resources, a 1-year trial of Google Vision AI is available with over $400 in credit for the trial. The trial gives free access to creating custom machine learning models. The trial allows users to create their own dataset which they would like to test their machine learning model on. They will also have access to Google’s Cloud Storage, which allows them to store the training and testing images for the model and the CSV files that can be used to organize them. The trial allows for 40 node hours to be used in order to train the model with the images.

Google’s AutoML contains documentation on creating a custom model including documentation on importing images, training, and evaluating models. Demo videos are also available for demonstrating AutoML. Finally, Qwicklab labs are also available for practicing using AutoML under a time limit [2].

Below is a list of links used in the research.

* Preparing the Training Data: <https://cloud.google.com/vision/automl/docs/prepare>
* Creating Datasets and Importing Images: <https://cloud.google.com/vision/automl/docs/create-datasets>
* Training the Model: <https://cloud.google.com/vision/automl/docs/train>
* Evaluating the Model: <https://cloud.google.com/vision/automl/docs/evaluate>

Platform Capabilities

One of the key successes of the platform was the ability to work with both single label classification and multiple label classification. The platform can provide exact information as to which images were labelled correctly and which were not. The platform also allowed for specifying which images can be used for training, validation, and testing in a CSV file. Using enough node hours, the platform can achieve high precision and recall with varying image data. The best practices used for this research are mentioned in the walkthrough.

One of the major limitations is that aside from choosing whether the model is a single label or multi-label classification problem, the user does not have much control over anything else about the model other than the confidence level at the evaluation stage and which images are used for training and testing. This is when the user is using a trial account. For example, the user cannot specify which classification model can be used (e.g., KNN or Random Forest). Even though this is an image recognition service and that the way the service works may be different from traditional machine learning, the user may want to see how well different classification algorithms perform on the dataset.

It appears as though the user does not have the ability to specify any particular hyperparameters, like the number of neighbors in KNN. The user will need to gather a certain number of images in order to achieve high precision, or recall. Considering that the problem domain involves finding the position of a user indoors, this requires over 50 images per label. Otherwise, the model will achieve low precision (e.g., using only 20 images for training and 10 for testing results in roughly 50% precision). A key limitation and disadvantage to using the Google AutoML service to locate a user indoors is the time it takes to train the model. With 16 node hours, it can take up to 3 hours to train the model. Finally, the accuracy metric is not displayed for the model, which is a critical metric for machine learning and indoor localization.

Comparison with IBM’s Platform

One of the major benefits to using IBM’s visual recognition service over AutoML is that training and evaluating the test results takes less time. The evaluation of the test images is very quick. The training time for IBM’s visual recognition service is significantly faster than the AutoML service. For the field of indoor localization, the time to evaluate the test results is significant considering that the user’s location needs to be updated as soon as possible. For IBM’s visual recognition service, it only takes the model roughly half an hour to train.

A major disadvantage for IBM’s visual recognition service is that the precision and recall metrics are not present, only the confidence level of the class label. This means that we will need to manually calculate confusion matrices based on the number of correct classifications. On the other hand, AutoML has the benefit of providing the confusion matrix automatically as well as which images were predicted correctly. Additionally, the IBM visual recognition service is limited to 250 MB of training data while AutoML can take more images than that. An advantage of AutoML over IBM’s visual recognition services is that the training and test data can be organized with a CSV file.

In terms of which platform classified the images better, the Google CloudAI platform actually achieved over 90% precision and recall. This means that it classified a lot of the test images correctly. However, this came at the cost of processing time, which is not ideal for locating a user indoors in real time. The IBM visual recognition service was not able to correctly classify the images of class “G2” but was able to classify the other three labels relatively well. However, this training was performed very quickly, which makes it suitable to locate a user within an indoor environment in real time despite the inaccuracies.

Indoor Localization with Image Recognition Tutorial

In this tutorial, we will learn how to develop a machine learning model with Google’s CloudAI in order to locate a user within a grid space indoors. This tutorial assumes that you have learned the following:

* Basic machine learning concepts including training, testing, and validation.
* Single label and multilabel classification.
* Introductory knowledge to image recognition.

So far, we have seen image recognition used in a variety of areas such as identifying a type of food, being able to detect a crime scene, facial recognition, or recognizing handwriting. However, there is one such field where image recognition may become helpful: the field of indoor localization. Indoor localization is a field that simply involves finding the location of a user within an indoor environment. Historically, researchers have used data from wireless signals emanating from built-in routers to track the user. Technologies such as GPS are used to locate a user outdoors, but because of the complexity of the indoor environment, GPS is not as effective indoors. One such technique for finding a user is known as position fingerprinting. Position fingerprinting involves planning a grid space within the indoor floor. The two stages of fingerprinting are the training and testing stages. In the training stage, a user may stand on each grid point with a mobile phone, and the phone will be able to receive information about the wireless signal strength values while at the grid point. Thus, the signal information for each grid point is stored in a database known as a radio map. In machine learning terms, the target variable for each grid point in the space is its number (e.g., grid point 1, 2, 3, etc.). The features of each grid point include the signal information received by the mobile phone while resting at the grid point. During the testing phase, a user may stand or move around indoors in real time, and the machine learning algorithm will try to find the closest grid point to the user with the radio map constructed previously.

We will implement an indoor localization system using image recognition. Instead of wireless signal data for each grid point, we will need to use images of a person standing at a grid point for the features of our model. We will implement this system using Google’s Cloud AI, which will allow us to perform single label and multilabel classification with our images.

Create a Google Cloud AI Account

To get started, use this link to create a Google Cloud AI Account if one is not created: <https://cloud.google.com/>. Terms and conditions will need to be accepted.

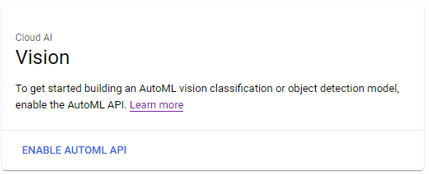
**Please note that it will require a credit card number. Please don’t panic though. It is only needed for confirming whether the user is a bot or not. We will not be spending any money for this tutorial.**

Once a Google Cloud AI account is created, we will use this link to go to the AutoML page: <https://cloud.google.com/automl>.

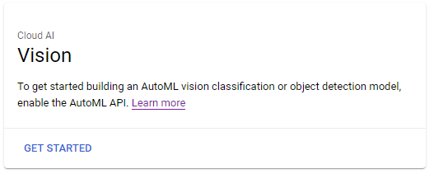
We need to select the “Try AutoML” dropdown button on the page and select the “Try AutoML Vision” button to begin creating custom machine learning models for image recognition.



We are required to enable the AutoML API for Vision after selecting the option in the main page.



We can then select the “Get Started” button.

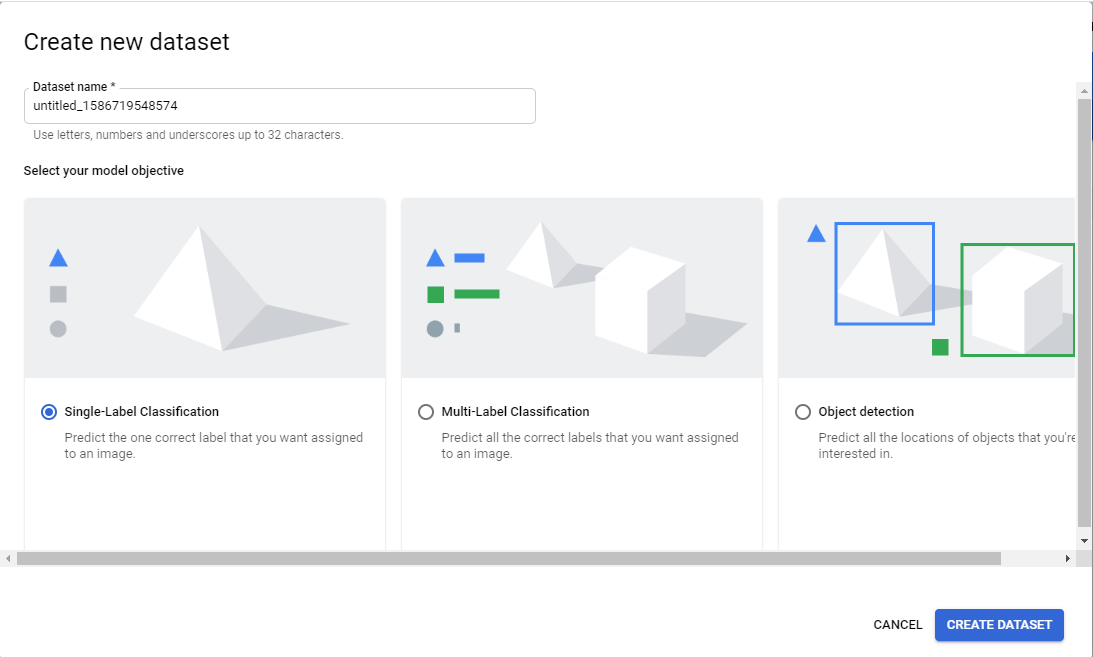


Create a New Dataset for Single Label Classification

Now that we have created our AutoML account, we will be taken to the home page for AutoML vision. In order to create our first custom machine learning model, we will need to create a new dataset by selecting “New Dataset” on the home page.



We have the option to either create a single label classification, multi-label classification, or an object detection model. We can name our model “indoor\_localization\_dataset” or any other name that is appropriate. For this tutorial, we will only be using single label classification. We need to select “Create Dataset” to create our single label classification model.



Collect the Training, Validation, and Test Images

The model we have created will require training, testing, and validation images. We can use the images provided by the zip file in this tutorial if we want to quickly get this step done. These sets of images provide a miniature-scale grid space with various obstacles in the indoor environment, different lighting, and different miniature people for variation in the dataset.

Otherwise, we have the option to create our own images. For the field of indoor localization, we ideally want to set up a grid space within the indoor environment that we want to test our model on. Otherwise, we can also try to create a small-scale model of our indoor space and grid. We need to create a grid in order provide the model with training images of the user or object standing at each grid point. This is so that when we test our model, the model can find the nearest grid point when the model is analyzing pictures of the user standing at random locations within the indoor grid space.

For creating images for the field of indoor localization for locating only one person within the room, we want to follow these best practices to prevent overfitting:

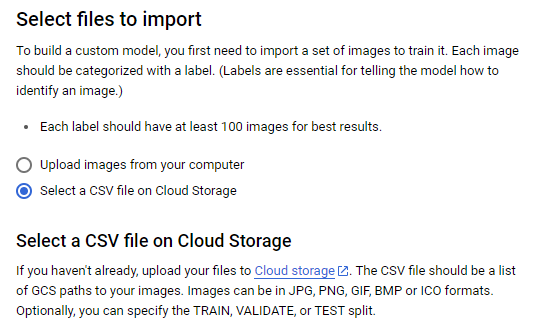
* Make sure that the images are consistent with the use-case the model is used in. For indoor localization, practically, we use images that are from the perspective of a security camera. However, if we want to consider that there are multiple security cameras in the room instead of just one, we could take multiple images of the user standing on a grid point at different corners of the room at different angles.
* Try to take pictures of the indoor environment at different levels of lighting.
* If possible, we want to try to move objects around in the indoor grid space and change the orientation of objects in the room. These include chairs, tables, computers, and many more. This is so that we won’t need to retrain the model as much when there are minor changes in the room.
* When standing at each grid point, try to change the orientation of the user. When we take pictures of the user standing at the top left grid point in the room for example, try to make them face left, right, forwards, and backwards from the camera.
* Ideally, when taking images of a person standing on or near a grid point, we want to have different people standing near that point in each image. This is so that the model can learn to recognize the position of the user and not the specific features of the person themselves. If we can’t find anyone else to stand on the grid point, try to have the user wear different clothes. If we choose to use a small-scale grid point space model instead, we can try to use different colors for the object that we are trying to locate in the model space. Try to use different types of objects. For example, we can try to take pictures of a red block sitting at the top left grid point for one image, and then use a yellow pyramid at the same grid point.

Depending on the problem that we are trying to solve, we may need different best practices depending on the use-case that we are considering.

For using the model, we need to have at least 10 images per label. The labels of course, will represent individual grid points in the space. We should have an 8:1:1 ratio of training, validation, and test images respectively for each label. If we use less images, the model may not be as precise and may have lower recall.

Create a Google Cloud Bucket for Storing the Images

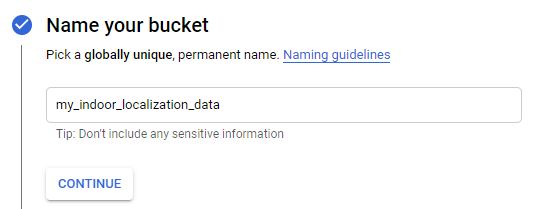
In order to use the images for our model, we can create a new Google Cloud Bucket. This can be done on the page we are currently on by selecting the “Cloud storage” link. The advantage of this is that we can organize which images we want to use as our training and test images.



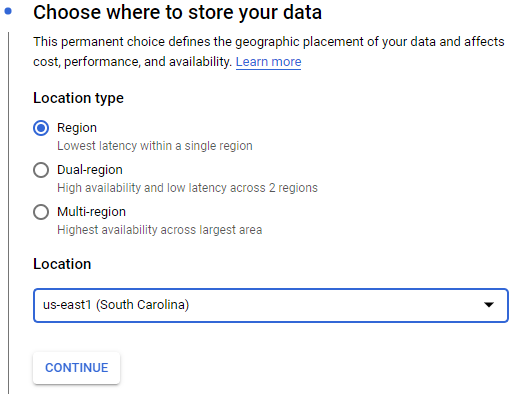
Afterwards, we should be able to create a new bucket for storing objects. **We must select a standard bucket and we must make sure that the region is identical to the one we have selected for our model.**



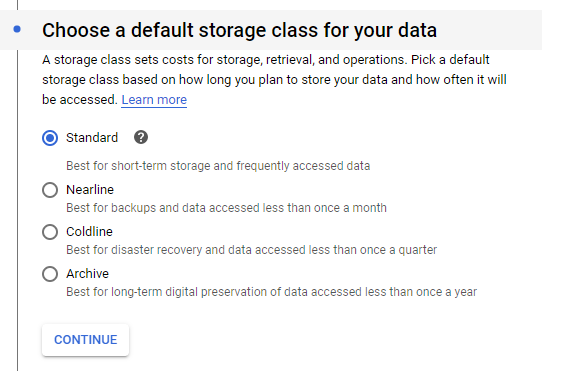
We will need to give our bucket a globally unique name. **Therefore, we will need to reformat our CSV file to use the name of our bucket. There is an Excel file attached to this walkthrough that will allow us to modify the name of our bucket if necessary. Once we have the name of our bucket in the Excel file, please copy all of the row values for the first three columns into a separate CSV file to use later.**



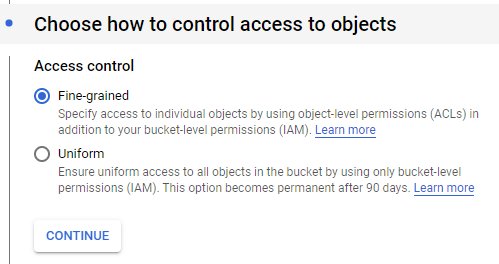
Next, select “Region” as the location type. **We need to select the same region as our project.** In our region, we would select “us-central1 (Iowa)” for example.



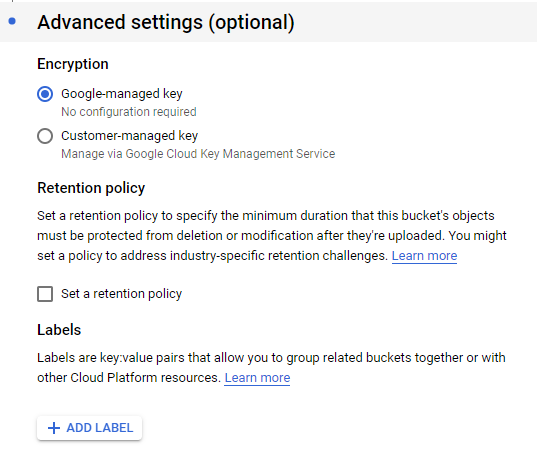
Select the “Standard” storage class for the data.



Select “Fine-grained” access control for the data.



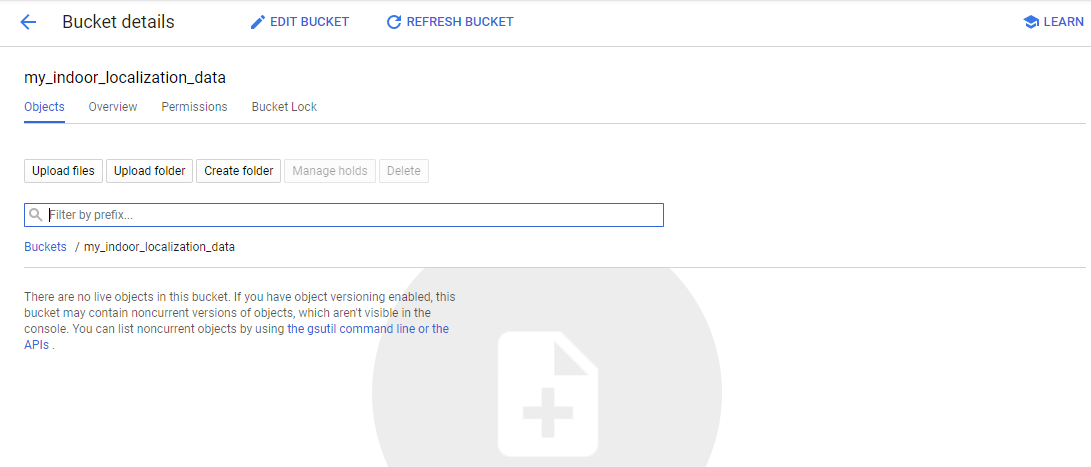
Next, ignore the advanced options as we will not be using them in this tutorial.



Finally, select “Create” to create the bucket.



We should now be able to add the images to our bucket. We would select “Upload folder” if we want to upload an entire folder of images. If we are using the zip file provided in this walkthrough, we can upload the individual folders for each class of images. **Make sure to upload the folder of images named “G1”, “G2”, “G3”, and “G4” and not the folder holding all four of those image folders.** The zip folder also includes the Excel file which we have used to create our CSV file which we will upload later.

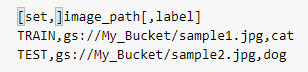


We should now be able to access the images on cloud storage. **If we are using custom images instead of the ones in the zip file, it is best practice to make sure to name the images based on their label and remember the names of the image files for adding the names into the CSV file.**

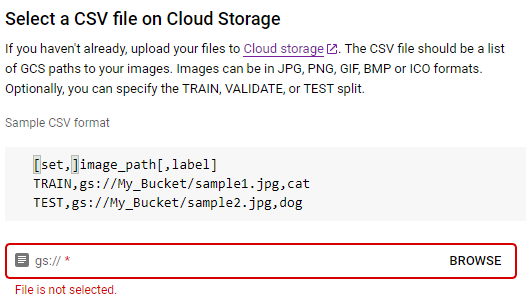
Create a CSV File to Organize Images

To make the process of organizing our images into training, testing, and validation sets easier, we can create a CSV file to organize our images and assign labels to them. If we already uploaded the zip folder contents to the cloud bucket, we should double check to make sure that the file names and the bucket name in the CSV file we made match up with the file names in our bucket. As mentioned before, we may need to change the names in the CSV file if they are different.

Otherwise, if we are using any other images than the ones in the zip file, we must make sure that our CSV file is in the following format:



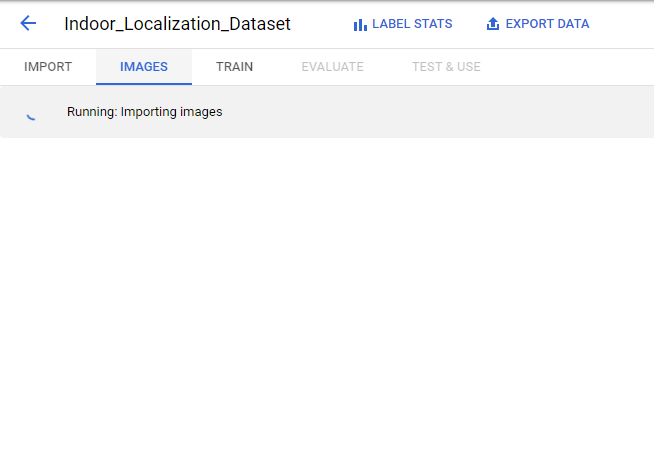
In the first column of the CSV file, we can assign the “TRAIN”, “TEST”, or “VALIDATE” values if we want to set specific images for use in training, testing, or validation. We will then need to select the CSV file that is stored in our bucket by selecting “Browse”.



Once this is done, select “Continue” to move to the next step.



Click on the “Images” tab to view the images imported and their labels. **It will take some time for the images to be imported.** If this operation is successful, we will have our images ready for training and testing.



Training and Testing the Model

To train the model, simply click the “Train” tab at the top and select “Start Training”.



Next, we need to give the model a name and make it cloud hosted. We should select at least 16 node hours to train for our model. We could train for 8 node hours instead, but it may lead to less precise test results. **For the trial account, we will have a maximum of 40 node hours available to us so we must make sure to be careful.** Select “Start Training” to train the model. This step will take time depending on how many node hours were selected.

Evaluation

Once the model has been trained and tested, we can observe the precision and recall of the model. Usually with fewer node hours, the precision and recall may be slightly lower. We can also observe the confusion matrix for our model to identify the true positives, true negatives, false positives, and false negatives of the prediction. Additionally, we will be able to see which test images were correctly labelled and which ones were not.

Conclusion

We have now created an image recognition model with Google’s AutoML to recognize the location of a user indoors. We can apply the image recognition technology to various other fields like healthcare, but we need to make sure to follow the best practices for image recognition.

References

[1] “Cloud AI  |  Google Cloud,” *Google*. [Online]. Available: https://cloud.google.com/products/ai. [Accessed: 20-Apr-2020].

[2] “Vision AI | Derive Image Insights via ML  |  Cloud Vision API,” *Google*. [Online]. Available: https://cloud.google.com/vision. [Accessed: 20-Apr-2020].

[3] “Evaluating models  |  Cloud AutoML Vision  |  Google Cloud,” *Google*. [Online]. Available: https://cloud.google.com/vision/automl/docs/evaluate. [Accessed: 20-Apr-2020].

[4] “Deploying your model  |  Cloud AutoML Vision  |  Google Cloud,” *Google*. [Online]. Available: https://cloud.google.com/vision/automl/docs/deploy. [Accessed: 20-Apr-2020].

[5] “Exporting Edge models  |  Cloud AutoML Vision  |  Google Cloud,” *Google*. [Online]. Available: https://cloud.google.com/vision/automl/docs/export-edge. [Accessed: 20-Apr-2020].

[6] “AutoML Vision documentation  |  Cloud AutoML Vision  |  Google Cloud,” *Google*. [Online]. Available: https://cloud.google.com/vision/automl/docs. [Accessed: 20-Apr-2020].

[7] “Base64 Encoding  |  Cloud AutoML Vision  |  Google Cloud,” *Google*. [Online]. Available: https://cloud.google.com/vision/automl/docs/base64. [Accessed: 20-Apr-2020].