Cigniti

Object Detection on Major Cloud Platforms

A Tech Brief

World's Leading AI & IP-led Digital Assurance & Digital Engineering Services Company

Introduction

APIs are becoming smarter in tandem with AI, and they can now perform extremely difficult and resource-intensive tasks. Object detection is one of them. A plethora of APIs and libraries promise to detect objects with incredible precision from video and images by combining pre-trained data sets with little or no manual intervention.

In this tech brief, we test several pre-trained Object Detection APIs.

We test these solutions for Food Detection cases. The aim is to understand the flexibility of training and reusing the models provided by all the three major cloud providers i.e GCP, AZURE, and AWS.In this tech brief, we will try to detail the complete pipeline from annotating the images to the model performance and usage of Endpoint API.

The Dataset we chose is from a Food store and the use case is to detect the ingredients kept in the boxes for the preparation of the sandwiches.

- Onions
- Banana pepper
- Tomatoes
- Pickles
- Guacamole
- Romaine lettuce
- Food (Which does not belong to any other classes)

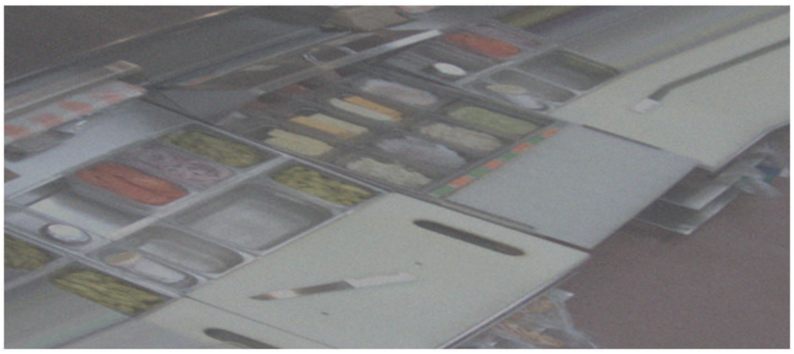
Pretrained APIs from the cloud providers have already models built-in and in general, they are a black box, so it is impossible to know the architecture of the models, but we will attempt to see the better usage of these pre-trained models provided by Major cloud providers.

Data Preparation

In the Food Detection problem, we have 8 classes in the data, namely onions, banana pepper, tomatoes, food, pickles, guacamole, romaine lettuce, and cheese.

Before going to detect images. We have to annotate the images. For getting better results we have added noise (Gaussian Noise) and rotated (few images 0-45 degree) images.

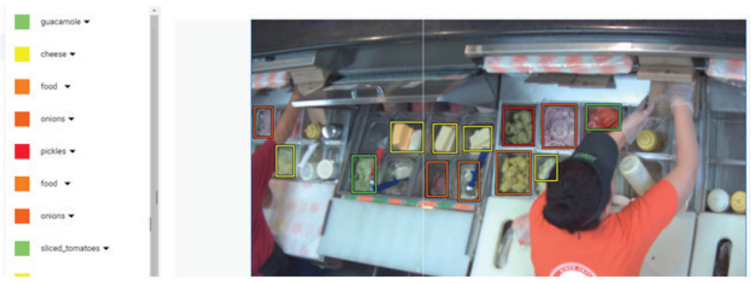
Example of a rotated image



Example of a Noise Image

A. GCP

In GCP we use Google Vision API to detect images



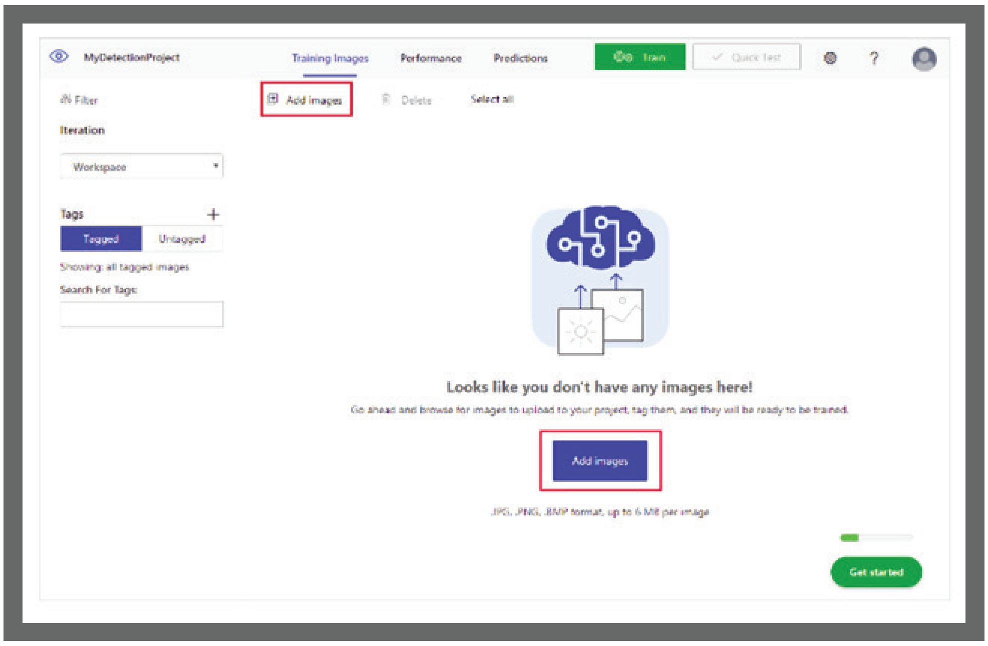
You need to do this for at least 10 images, but Google recommends that you do it for 100 images to have better accuracy.

View Label Statistics:

- Select the non-empty dataset from the Datasets page.
- Select the Label Stats option at the top of the Dataset details page. This opens a window on the right side of the screen where you can view statistics for your labeled bounding boxes in images.
- Select Done to close the label statistics window.

B. Azure

In Azure, we use Custom Vision API (Azure cognitive services) for object detection. After creating a project, Click add images to add images and upload.

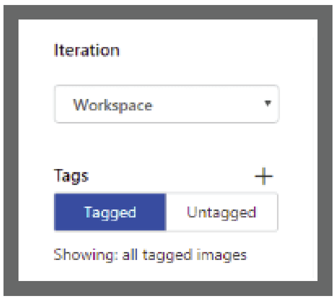


Looks like you don't have any images here!

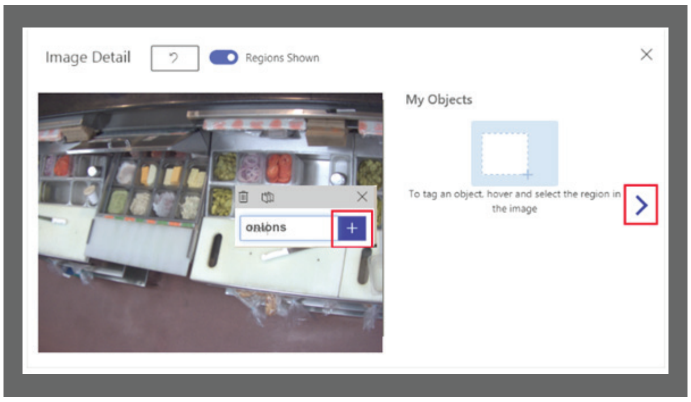
Go ahead and browse for images to upload to your project, tag them, and they will be ready to be trained.

.JPG, .PNG, .BMP format, up to 6 MB per image

Uploaded images can be found in the Untagged section



Uploaded images can be found in the Untagged section

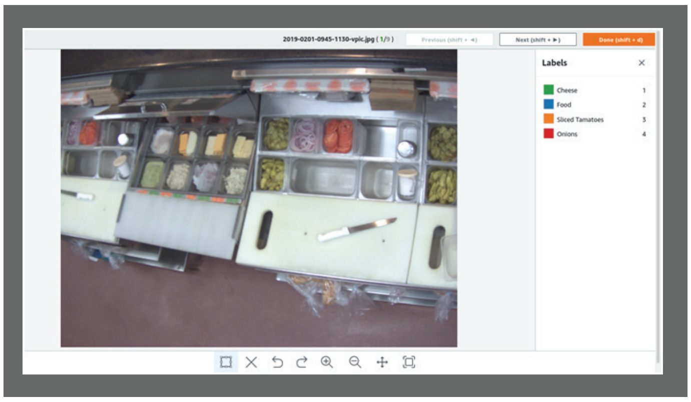


To tag an object, hover and select the region in the image

C. AWS

In AWS we use Custom Object Detection for object detection.

1. Create a Project and then add a dataset to it.
2. Once you have the dataset add the required labels.
3. Now select the images this in turn highlights the Assign labels and Draw bounding box.
4. Click on the Draw **bounding box** to label images. (Make sure you Save the changes by Clicking **Save changes**)



🟩	Cheese	1
🟦	Food	2
🟧	Sliced Tamatoes	3
🟥	Onions	4

Training

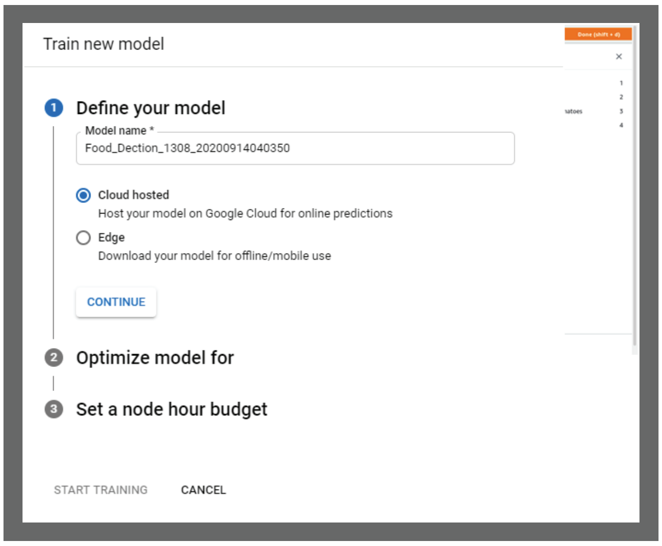
A. GCP

When you have a dataset with a solid set of annotated training images with bounding boxes and labels, you are ready to create and train the custom model.

AutoML API uses the images from the dataset to train the model, test it, and evaluate its performance. You review the results, adjust the training dataset as needed, and train a new model using the improved dataset.

- Select the “Train” tab and “Train new model” button.
- From the Define your model section, specify a Model name, (or accept the default name)
- There are two options available in this section namely,

  1. Cloud Hosted
  2. Edge



Train new model

1 Define your model

Model name *
Food_Dection_1308_20200914040350

[✓ Cloud hosted]
Host your model on Google Cloud for online predictions

Edge
Download your model for offline/mobile use

CONTINUE

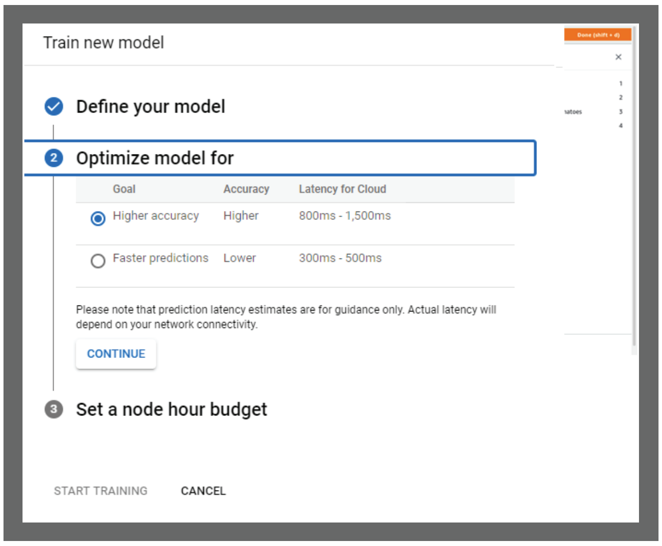
2 Optimize model for

3 Set a node hour budget

START TRAINING CANCEL

1. Training Cloud Hosted Models

- Select Cloud-hosted radio_button as the model type if not already specified. After selecting to train a Cloud-hosted. select Continue.
- In the following Optimize model for section, select your desired optimization criterion: Higher accuracy or Faster prediction.
- Faster prediction means a faster model and vice versa. After selecting the optimization specification select Continue.
- In the following "Set a node hour budget" section specifies your desired node budget. By default, 24 node hours should be sufficient for most datasets to train your model. This recommended value is an estimation to make the model fully converged. However, you can choose another amount. The minimum amount of node hours for Object Detection is 20. For Image Classification, this minimum amount is 8.
- In this section, you can also opt into auto-deployment of your model after training by selecting "Deploy model to 1 node after training"


Train new model

Define your model

2 Optimize model for

Goal	Accuracy	Latency for Cloud
[✓ Higher accuracy]	Higher	800ms - 1,500ms
Faster predictions	Lower	300ms - 500ms

Please note that prediction latency estimates are for guidance only. Actual latency will depend on your network connectivity.

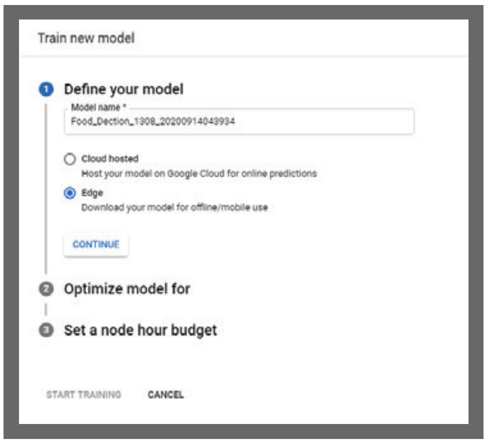
CONTINUE

3 Set a node hour budget

START TRAINING CANCEL

2. Training Edge Exported Models

- Select Edge radio_button as the model type if not already specified. After selecting to train an Edge model select Continue.
- In the following Optimize model for section, select your desired optimization criterion
- In the following "Set a node hour budget" section use the recommended node hour budget or specify a different value
- Select Start training to begin model training


Train new model

1 Define your model

Model name *
Food_Dection_1308_20200914043934

Cloud hosted [✓ Edge]

Cloud hosted: Host your model on Google Cloud for online predictions

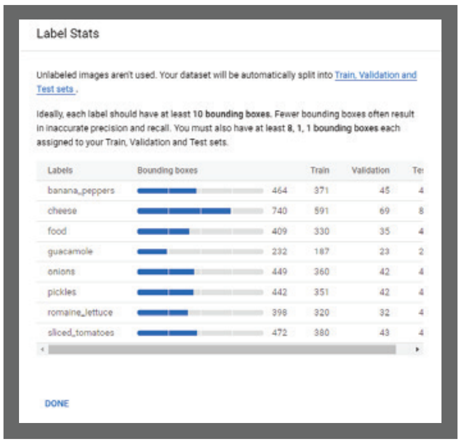
Edge: Download your model for offline/mobile use

CONTINUE

2 Optimize model for

3 Set a node hour budget

START TRAINING CANCEL


Label Stats

Unlabeled images aren't used. Your dataset will be automatically split into Train, Validation and Test sets .

Ideally, each label should have at least **10 bounding boxes**. Fewer bounding boxes often result in inaccurate precision and recall. You must also have at least **8, 1, 1 bounding boxes each** assigned to your Train, Validation and Test sets.

Labels	Bounding boxes	Train	Validation	Te
banana_peppers	464	371	45	4
cheese	740	591	69	8
food	409	330	35	4
guacamole	232	187	23	2
onions	449	360	42	4
pickles	442	351	42	4
romaine_lettuce	398	320	32	4
sliced_tomatoes	472	380	43	4

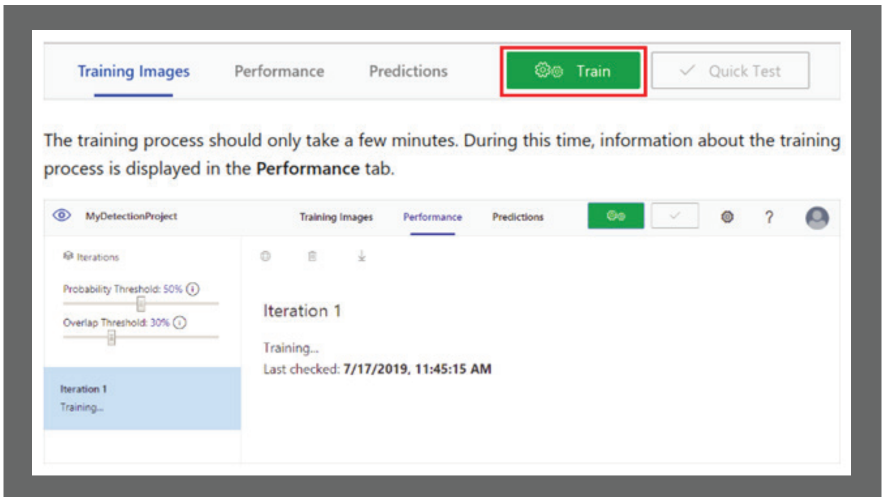
DONE

B. Azure

To train the detector model, select the Train button. The detector uses all of the current images and their tags to create a model that identifies each tagged object.

There are two options to Train

- Quick train – Takes less time to train and we get less accuracy
- Advance train – Takes more time to train we get more accuracy



The training process should only take a few minutes. During this time, information about the training process is displayed in the **Performance** tab.

Probability Threshold: 50%
Overlap Threshold: 30%

Iteration 1
Training...

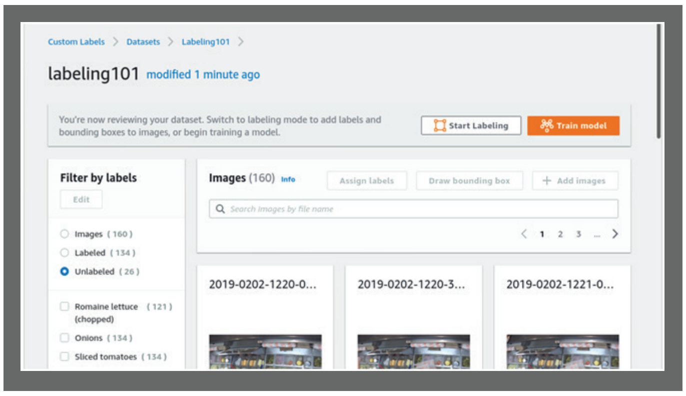
Iteration 1
Training...
Last checked: **7/17/2019, 11:45:15 AM**

Select an option and click Train

C. AWS

The labeling tool lets you know whether you've enough images labeled to train the model on the top of the page. Be vigilant of it. Once you have enough labeled images, exit from the labeling tool. To exit Click on the Exit button.

1. You'll see the Train Model option available on the right side on the screen, click on it to start training.



labeling101 modified 1 minute ago

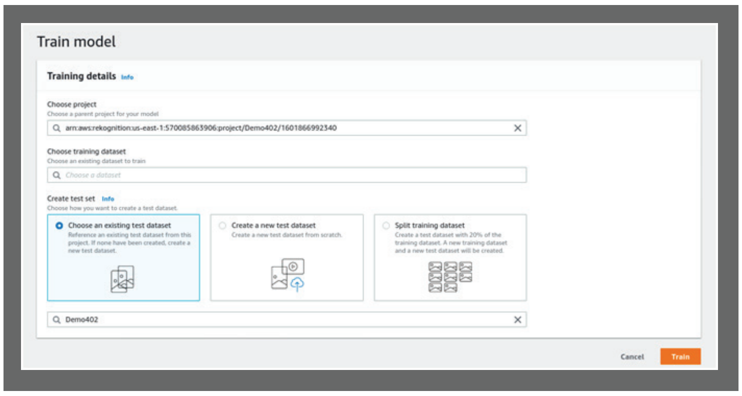
You're now reviewing your dataset. Switch to labeling mode to add labels and bounding boxes to images, or begin training a model.

Images (160)
Labeled (134)
Unlabeled (26)

Romaine lettuce (chopped) (121)
Onions (134)
Sliced tomatoes (134)

2019-0202-1220-0...
2019-0202-1220-3...
2019-0202-1221-0...

2. Choose the labeled and test dataset.


Train model

Training details Info

Choose project
Choose a parent project for your model
arn:aws:rekognition:us-east-1:570085863906:project/Demo402/1601866992340

Choose training dataset
Choose an existing dataset to train
Choose a dataset

Create test set Info
Choose how you want to create a test dataset.

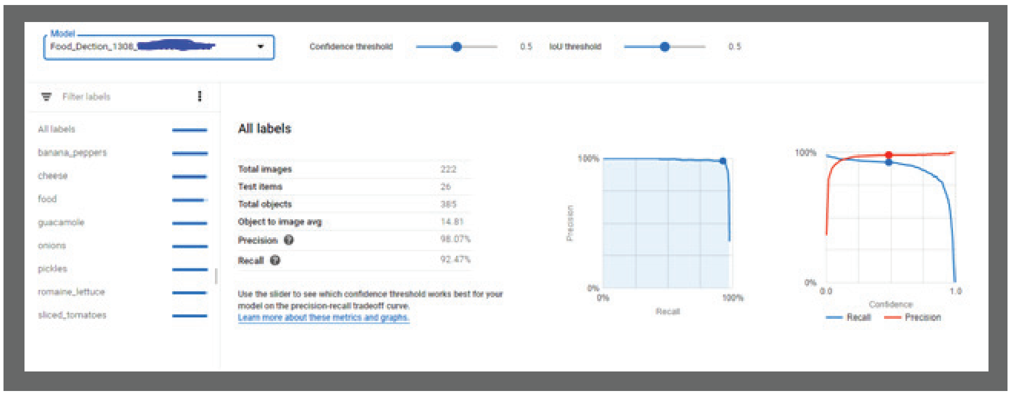
- [✓ Choose an existing test dataset] Reference an existing test dataset from this project. If none have been created, create a new test dataset.
- Create a new test dataset: Create a new test dataset from scratch.
- Split training dataset: Create a test dataset with 20% of the training dataset. A new training dataset and a new test dataset will be created.

Demo402

Models Evaluation

A. GCP

- Click the Models tab (with 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.
- Click the row for the model you want to evaluate.
- If necessary, click the Evaluate tab just below the title bar.


Model
Food_Dection_1308_[redacted]
Confidence threshold 0.5 IoU threshold 0.5

Filter labels

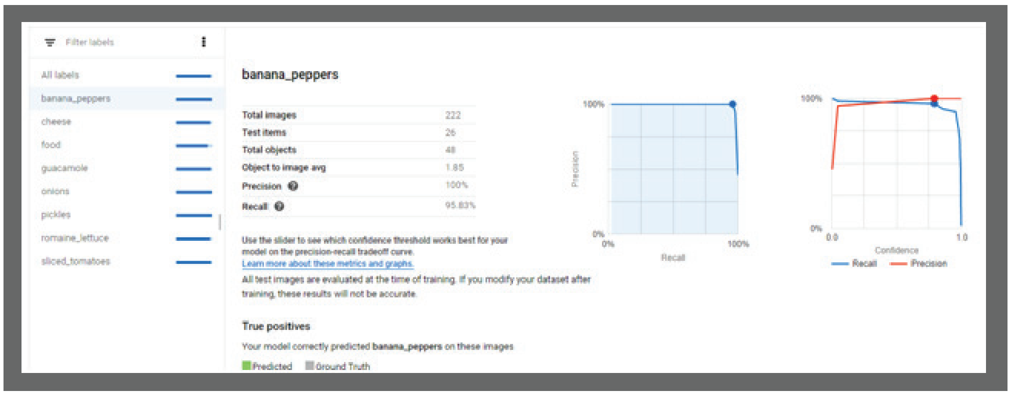
All labels
banana_peppers
cheese
food
guacamole
onions
pickles
romaine_lettuce
sliced_tomatoes

All labels

Total images	222
Test items	26
Total objects	385
Object to image avg	14.81
Precision	98.07%
Recall	92.47%

Use the slider to see which confidence threshold works best for your model on the precision-recall tradeoff curve.
Learn more about these metrics and graphs.

To view the metrics for a specific label, select the label name from the list of labels in the lower part of the page


Filter labels

All labels
banana_peppers
cheese
food
guacamole
onions
pickles
romaine_lettuce
sliced_tomatoes

banana_peppers

Total images	222
Test items	26
Total objects	48
Object to image avg	1.85
Precision	100%
Recall	95.83%

Use the slider to see which confidence threshold works best for your model on the precision-recall tradeoff curve.
Learn more about these metrics and graphs.
All test images are evaluated at the time of training. If you modify your dataset after training, these results will not be accurate.

True positives

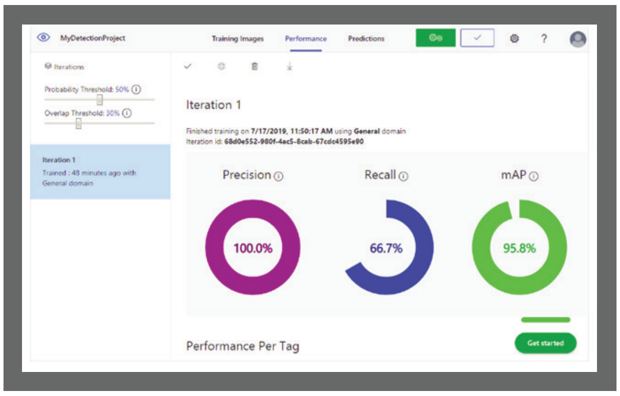
Your model correctly predicted **banana_peppers** on these images

🟩 Predicted ▨ Ground Truth

B. Azure

After training model performance is displayed with precision, recall, and mean average precision

- Precision indicates the fraction of identified classifications that were correct. For example, if the model identified 100 images as dogs, and 99 of them were actually of dogs, then the precision would be 99%.
- Recall indicates the fraction of actual classifications that were correctly identified. For example, if there were 100 images of apples, and the model identified 80 as apples, the recall would be 80%.


MyDetectionProject
Training Images Performance Predictions

Iterations
Probability Threshold: 50%
Overlap Threshold: 30%

Iteration 1
Trained : 48 minutes ago with General domain

Iteration 1

Finished training on **7/17/2019, 11:50:17 AM** using **General** domain
Iteration id: **68d0e552-980f-4ac5-8cab-67cdc4595e90**

Precision 100.0%
Recall 66.7%
mAP 95.8%

Performance Per Tag

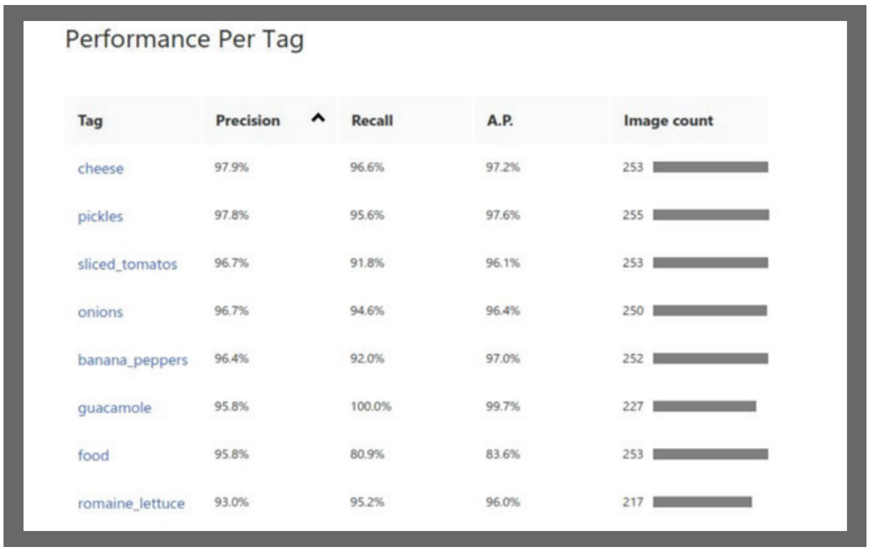
Probability Threshold

The probability Threshold slider is the level of confidence that a prediction needs to have to be considered correct

When you interpret prediction calls with a high probability threshold, they tend to return results with high precision at the expense of recall—the detected classifications are correct, but many remain undetected. A low probability threshold does the opposite—most of the actual classifications are detected, but there are more false positives within that set. With this in mind, you should set the probability threshold according to the specific needs of your project. Later, when you're receiving prediction results on the client-side, you should use the same probability threshold value as you used here.

Each time we train a detector we create a new Iteration

Final iteration

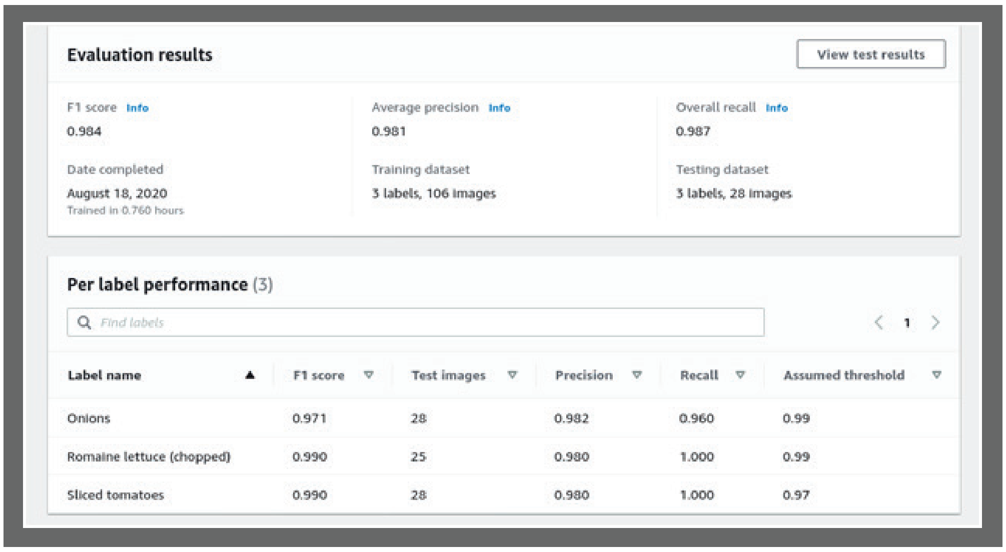


Performance Per Tag

Tag	Precision	Recall	A.P.	Image count
cheese	97.9%	96.6%	97.2%	253
pickles	97.8%	95.6%	97.6%	255
sliced_tomatos	96.7%	91.8%	96.1%	253
onions	96.7%	94.6%	96.4%	250
banana_peppers	96.4%	92.0%	97.0%	252
guacamole	95.8%	100.0%	99.7%	227
food	95.8%	80.9%	83.6%	253
romaine_lettuce	93.0%	95.2%	96.0%	217

C. AWS

After training model performance is displayed with Average precision, Overall recall, and F1 score. And you can see the test dataset with predicted labels as well.



Evaluation results

View test results

F1 score Info
0.984

Average precision Info
0.981

Overall recall Info
0.987

Date completed
August 18, 2020
Trained in 0.760 hours

Training dataset
3 labels, 106 images

Testing dataset
3 labels, 28 images

Per label performance (3)

Find labels

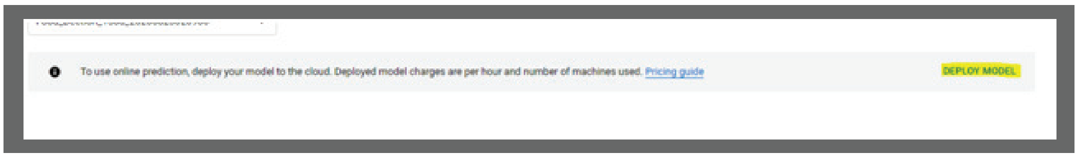
Label name	F1 score	Test images	Precision	Recall	Assumed threshold
Onions	0.971	28	0.982	0.960	0.99
Romaine lettuce (chopped)	0.990	25	0.980	1.000	0.99
Sliced tomatoes	0.990	28	0.980	1.000	0.97

Note: You have no control over IoU Threshold

Test & Deploy Models

A. GCP

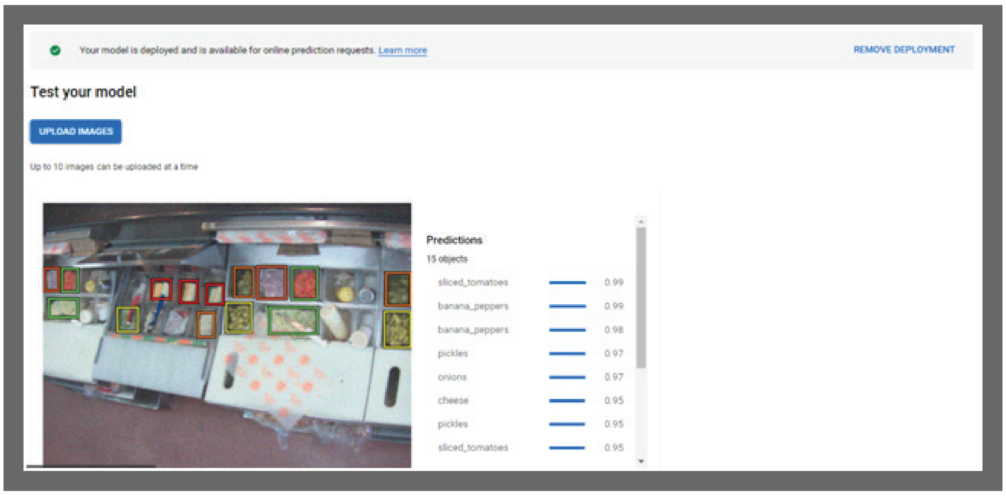
Once the model has been trained, you can proceed to the Test & Use tab and deploy the trained model. Click Upload Images to upload the images that you want to label.



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

Once the model has been deployed, you can upload your images to the Cloud Console and test the model's accuracy.

We have two options here "Individual Prediction" and "Batch Prediction".



Your model is deployed and is available for online prediction requests. Learn more

Test your model

Up to 10 images can be uploaded at a time

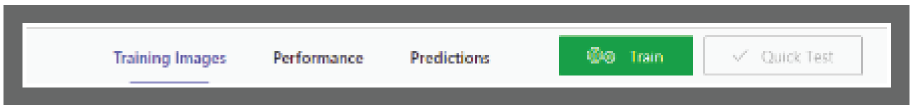
Predictions

15 objects

sliced_tomatoes	0.99
banana_peppers	0.99
banana_peppers	0.98
pickles	0.97
onions	0.97
cheese	0.95
pickles	0.95
sliced_tomatoes	0.95

B. Azure

Once the model is trained we can test the images using a quick test button



And upload a test image

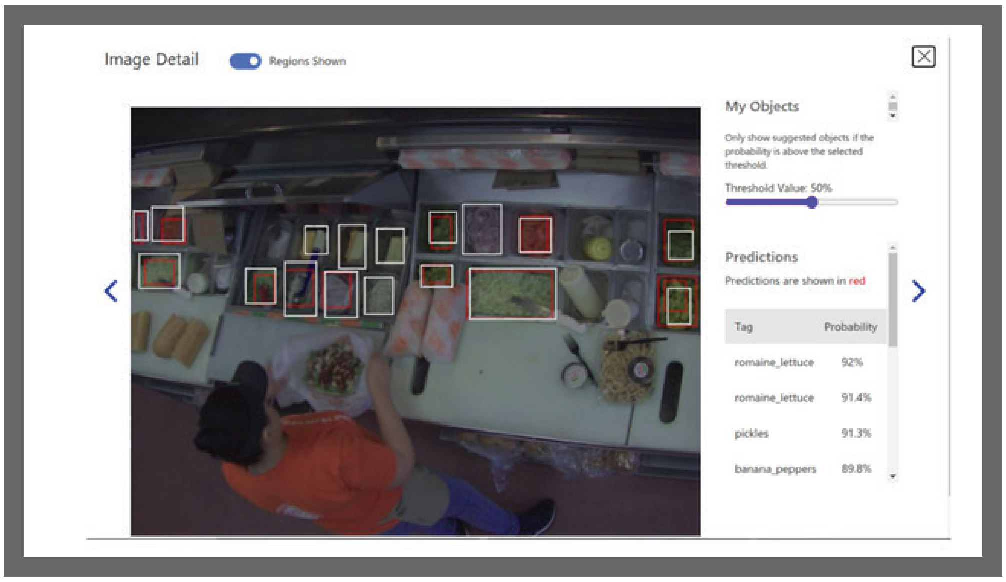


Image Detail

Regions Shown

My Objects

Only show suggested objects if the probability is above the selected threshold.

Threshold Value: 50%

Predictions

Predictions are shown in red

Tag	Probability
romaine_lettuce	92%
romaine_lettuce	91.4%
pickles	91.3%
banana_peppers	89.8%

C. AWS

Testing the model is quite different in AWS as it doesn't allow us to upload images to the model or to export the model to local in any format.

To test the model one should follow the guidelines given right under the performance results.

Starting or Stopping an Amazon Rekognition Custom Labels Model

The Amazon Rekognition Custom Labels console provides example code that you can use to start and stop a model.

To start or stop a model (console)

1. If you haven't already:
   a. Create or update an IAM user with AmazonRekognitionFullAccess permissions. For more information, see Step 2: Create an IAM Administrator User and Group.
   b. Install and configure the AWS CLI and the AWS SDKs. For more information, see Step 2: Set Up the AWS CLI and AWS SDKs.
2. Open the Amazon Rekognition console at https://console.aws.amazon.com/rekognition/

- Choose Use **Custom Labels.**
- Choose **Get started.**
- In the left navigation pane, choose **Projects.**
- On the **Projects** resources page, choose the project that contains the trained model that you want to start or stop.
- In the **Models** section, choose the model that you want to start or stop. The summary results are shown.
- In the **Use model section**, choose **API Code**.
- At the command prompt, use the AWS CLI command that calls start-project-version to start your model. Use the code snippet that calls stop-project-version to stop your model. The value of –project-version-arn should be the Amazon Resource Name (ARN) of your model.
- Choose your project name at the top of the page to go back to the project overview page.
- In the **Model** section, check the status of the model. When the status is **The model is running,** you can use the model to analyze images.

Analyzing an Image with a Trained Model

To analyze an image with a trained Amazon Rekognition Custom Labels model, you call the DetectCustomLabels API. The result from DetectCustomLabels is a prediction that the image contains specific objects, scenes, or concepts.

To call DetectCustomLabels, you specify the following:

- The Amazon Resource Name (ARN) of the Amazon Rekognition Custom Labels model that you want to use.
- The image that you want the model to make a prediction with. You can provide an input image as an image byte array (base64-encoded image bytes), or as an Amazon S3 object. For more information, see Image.

Custom labels are returned in an array of Custom Label objects. Each custom label represents a single object, scene, or concept found in the image. A custom label includes:

- A label for the object, scene, or concept found in the image.
- A bounding box for objects found in the image. The bounding box coordinates show where the object is located on the source image. The coordinate values are a ratio of the overall image size. For more information, see BoundingBox.
- The confidence that Amazon Rekognition Custom Labels has in the accuracy of the label and bounding box.

During training a model calculates a threshold value that determines if a prediction for a label is true. By default, DetectCustomLabels doesn't return labels whose confidence value is below the model's calculated threshold value. To filter labels that are returned, specify a value for MinConfidence that is higher than the model's calculated threshold. You can get the model's calculated threshold from the model's training results shown in the Amazon Rekognition Custom Labels console. To get all labels, regardless of confidence, specify a MinConfidence value of 0.

If you're finding the confidence values returned by DetectCustomLabels are too low, consider retraining the model. For more information, see Training an Amazon Rekognition Custom Labels Model. You can restrict the number of custom labels returned from DetectCustomLabels by specifying the MaxResults input parameter. The results are returned sorted from the highest confidence to the lowest.

For other examples that call DetectCustomLabels, see Examples.

For information about securing DetectCustomLabels, see Securing DetectCustomLabels.

To detect custom labels (API)

1. If you haven't already:
    1. Create or update an IAM user with AmazonRekognitionFullAccess and AmazonS3ReadOnlyAccess permissions. For more information, see Step 2: Create an IAM Administrator User and Group.
    2. Install and configure the AWS CLI and the AWS SDKs. For more information, see Step 2: Set Up the AWS CLI and AWS SDKs.
2. Train and deploy your model. For more information, see Getting Started with Amazon Rekognition Custom Labels.
3. Ensure the IAM user calling DetectCustomLabels has access to the model you used in step 3. For more information, see Securing DetectCustomLabels.
4. Upload an image that contains text to an S3 bucket.

For instructions, see Uploading Objects into Amazon S3 in the Amazon Simple Storage Service Console User Guide.

5. Use the following examples to call the DetectCustomLabels operation.

   Use the following example code to start a model.

CLI

Change the value of project-version-arn to the ARN of the model that you want to start.Change the value of –min-inference units to the number of inference units that you want to use.

aws rekognition start-project-version –project-version-arn model_arn\ –min-inference-units "1"

Python

The following example code displays bounding boxes around custom labels detected in an image. Replace the following values:

- bucket with the name of the Amazon S3 bucket that you used in step 4.
- image with the name of the input image file you uploaded in step 4.

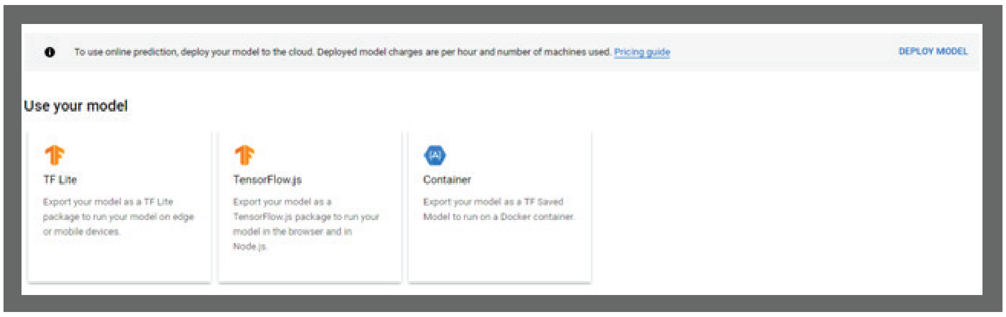
- min_confidence with the minimum confidence (0-100). Objects detected with a confidence lower than this value are not returned.
- model with the ARN of the model that you want to use with the input image.

For example code snippet look here: Python Code

Export Edge Model

A. GCP

After the model has been created on our custom data, We can export the model. We can export the model in either generic TensorFlow Lite, TensorFlow.js, or container format.



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

DEPLOY MODEL

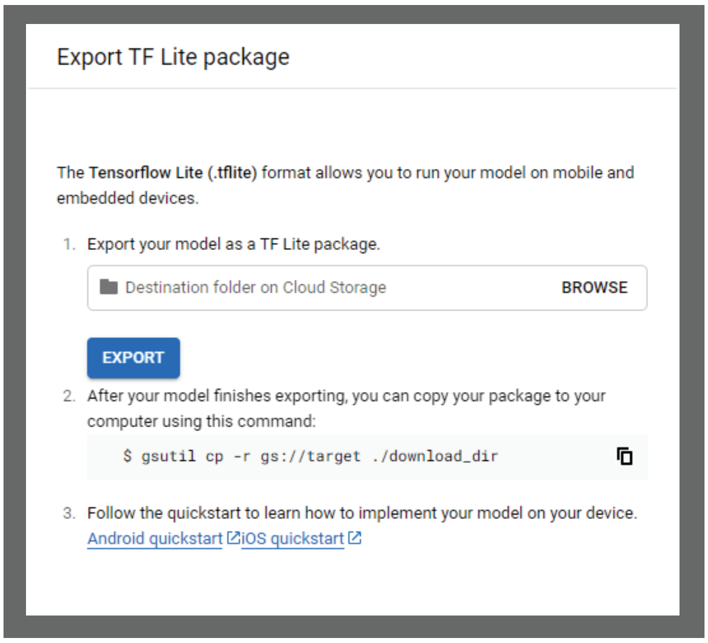
Use your model

TF Lite
Export your model as a TF Lite package to run your model on edge or mobile devices.

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

Container
Export your model as a TF Saved Model to run on a Docker container.

Once we select the TFLite option. After selecting the TF Lite option and specifying the export location on Cloud Storage in the side window, select Export to export your Edge TF Lite model.



Export TF Lite package

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

1. Export your model as a TF Lite package.

   Destination folder on Cloud Storage — BROWSE

   EXPORT

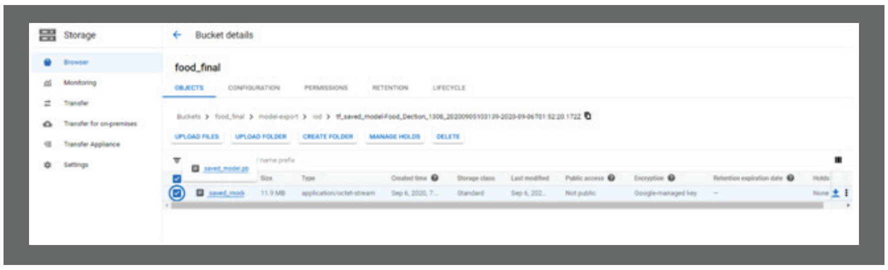
2. After your model finishes exporting, you can copy your package to your computer using this command:

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

3. Follow the quickstart to learn how to implement your model on your device. Android quickstart iOS quickstart

In the Use your model section, select the “Container option”. After selecting the Container option and specifying the export location on Cloud Storage in the side window, click Export to export your Edge model.

The folder contains a TensorFlow model named “saved_model.pb”



Storage — Browser, Monitoring, Transfer, Transfer for on-premises, Transfer Appliance, Settings

Bucket details

food_final

OBJECTS | CONFIGURATION | PERMISSIONS | RETENTION | LIFECYCLE

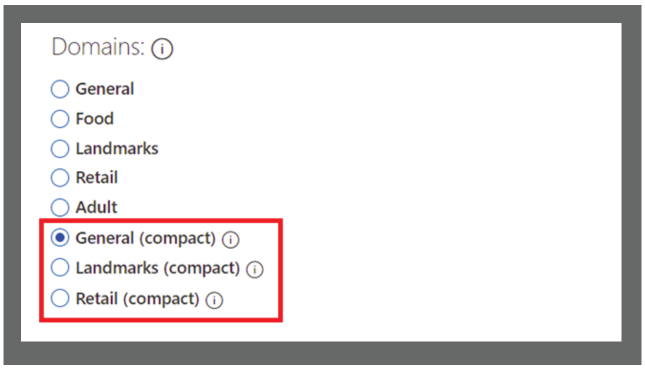
Buckets > food_final > model-export > iod > tf_saved_model-Food_Dection_1308_20200905103139-2020-09-06T01:52:20.172Z

UPLOAD FILES | UPLOAD FOLDER | CREATE FOLDER | MANAGE HOLDS | DELETE

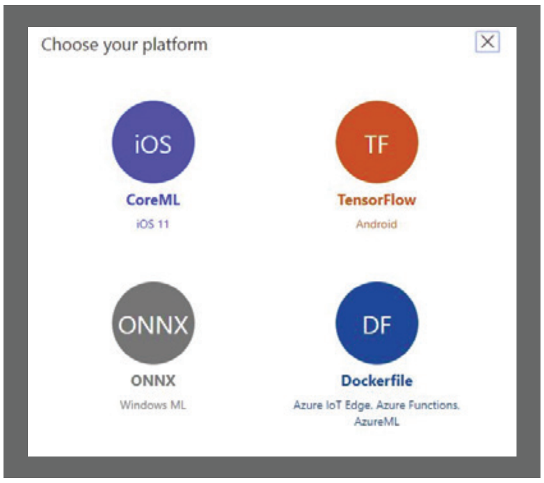
Name	Size	Type	Created time	Storage class	Last modified	Public access	Encryption	Retention expiration date	Holds
saved_model.pb	11.9 MB	application/octet-stream	Sep 6, 2020, 7...	Standard	Sep 6, 202...	Not public	Google-managed key	–	None

B. Azure

We can also export the model to run inference (Model should be trained on Compact domain)


Domains: ⓘ

- ○ General
- ○ Food
- ○ Landmarks
- ○ Retail
- ○ Adult
- ● General (compact) ⓘ
- ○ Landmarks (compact) ⓘ
- ○ Retail (compact) ⓘ



we can export the model to the following platforms

- TensorFlow
- CoreML
- ONNX
- Vision AI
- A Docker container for Windows, Linux, or ARM architecture. The container

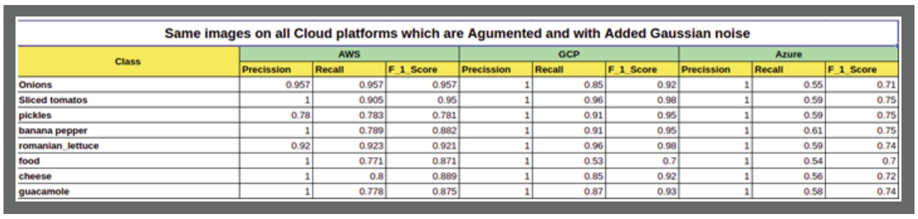
includes a TensorFlow model and service code to use the Custom Vision API.

C. AWS

AWS doesn't provide export options for these custom models.

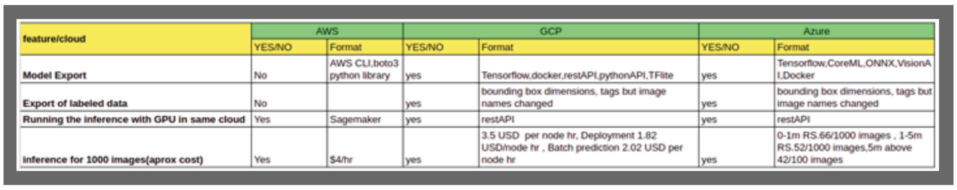
Conclusion

From the below table it is evident that GCP outperforms the remaining two platforms for this dataset, however, it is worth trying all three on your datasets before choosing any one. For our use case, GCP is a good starting point to go ahead with.


Same images on all Cloud platforms which are Agumented and with Added Gaussian noise

Class	AWS			GCP			Azure		
	Precission	Recall	F_1_Score	Precission	Recall	F_1_Score	Precission	Recall	F_1_Score
Onions	0.957	0.957	0.957	1	0.85	0.92	1	0.55	0.71
Sliced tomatos	1	0.905	0.95	1	0.96	0.98	1	0.59	0.75
pickles	0.78	0.783	0.781	1	0.91	0.95	1	0.59	0.75
banana pepper	1	0.789	0.882	1	0.91	0.95	1	0.61	0.75
romanian_lettuce	0.92	0.923	0.921	1	0.96	0.98	1	0.59	0.74
food	1	0.771	0.871	1	0.53	0.7	1	0.54	0.7
cheese	1	0.8	0.889	1	0.85	0.92	1	0.56	0.72
guacamole	1	0.778	0.875	1	0.87	0.93	1	0.58	0.74

The following table consolidates the features we want to see from the cloud provider, and here GCP and AZURE are having more export options.AWS we can use it as an endpoint, and Amazon Rekognition does not allow the model export



feature/cloud	AWS		GCP		Azure	
	YES/NO	Format	YES/NO	Format	YES/NO	Format
Model Export	No	AWS CLI,boto3 python library	yes	Tensorflow,docker,restAPI,pythonAPI,TFlite	yes	Tensorflow,CoreML,ONNX,VisionAI,Docker
Export of labeled data	No		yes	bounding box dimensions, tags but image names changed	yes	bounding box dimensions, tags but image names changed
Running the inference with GPU in same cloud	Yes	Sagemaker	yes	restAPI	yes	restAPI
inference for 1000 images(aprox cost)	Yes	$4/hr	yes	3.5 USD per node hr, Deployment 1.82 USD/node hr , Batch prediction 2.02 USD per node hr	yes	0-1m RS.66/1000 images , 1-5m RS.52/1000 images,5m above 42/100 images

Future Work

- Future work: We would like to make similar experiments of object detection on the cloud for medical images and see the performance
- AWS Sage maker integration along with the Rekognition needs to be checked for object detection.

Success Stories

Senior IT leaders share how our services helped them win in the platform age.

CTO Speak

"Data pipeline buildout, Software development, Salesforce development, AWS System admin/DevOps, BI/Dashboard. The execution has been very good.

- Satyadeep "Bobby" Patnaik, CTO

Lafayette Square

CEO Speak

"They understood that product development was iterative and they patiently worked through our requirements even as they rapidly evolved.

- Dr. Ganesh Naidoo, CEO

med mate

CTO Speak

"Proven technical ability in both web and mobile development; strong project/product management expertise; the ability to become part of the extended BA365 team.

- Graeme Dollar, CTO

BUSINESS ACCELERATOR 365

COO Speak

"I have worked with hundreds of service providers and consultants, RoundSqr (Part of Cigniti) is absolutely one of the best. The people are highly skilled, very hard-working, and have a "can do" attitude.

- Mark Mortimer, COO

Adelphoi

Analyst Recognitions

NelsonHall recognized Cigniti as a "LEADER" i n its 2022 NEAT evaluation for Overall **Quality** Engineering, Continuous Testing, Application **Security Testing, and AI &** Cognitive

Cigniti is mentioned as a "Pure Play Testing Vendor" in Gartner's Market Guide for Application Testing Services, 2022

Cigniti is mentioned as "API Testing Vendor" in Gartner's Hype Cycle for Managed IT Services and APIs, 2022

Recognized as "LEADER" in ISG IPL for Next-Gen ADM Services under Continuous Testing Specialists category for the US region for 2021 and 2022

Recognized as "RISING STAR" in UK Region in ISG IPL for Next-Gen ADM Services 2022

Cigniti is recognized as a **Strong Performer** in the Forrester Wave: **Continuous Automation and Testing** Services, Q3 2021.

One of the top 3 leaders in Agile Testing and DevOps in the Forrester Wave: Continuous Automation and Testing Services, Q3 2017.

Provides Cigniti with **"Best in Class"** rating for Buyer satisfaction.

About Cigniti

Cigniti Technologies Limited (NSE: CIGNITITEC; BSE: 534758) is the World's Leading AI & IP-led Digital Assurance and Digital Engineering Services Company providing software quality engineering, software testing, automation, and consulting services. 4100+ Cignitians worldwide help Fortune 500 & Global 2000 enterprises across 24 countries accelerate their digital transformation journey across various stages of digital adoption and help them achieve market leadership by providing transformation services leveraging IP & Platform-led innovation with expertise across multiple verticals and domains.

Our global customers have benefitted with measurable outcomes, millions of dollars of savings, significant ROI, and delightful, frictionless experiences utilizing our flagship digital assurance and full cycle software quality engineering services. Our AI-led digital engineering services cover Data engineering services, softwareplatform, and digital product engineering, AI/ML engineering services, intelligent automation, big data analytics, and Blockchain development.

We are headquartered in Hyderabad, India, with global offices spread across the USA, Canada, UK, UAE, Australia, South Africa, Czech Republic, and Singapore.

To learn more, visit www.cigniti.com

Website LinkedIn YouTube Blog Facebook Twitter