Microsoft

Construyendo un detector de objetos con Azure Custom Vision .NET SDK

Luis Beltrán
Microsoft MVP

Luis Beltrán

- Investigador en Tomas Bata University en Zlín, República Checa.
- Docente en Tecnológico Nacional de México en Celaya, Mexico.
- Alto interés en Xamarin, Azure e Inteligencia Artificial

@darkicebeam

luis@luisbeltran.mx

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky

TECNOLÓGICO NACIONAL DE MÉXICO

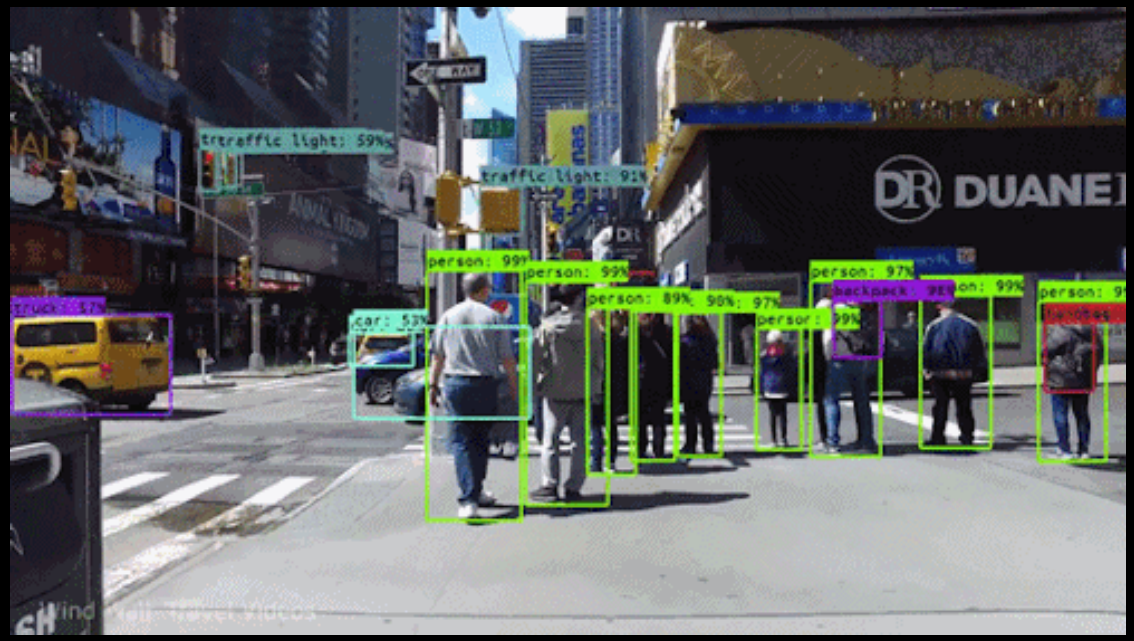
ITCelaya

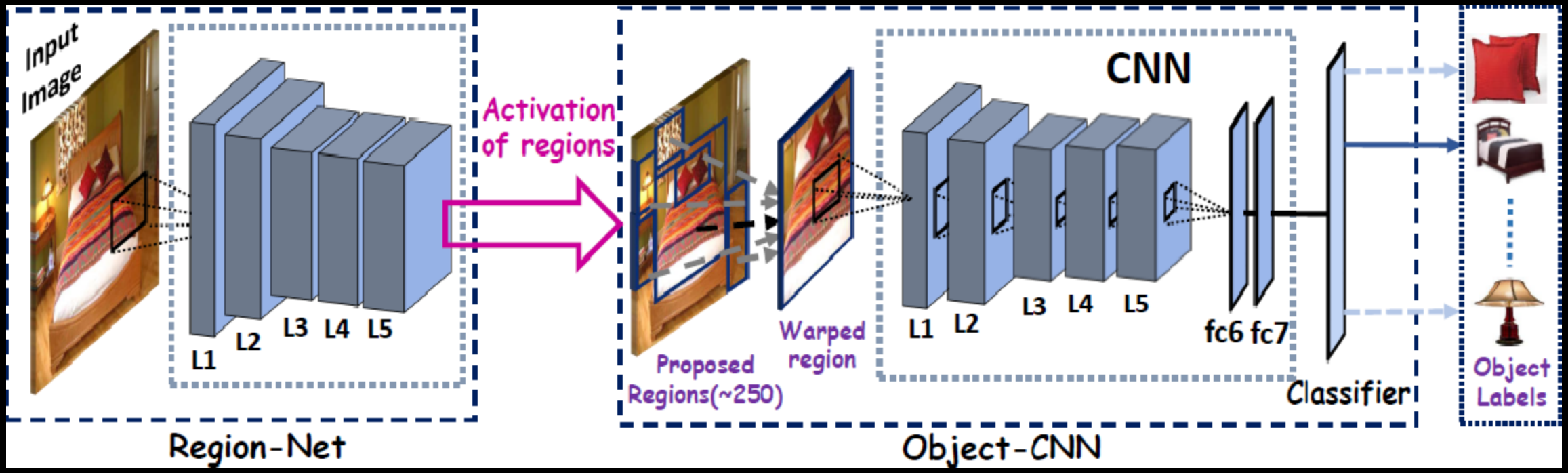
MVP Microsoft® Most Valuable Professional

Slides disponibles en el sitio del evento o en https://bit.ly/Luis24Horas

Detección de objetos

Es el proceso de **identificar** elementos de interés en videos e imágenes y además, de **localizarlo** dentro de la imagen/video, típicamente a través de las coordenadas de la “caja” (***bounding box***) que lo contiene.

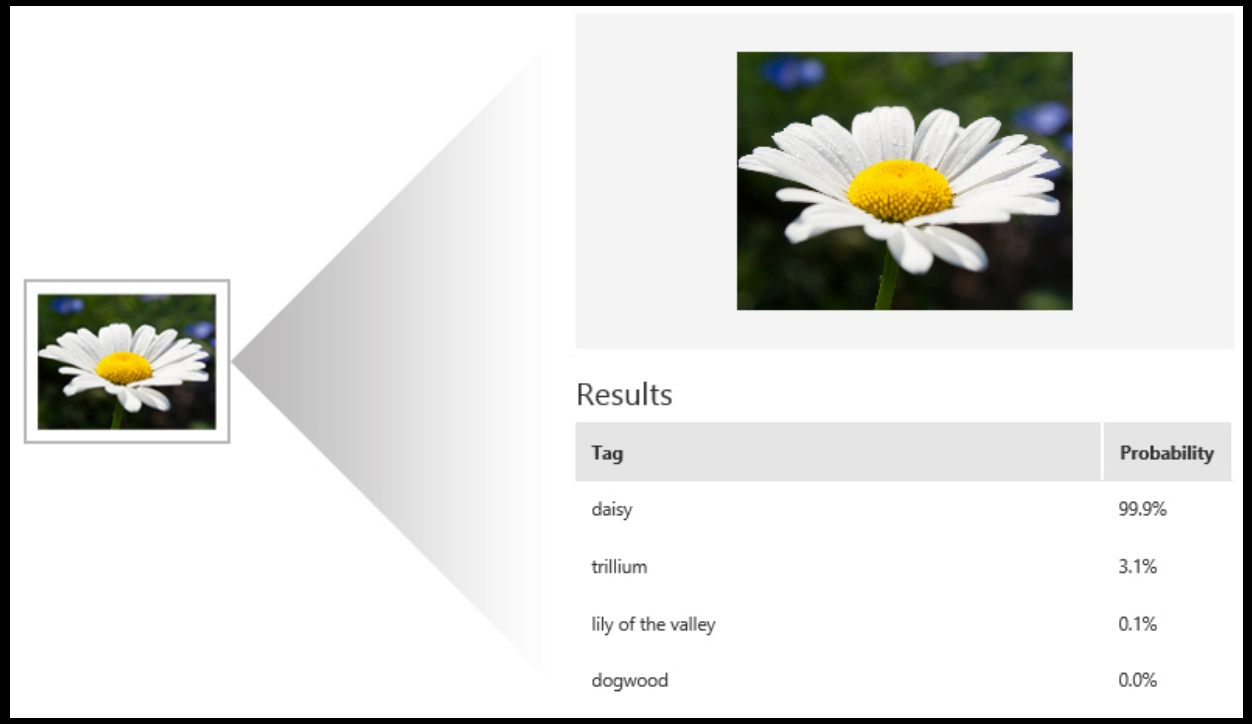



Input Image
L1
L2
L3
L4
L5
Region-Net
Activation of regions
Proposed Regions(~250)
Warped region
CNN
L1
L2
L3
L4
L5
fc6
fc7
Classifier
Object Labels
Object-CNN

Custom Vision Service

Reconocimiento de imágenes adaptado a las necesidades de tu caso de uso.

Un servicio y plataforma de IA para aplicar visión de computadora personalizada en tus escenarios específicos



customvision.ai

Construyendo un detector de objetos

Crea un Proyecto

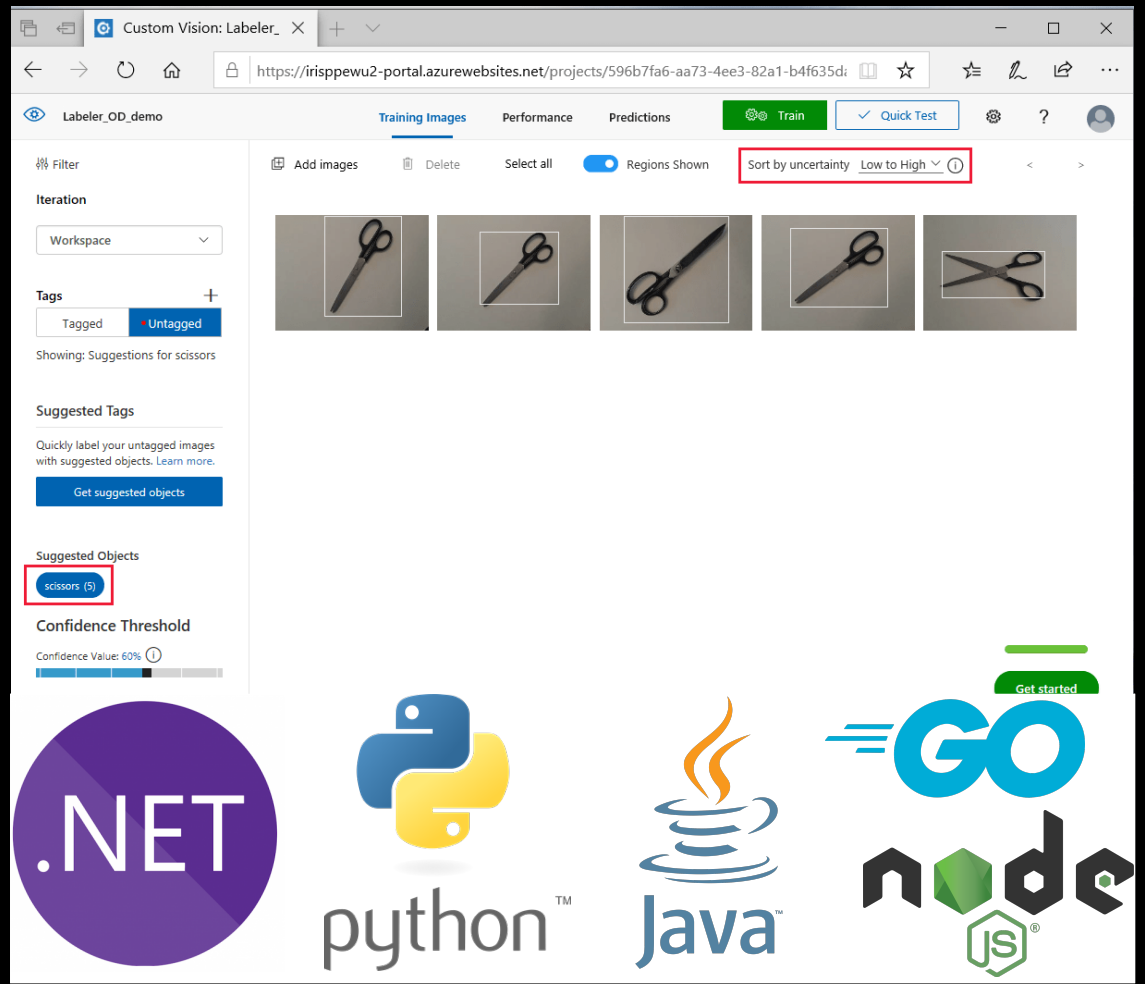
- Elige un dominio

Carga imágenes

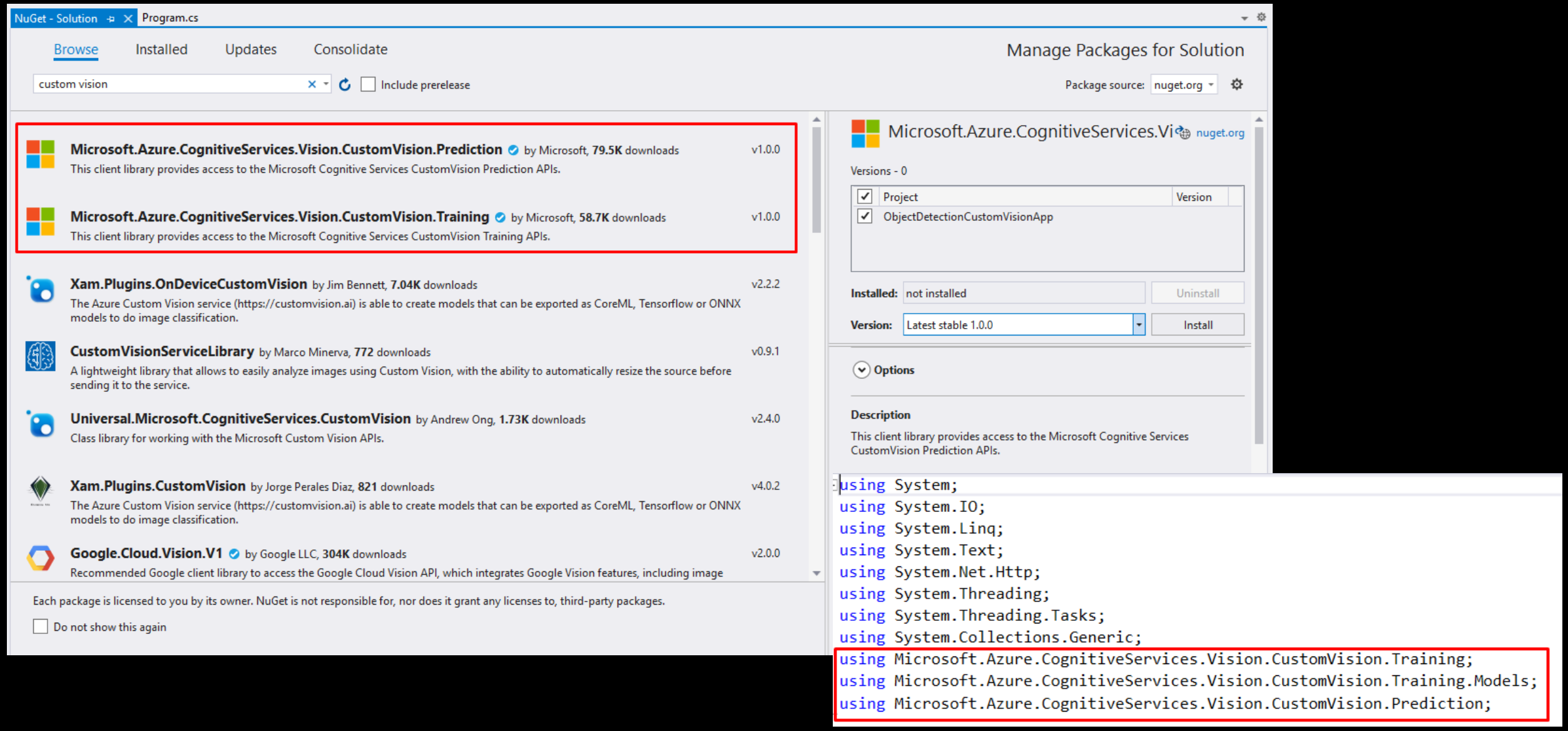
- Localiza el objeto de interés
- Etiquétalo

Entrena el detector

Evalúa, publica y exporta el modelo



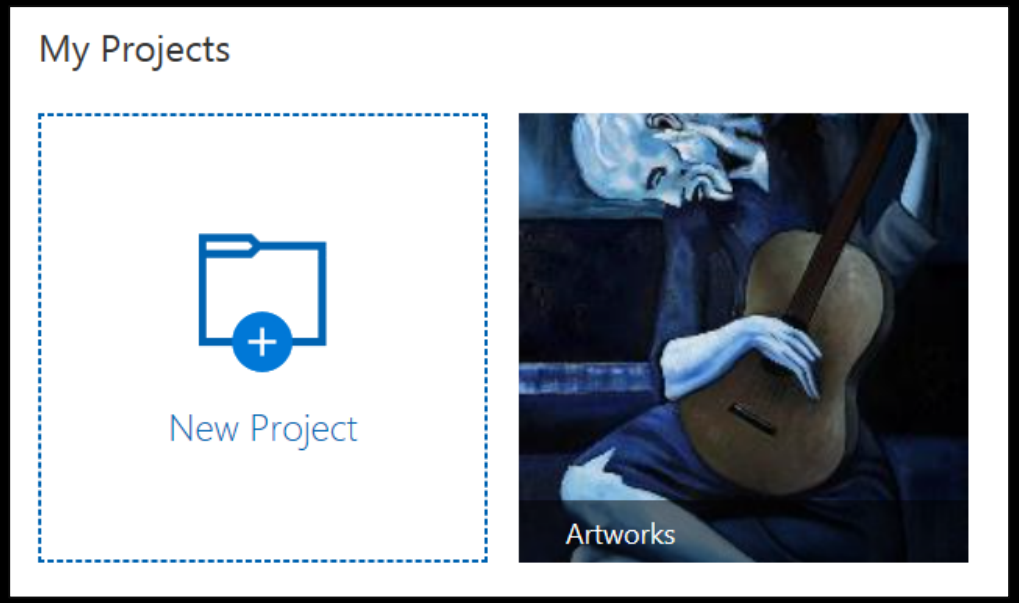
Custom Vision .NET SDK



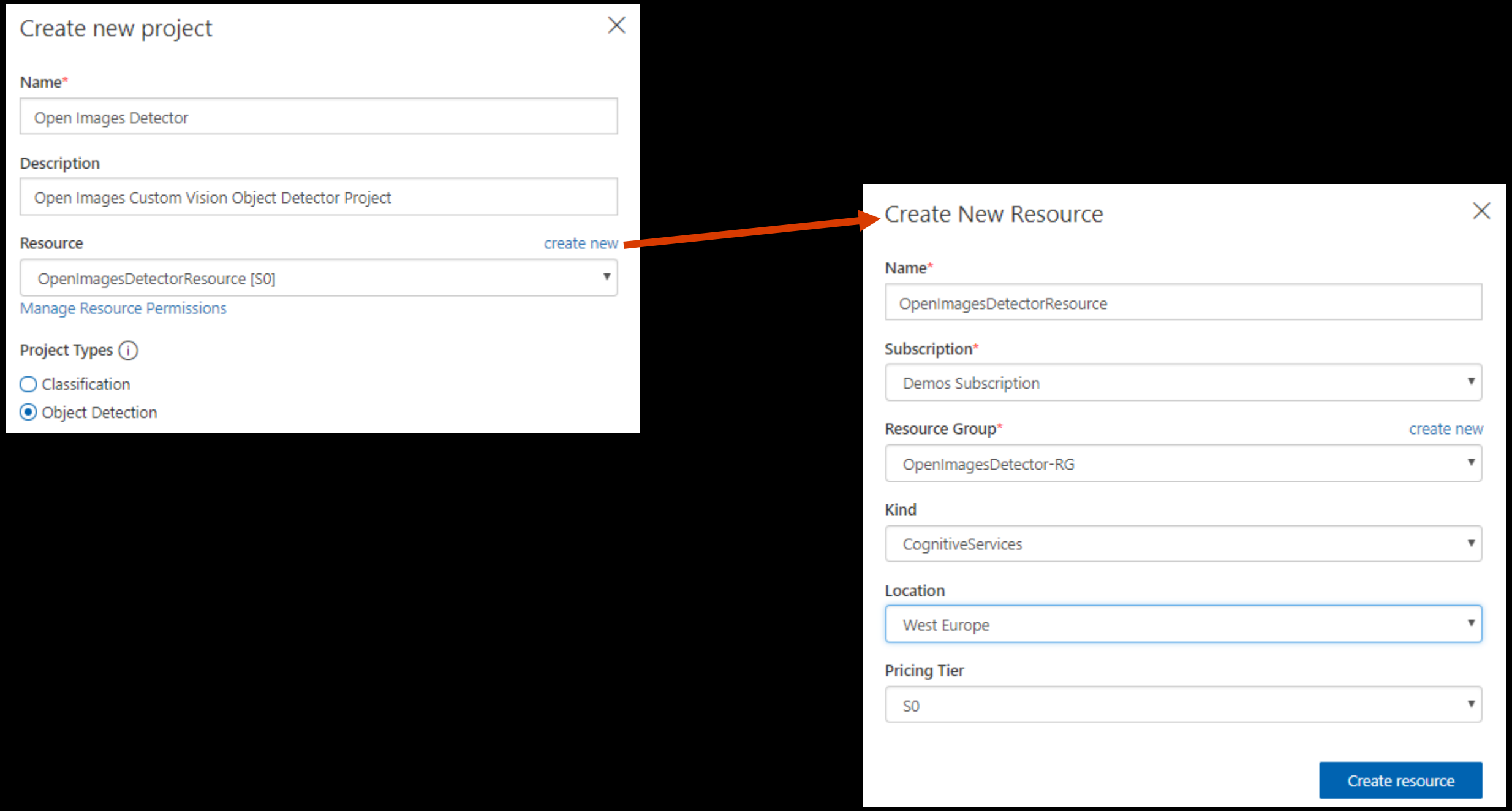
Creando un proyecto

Custom Vision está organizado de forma jerárquica. En el nivel más alto se encuentra el proyecto, el cual representa los datos y el modelo generado para una tarea específica.

Un **detector de objetos** es un modelo construido con el servicio de Custom Vision por medio de imágenes de entrenamiento.



Object Detector = Project

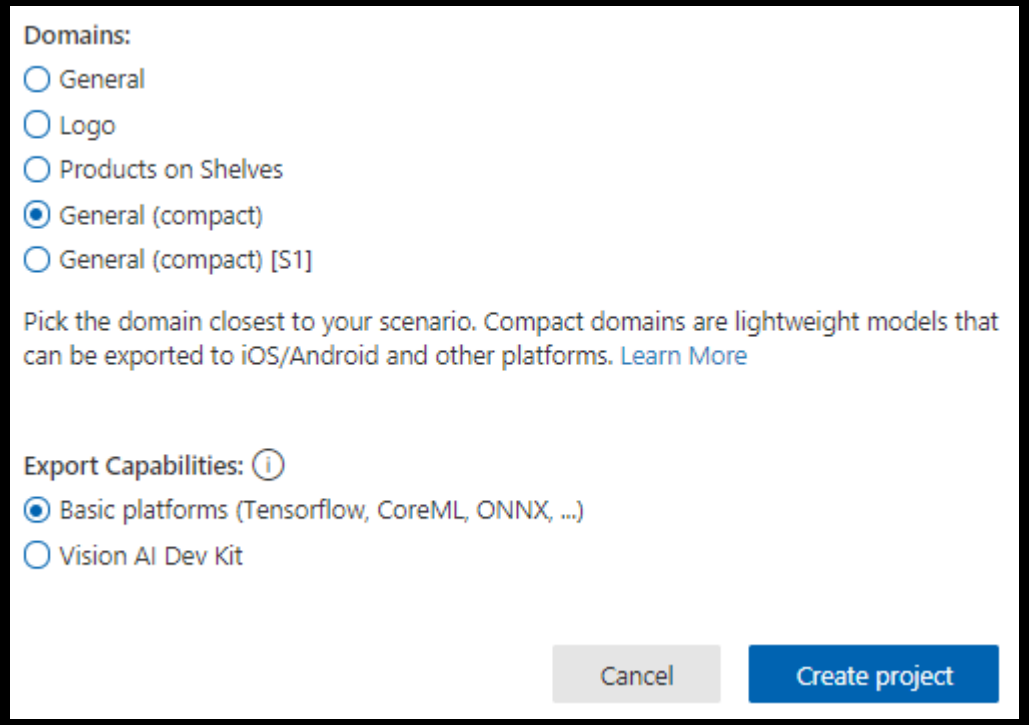

Create new project
Name*
Open Images Detector
Description
Open Images Custom Vision Object Detector Project
Resource
create new
OpenImagesDetectorResource [S0]
Manage Resource Permissions
Project Types
Classification
Object Detection
Create New Resource
Name*
OpenImagesDetectorResource
Subscription*
Demos Subscription
Resource Group*
create new
OpenImagesDetector-RG
Kind
CognitiveServices
Location
West Europe
Pricing Tier
S0
Create resource

Eligiendo un dominio

Cuando se crea un proyecto de detección de objetos, se elige un dominio, el cual **optimiza** el detector para reconocer elementos que pertenecen a ciertos subconjuntos:

- **Logo:** logos de marcas.
- **Products on Shelves:** productos en estantes.
- **General:** para todo lo demás.

Además, existen los dominios **Compact,** que son detectores de objetos ligeros optimizados para sistemas con restricción de tiempo.



Project Settings

General

Project Name*

Open Images Detector

Resources:

OpenImagesDetectorResource

Subscription: Demos Subscription

Resource Group: OpenImagesDetector-RG

Resource Kind: All Cognitive Services

Key:

[redacted]

Endpoint:

https://westeurope.api.cognitive.microsoft.com/

Resource Id:

/subscriptions/[redacted]/resourceG

Pricing Tier: S0

1 projects created; 99 remain

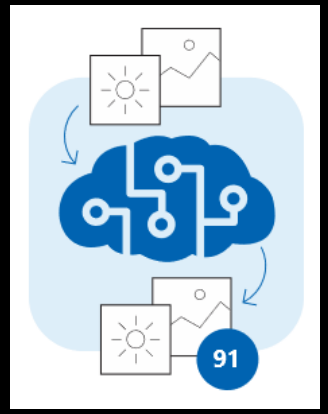
Unlimited

```
// Project name must match the one in Custom Vision
var projectName = "Open Images Detector";
var publishedModelName = "OpenImagesDetectorModel";

// Replace with your own from Custom Vision project
var customVisionKey = "[redacted]";
var endpoint = "https://westeurope.api.cognitive.microsoft.com/";
var resourceId = "/subscriptions/[redacted]/re
```

```
// Training Client
var trainingApi = new CustomVisionTrainingClient()
{
    ApiKey = customVisionKey,
    Endpoint = endpoint
};

Console.WriteLine($"----- Selecting existing project: {projectName}... -----");

var projects = await trainingApi.GetProjectsAsync();
var project = projects.FirstOrDefault(x => x.Name == projectName);

if (project != null)
{
    Console.WriteLine($"\t{projectName} found in Custom Vision workspace.");

    var domainId = project.Settings.DomainId;
    var projectDomain = await trainingApi.GetDomainAsync(domainId);

    Console.WriteLine($"\tProject domain: {projectDomain.Name}.");
    Console.WriteLine($"\tProject type: {projectDomain.Type}.");

    WriteSeparator();
}
```

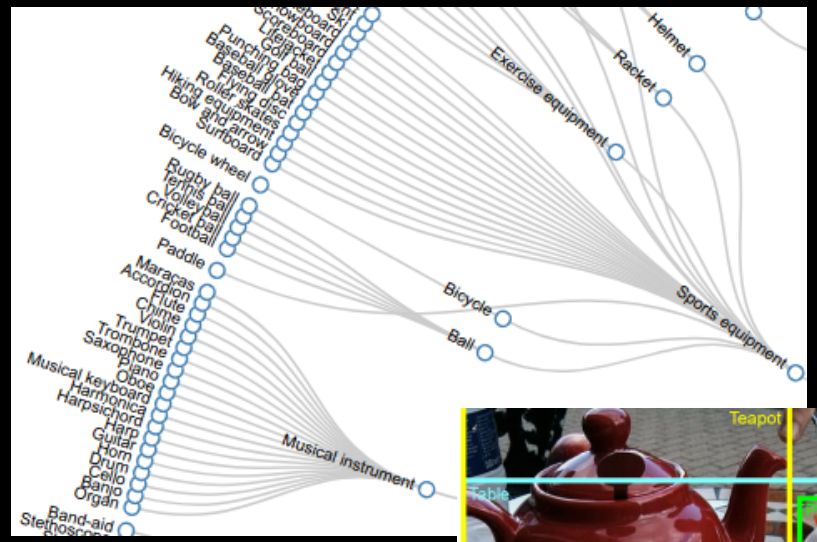
Cargando y etiquetando imágenes

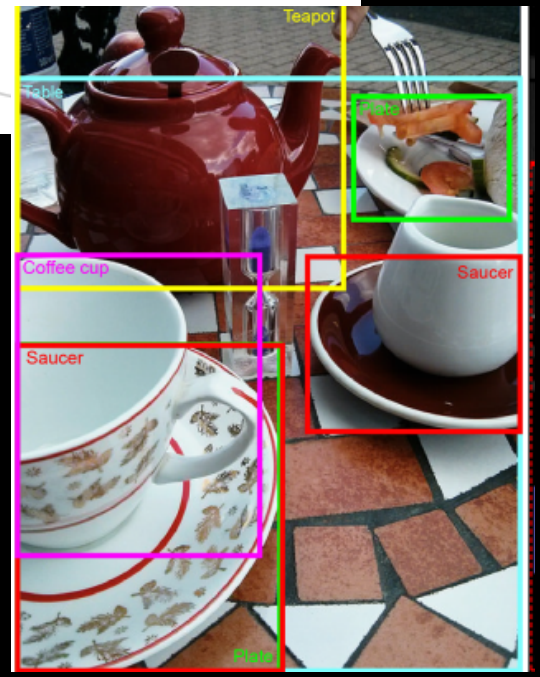
Para crear un detector de objetos con una precisión alta y confiable, Custom Vision Service requiere de varias **imágenes de entrenamiento**.



Una imagen de entrenamiento es una fotografía en la que el servicio de Custom Vision identifica de qué objeto (**tag**) se trata y su **ubicación** dentrp de la imagen.

Google Open Images Dataset V4





Open Images is a dataset of **~9M images** that have been annotated with **image-level labels**, **object bounding boxes** and **visual relationships**.

The training set of V4 contains **14.6M bounding boxes** for **600 object classes** on **1.74M images**, making it the largest existing dataset with object location annotations.

The boxes have been largely manually drawn by professional annotators to ensure accuracy and consistency.

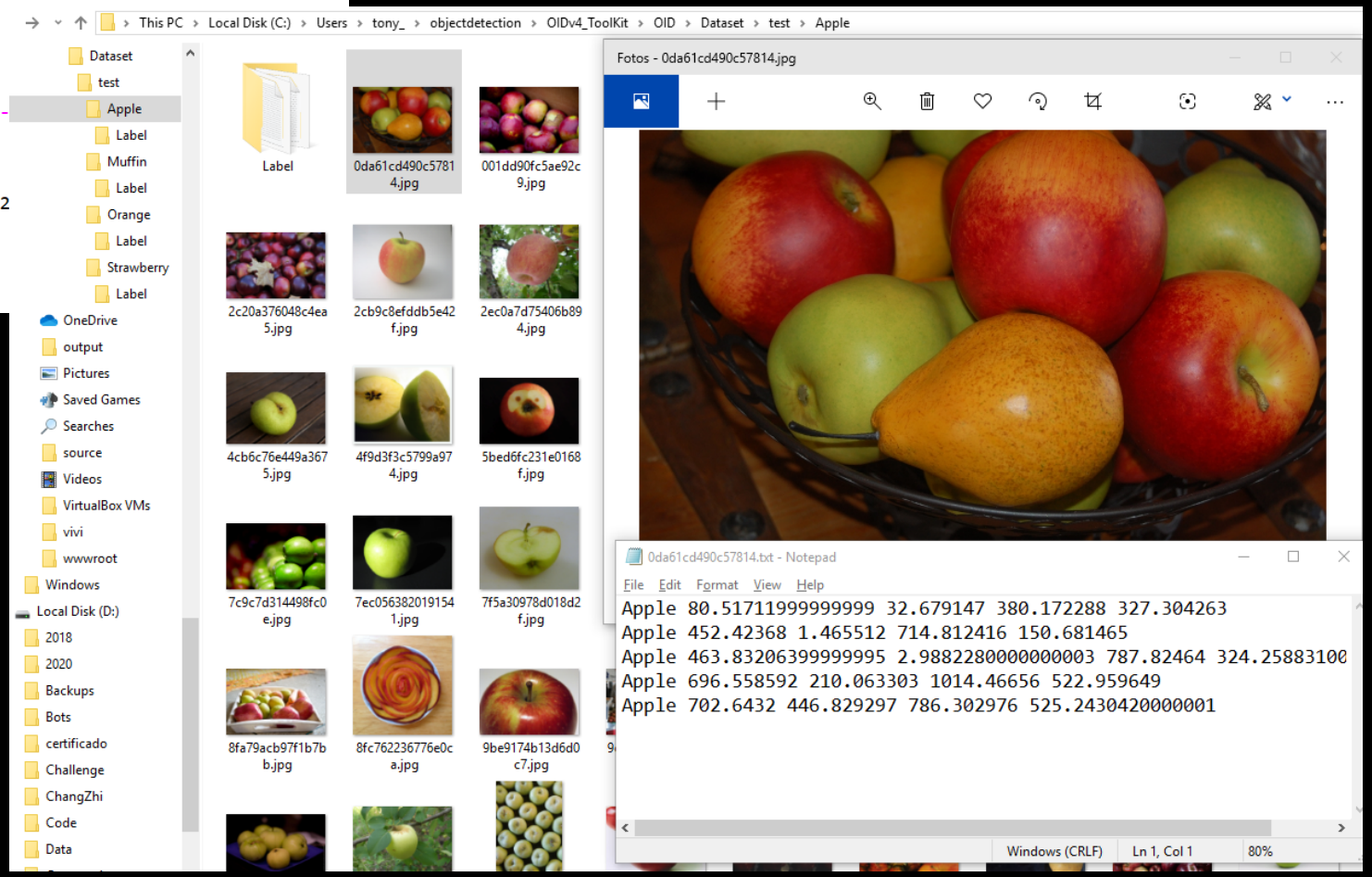
The images are very diverse and often contain complex scenes with several objects (8.4 per image on average). This also encorages structural image annotations, such as visual relationships.

Moreover, the dataset is annotated with image-level labels spanning thousands of classes.

https://storage.googleapis.com/openimages/web/factsfigures_v4.html

OIDv4 Toolkit

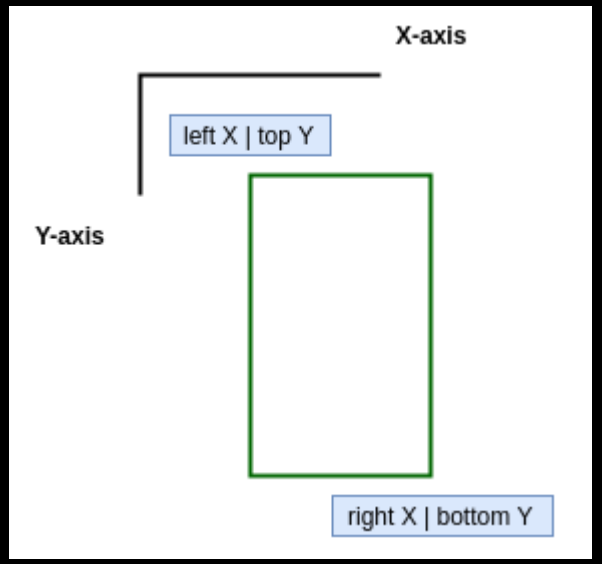
```
------------------------------------------------------Apple-------------------------------------------------------
    [INFO] | Downloading test images.
    [INFO] | [INFO] Found 144 online images for test.
    [INFO] | Download of 144 images in test.
100%|                                                                                | 144/144 [10:10<00:00,  4.24s/it]]
  [INFO] | Done!
    [INFO] | Creating labels for Apple of test.
    [INFO] | Labels creation completed.
    [INFO] | Downloading Orange.

------------------------------------------------------Orange------------------------------------------------------
    [INFO] | Downloading test images.
    [INFO] | [INFO] Found 208 online images for test.
    [INFO] | Download of 208 images in test.
100%|                                                                                | 2
    [INFO] | Done!
    [INFO] | Creating labels for Orange of test.
    [INFO] | Labels creation completed.
    [INFO] | Downloading Muffin.
```



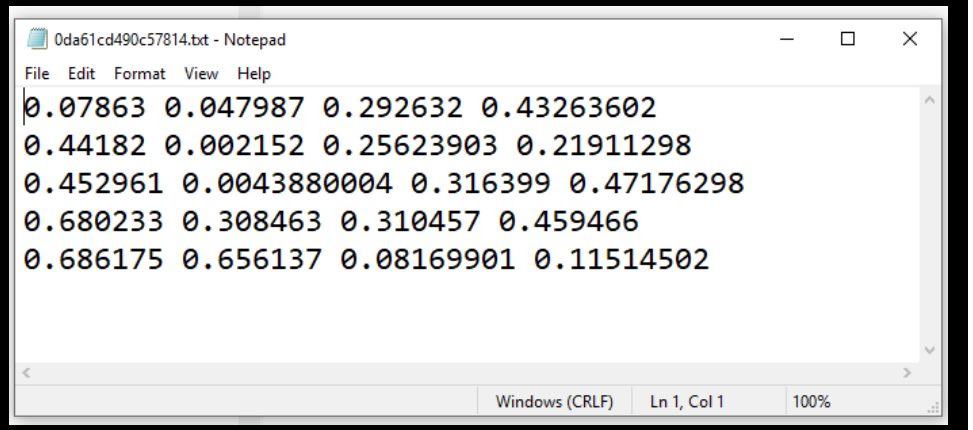
https://github.com/EscVM/OIDv4_ToolKit

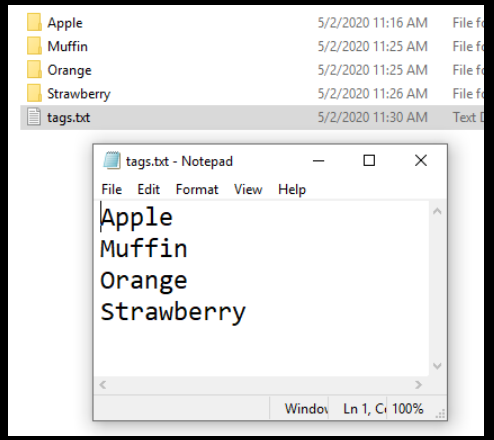
OIDv4ToCV project

- Custom Vision requiere **coordenadas normalizadas de imagen [0, 1]** (x_min, y_min, width, height).
- Google Open Images Dataset v4 está **normalizado** (x_min, x_max, y_min, y_max).
- OIDv4 Toolkit genera **coordenadas desnormalizadas (pixels en la imagen)** (x_min, y_min, x_max, y_max)
- OIDv4ToCV "des-desnormaliza" las coordenadas para su uso en Custom Vision.

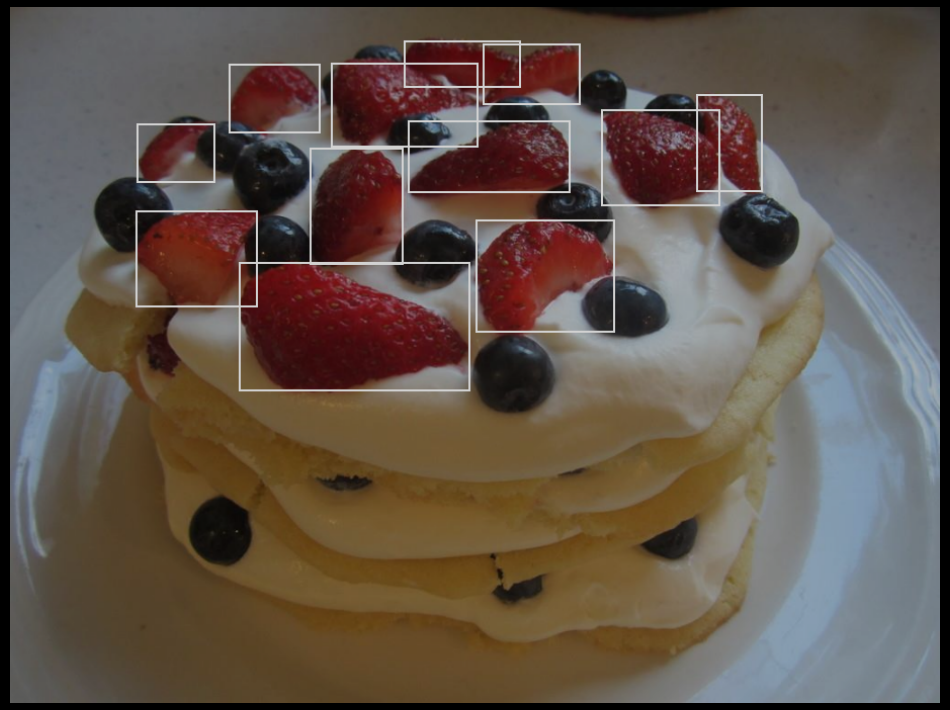


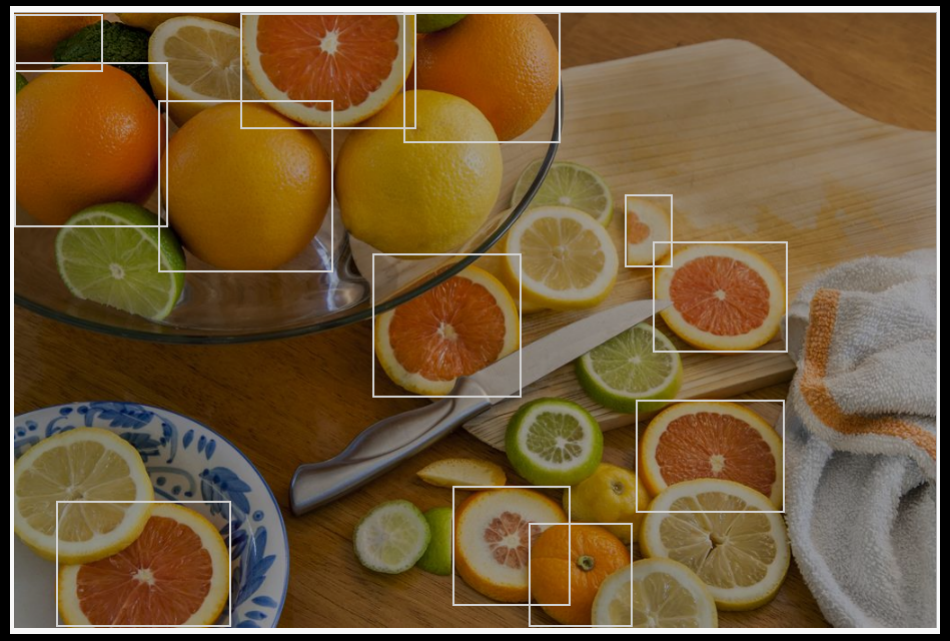

```
----- Apple Class-----
       Image: 001dd90fc5ae92c9.jpg
              Normalized Data: 0 0 1 1
       Image: 08d2eed131f4b5d9.jpg
              Normalized Data: 0 0 0.220249 0.326184
              Normalized Data: 0.001591 0.337568 0.177223 0.339766
       Image: 096ca56a5c51f6d6.jpg
              Normalized Data: 0 0.00095699995 1 0.999043
       Image: 0da61cd490c57814.jpg
              Normalized Data: 0.07863 0.047987 0.292632 0.43263602
              Normalized Data: 0.44182 0.002152 0.25623903 0.21911298
              Normalized Data: 0.452961 0.0043880004 0.316399 0.47176298
              Normalized Data: 0.680233 0.308463 0.310457 0.459466
              Normalized Data: 0.686175 0.656137 0.08169901 0.11514502
```





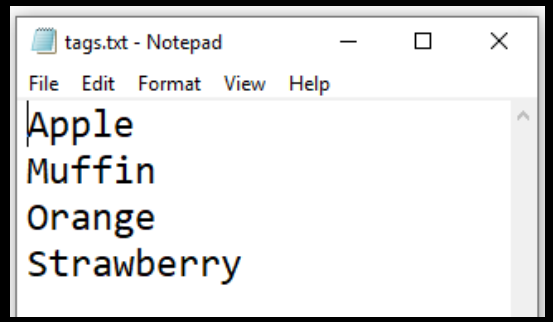
https://github.com/icebeam7/OIDv4ToCV



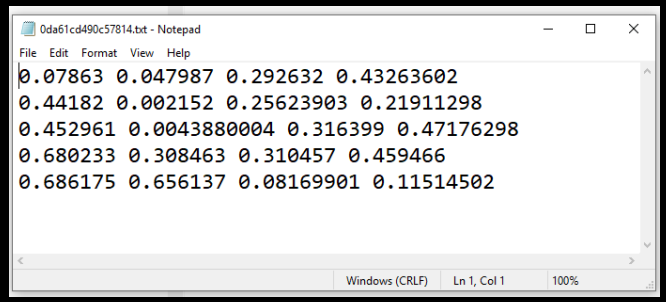



Happy
Birthday!
Tori & Tara

```
// Retrieve the tags that already exist in the project
Console.WriteLine("----- Retrieving tags... -----");

var modelTags = await trainingApi.GetTagsAsync(project.Id);

var tags = new List<Tag>();
var onlineImages = 0;

// Obtain tags from our dataset
var tagsFile = Path.Combine(trainingFolder, "tags.txt");
var imageLabels = await File.ReadAllLinesAsync(tagsFile);
```




```
foreach (var label in imageLabels)
{
    // Check if the label already exists
    var tag = modelTags.FirstOrDefault(x => x.Name == label);

    if (tag == null)
    {
        // If not, create it
        tag = await trainingApi.CreateTagAsync(project.Id, label);

        Console.WriteLine($"\tTag {tag.Name} was created.");
    }
    else
    {
        // If so, just count images with this tag
        onlineImages += tag.ImageCount;
        Console.WriteLine($"\tTag {label} was NOT created (it already exists)");
    }

    tags.Add(tag);
}
```

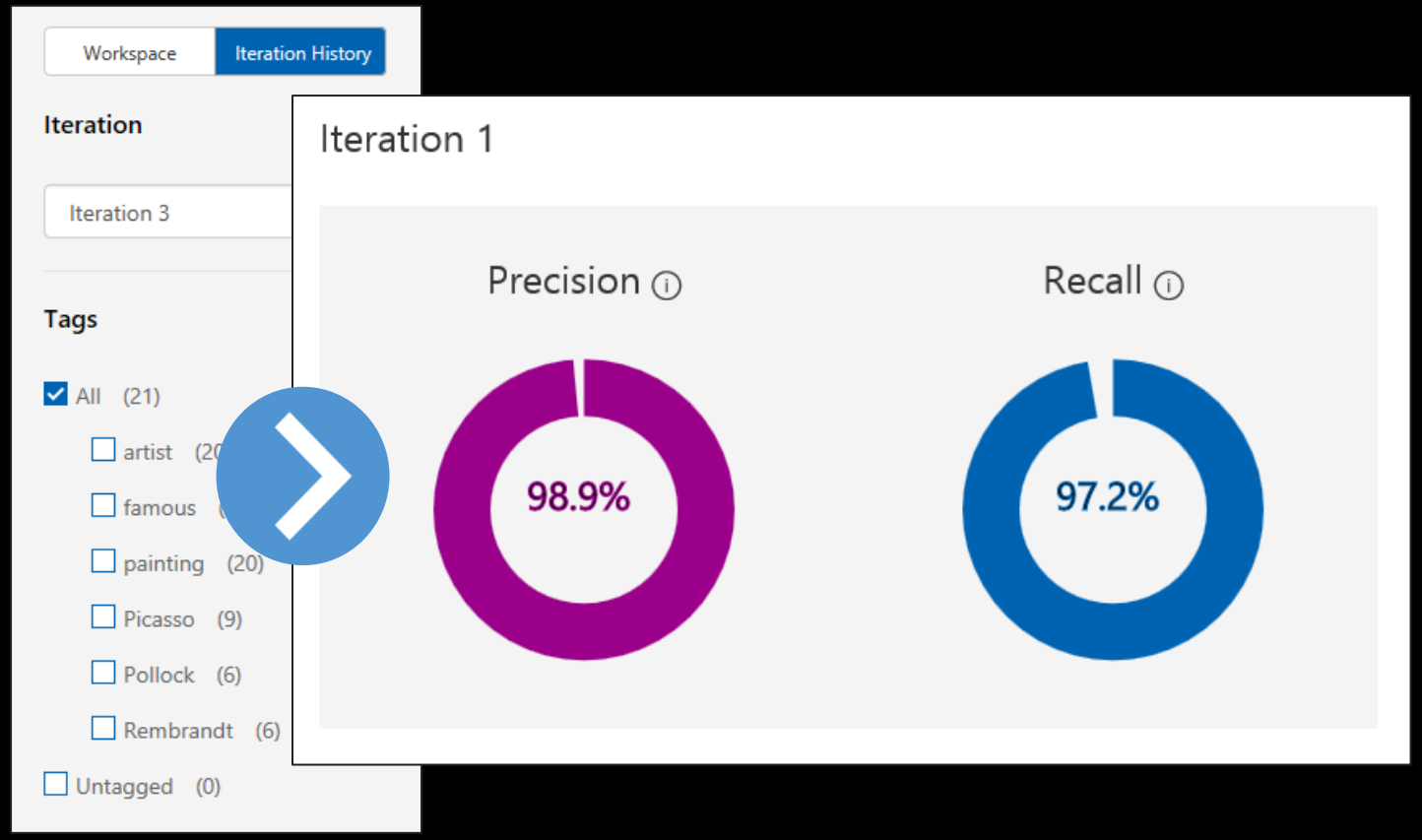



```
Console.WriteLine("----- Accessing images -----");
var imageFileEntries = new List<ImageFileCreateEntry>();

foreach (var label in imageLabels)
{
    var tagFolder = Path.Combine(trainingFolder, label);
    var tagImages = Directory.GetFiles(tagFolder);
    var tagLabels = Path.Combine(tagFolder, "normalizedLabel");

    foreach (var image in tagImages)
    {
        var imageName = Path.GetFileNameWithoutExtension(image);
        var imageFile = Path.GetFileName(image);
        var imageLabelFile = $"{imageName}.txt";
        var tagCV = tags.Single(x => x.Name == label);

        var imageBoundingBoxes = await File.ReadAllLinesAsync(Path.Combine(tagLabels, imageLabelFile));
        var imageRegions = new List<Region>();

        foreach (var bbox in imageBoundingBoxes)
        {
            var normalizedData = bbox.Split();
            var left = float.Parse(normalizedData[0]);
            var top = float.Parse(normalizedData[1]);
            var width = float.Parse(normalizedData[2]);
            var height = float.Parse(normalizedData[3]);

            imageRegions.Add(new Region(tagCV.Id, left, top, width, height));
        }

        Console.WriteLine($"\tAdding image {imageName} with its regions.");

        imageFileEntries.Add(new ImageFileCreateEntry(
            imageFile, await File.ReadAllBytesAsync(image), null, imageRegions));
    }
}
```

```
var batchPageSize = 64;
var numBatches = imageFileEntries.Count / batchPageSize;
var sizeLastBatch = imageFileEntries.Count % batchPageSize;

for (int batch = 0; batch < numBatches; batch++)
{
    Console.WriteLine($"\tUploading images batch #{batch}.");

    var entries = imageFileEntries.Skip(batch * batchPageSize).Take(batchPageSize).ToList();

    await trainingApi.CreateImagesFromFilesAsync(
        project.Id, new ImageFileCreateBatch(entries));
}

if (sizeLastBatch > 0)
{
    Console.WriteLine($"\tUploading last batch.");
    var lastEntries = imageFileEntries.Skip(numBatches * batchPageSize).Take(sizeLastBatch).ToList();

    await trainingApi.CreateImagesFromFilesAsync(project.Id,
        new ImageFileCreateBatch(lastEntries));
}
```

Entrenando el detector de objetos

Cada vez que se (re)entrena el detector, se está creando una iteración del modelo.



```
// Now there are images with tags start training the project
Console.WriteLine("----- Starting the Training process... -----");

iteration = await trainingApi.TrainProjectAsync(project.Id);

// The returned iteration will be in progress, and can be queried periodically to see when it has completed
while (iteration.Status == "Training")
{
    Thread.Sleep(1000);
    Console.WriteLine($"\tIteration '{iteration.Name}' status: {iteration.Status}");

    // Re-query the iteration to get it's updated status
    iteration = await trainingApi.GetIterationAsync(project.Id, iteration.Id);
}

Console.WriteLine($"\tIteration '{iteration.Name}' status: {iteration.Status}");
```

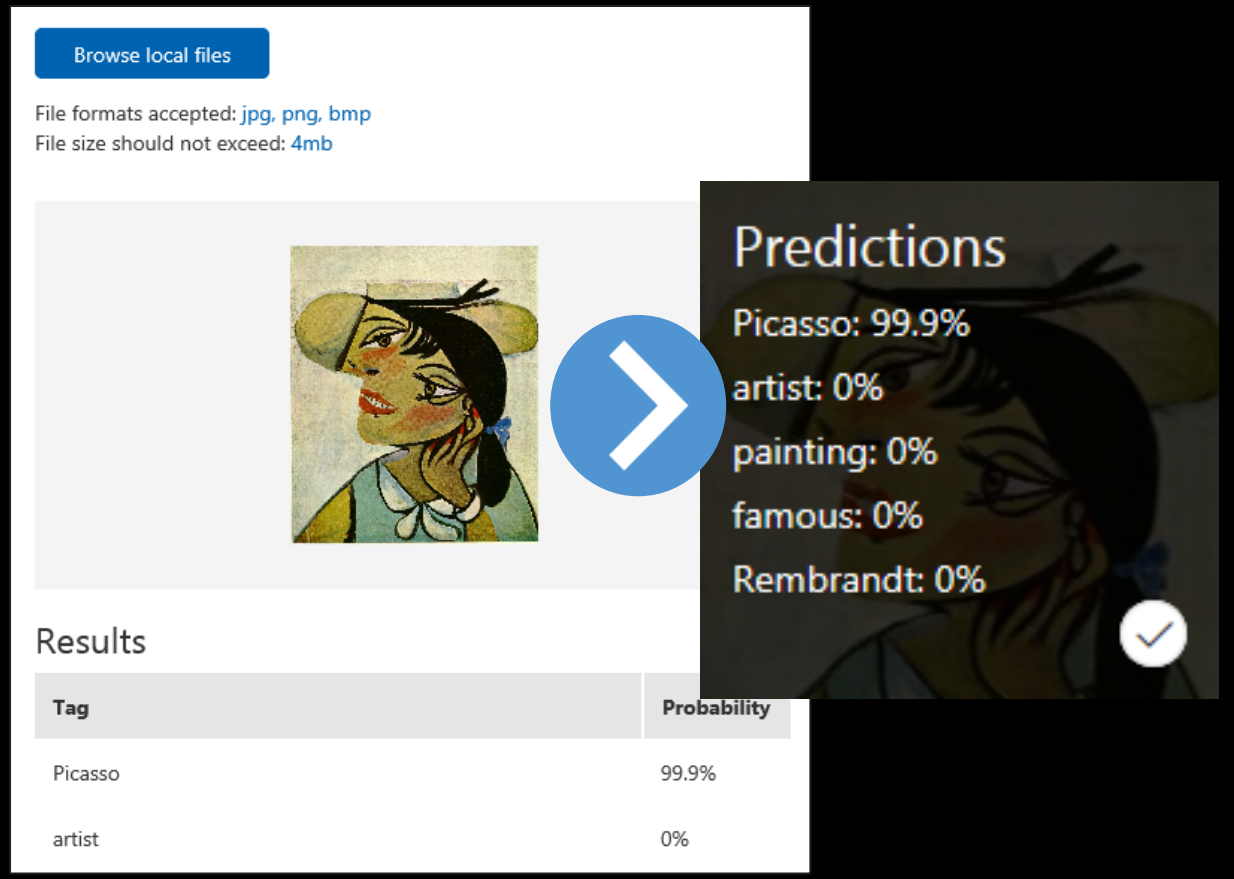
Evaluando y probando el modelo

Después de entrenar el modelo, éste puede **probarse** rápidamente con nuevas imágenes.

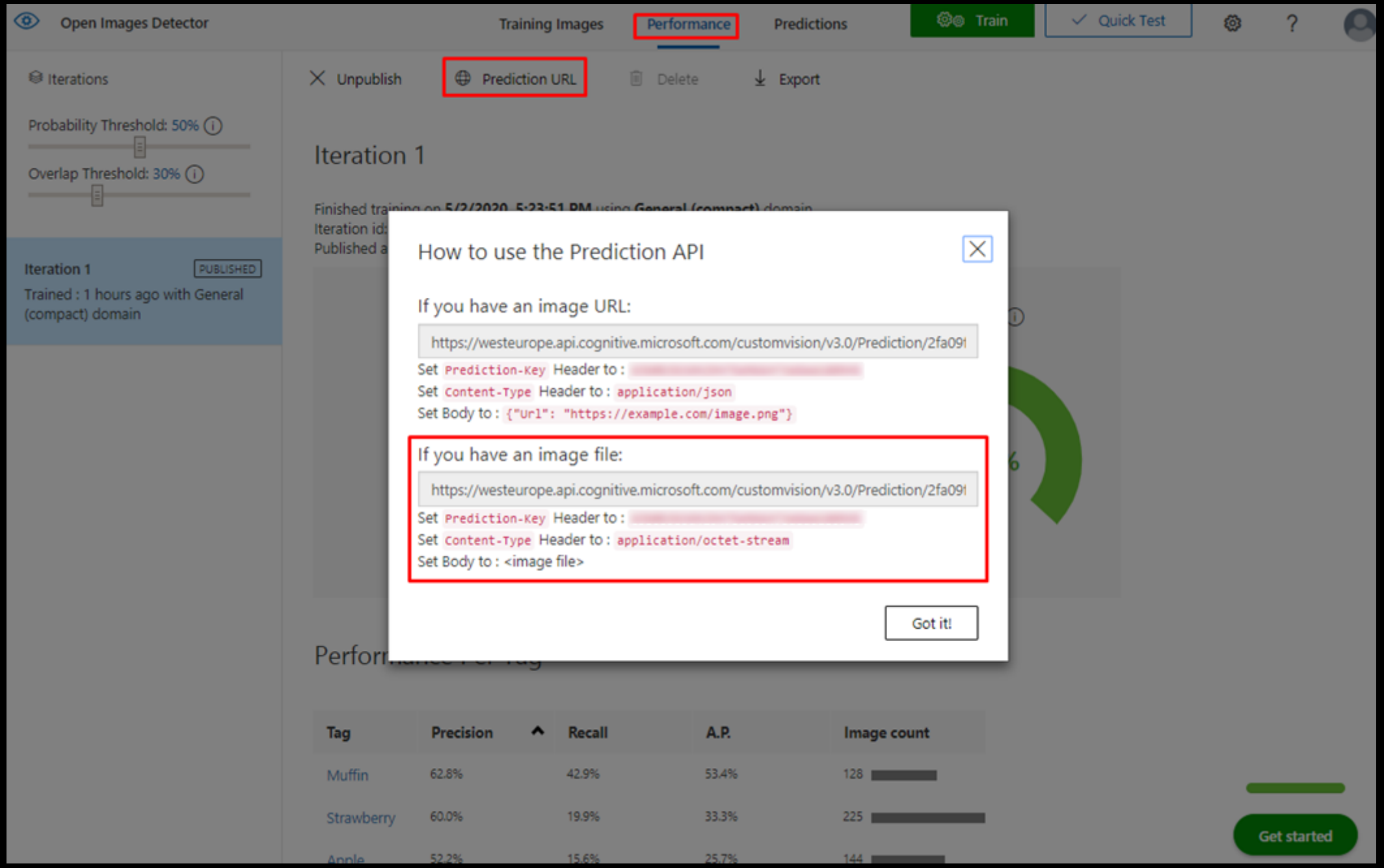
La prueba utiliza la última iteración (la más reciente).

Si el modelo proporciona resultados precisos, puede ser publicado, haciendo que el detector:

- Pueda **accederse** desde un **endpoint HTTP** (o a través del **SDK**) para detección de objetos online, a través de Internet.
- Pueda **exportase** a una plataforma para una detección de objetos sin requerir Internet (en modo offline).



```
// The iteration is now trained. Publish it to the prediction endpoint.
Console.WriteLine($"----- Starting the Publication process. -----");

await trainingApi.PublishIterationAsync(
    project.Id, iteration.Id, publishedModelName, resourceId);

Console.WriteLine($"\tIteration '{iteration.Name}' published.");
```


Open Images Detector
Training Images
Performance
Predictions
Train
Quick Test
Iterations
Unpublish
Prediction URL
Delete
Export
Probability Threshold: 50%
Overlap Threshold: 30%
Iteration 1
PUBLISHED
Trained : 1 hours ago with General (compact) domain
Iteration 1
How to use the Prediction API
If you have an image URL:
https://westeurope.api.cognitive.microsoft.com/customvision/v3.0/Prediction/2fa09
Set Prediction-Key Header to :
Set Content-Type Header to : application/json
Set Body to : {"Url": "https://example.com/image.png"}
If you have an image file:
https://westeurope.api.cognitive.microsoft.com/customvision/v3.0/Prediction/2fa09
Set Prediction-Key Header to :
Set Content-Type Header to : application/octet-stream
Set Body to : <image file>
Got it!
Tag
Precision
Recall
A.P.
Image count
Muffin
62.8%
42.9%
53.4%
128
Strawberry
60.0%
19.9%
33.3%
225
Get started

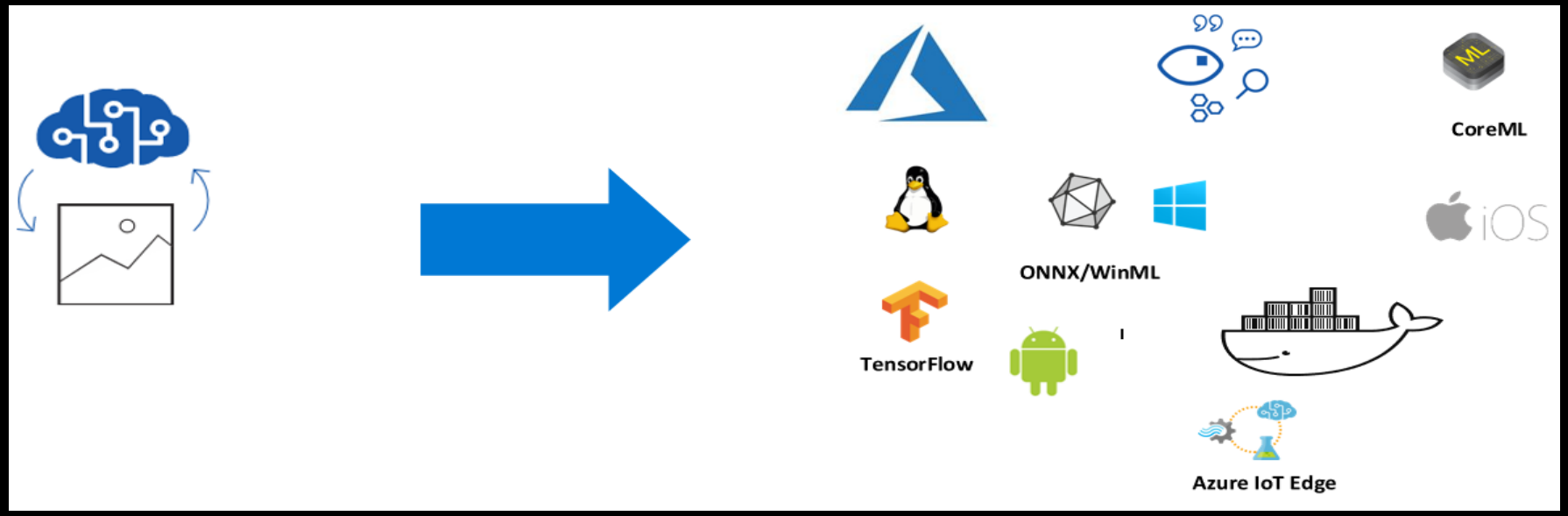
```
// Prediction Client
var predictionClient = new CustomVisionPredictionClient()
{
    ApiKey = customVisionKey,
    Endpoint = endpoint
};
```

```
var testImages = LoadImagesFromDisk("Test");

foreach (var image in testImages)
{
    var imageName = Path.GetFileName(image);

    using (var stream = new MemoryStream(File.ReadAllBytes(image)))
    {
        Console.WriteLine($"\tImage: {imageName}");

        var result = await predictionClient.DetectImageAsync(
            project.Id, publishedModelName, stream);

        // Loop over each prediction and write out the results
        foreach (var prediction in result.Predictions.OrderByDescending(x => x.Probability))
        {
            Console.WriteLine($"\t\tFor Tag '{prediction.TagName}': {prediction.Probability:P3} " +
                $"[{prediction.BoundingBox.Left}, {prediction.BoundingBox.Top}, " +
                $"{prediction.BoundingBox.Width}, {prediction.BoundingBox.Height}]");
        }

        WriteSeparator();
        Console.WriteLine("Press any key for next image...");
        Console.ReadKey();
    }
}
```

Ejecuta el modelo donde desees

Ejecuta tus modelos donde los necesites, de acuerdo a tus escenarios y requerimientos únicos.

Exporta tus modelos entrenados a dispositivos o contenedores para escenarios de baja latencia.



Domains:

- ○ General
- ○ Logo
- ○ Products on Shelves
- ◉ General (compact)
- ○ General (compact) [S1]

Pick the domain closest to your scenario. Compact domains are lightweight models that can be exported to iOS/Android and other platforms. Learn More

Export Capabilities: ⓘ

- ◉ Basic platforms (Tensorflow, CoreML, ONNX, ...)
- ○ Vision AI Dev Kit

Cancel | Create project

Training Images | Performance | Predictions

↓ Export

Choose your platform

- iOS — CoreML — iOS 11
- TF — TensorFlow — Android
- ONNX — ONNX — Windows ML
- DF — Dockerfile — Azure IoT Edge, Azure Functions, AzureML
- VAIDK — Vision AI Dev Kit ⓘ

```
platform = "TensorFlow";
extension = "zip";
```

```
do
{
    var exports = await trainingApi.GetExportsAsync(project.Id, iteration.Id);

    export = exports.FirstOrDefault(x => x.Platform == platform);

    if (export == null)
        export = await trainingApi.ExportIterationAsync(project.Id, iteration.Id, platform);

    Thread.Sleep(1000);
    Console.WriteLine($"\tStatus: {export.Status}");
} while (export.Status == "Exporting");
```

```
if (export.Status == ExportStatus.Done)
{
    Console.WriteLine($"\tDownloading {platform} model");
    var filePath = Path.Combine(Environment.CurrentDirectory, $"{publishedModelName}_{platform}.{extension}");

    using (var httpClient = new HttpClient())
    {
        using (var stream = await httpClient.GetStreamAsync(export.DownloadUri))
        {
            using (var file = new FileStream(filePath, FileMode.Create))
            {
                await stream.CopyToAsync(file);
                Console.WriteLine($"\tModel successfully exported. You can find it here:\n\t{filePath}.");
                WriteSeparator();
            }
        }
    }
}
```

Demo: Creando un detector de objetos con Custom Vision .NET SDK

Información del proyecto

```
----- Selecting existing project: Open Images Detector... -----
       Open Images Detector found in Custom Vision workspace.
       Project domain: General (compact).
       Project type: ObjectDetection.
------------------------------
```

Añadiendo etiquetas al modelo

```
----- Retrieving tags... -----
       Tag Apple was created.
       Tag Muffin was created.
       Tag Orange was created.
       Tag Strawberry was created.
------------------------------
```

Cargando imágenes de entrenamiento

```
----- Accessing images -----
       Adding image 001dd90fc5ae92c9 with its regions.
       Adding image 08d2eed131f4b5d9 with its regions.
       Adding image 096ca56a5c51f6d6 with its regions.
       Adding image 0da61cd490c57814 with its regions.
       Adding image 10628bb9eeafafbb with its regions.
       Adding image 1124df73ba9c7557 with its regions.
       Adding image 123c5bfc13f1c62d with its regions.
       Adding image 12fb96c87a65c542 with its regions.
       Adding image 13628bbfec1fe2c3 with its regions.
```

```
       Uploading images batch #0.
       Uploading images batch #1.
       Uploading images batch #2.
       Uploading images batch #3.
       Uploading images batch #4.
       Uploading images batch #5.
       Uploading images batch #6.
       Uploading images batch #7.
       Uploading images batch #8.
       Uploading images batch #9.
       Uploading images batch #10.
       Uploading last batch.
```

Proyecto open-source disponible en GitHub: https://github.com/icebeam7/ObjectDetectionCustomVisionApp

Entrenando el modelo

```
----- Starting the Training process... -----
       Iteration 'Iteration 1' status: Training
       Iteration 'Iteration 1' status: Training
       Iteration 'Iteration 1' status: Training
       Iteration 'Iteration 1' status: Training
       Iteration 'Iteration 1' status: Training
       Iteration 'Iteration 1' status: Training
       Iteration 'Iteration 1' status: Training
       Iteration 'Iteration 1' status: Training
       Iteration 'Iteration 1' status: Training
       Iteration 'Iteration 1' status: Training
       Iteration 'Iteration 1' status: Training
       Iteration 'Iteration 1' status: Training
       Iteration 'Iteration 1' status: Completed
```

Publicando el modelo

```
----- Starting the Publication process. -----
       Iteration 'Iteration 1' published.
-------------------------------
```

Proyecto open-source disponible en GitHub: https://github.com/icebeam7/ObjectDetectionCustomVisionApp

Evaluando el modelo con imágenes de prueba

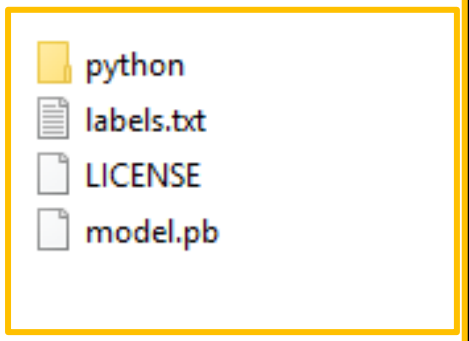
```
----- Making predictions -----
       Image: Test01.jpg
              For Tag 'Muffin': 97.467% [0.12562418, 0.266169071, 0.6825628, 0.6615833]
              For Tag 'Muffin': 4.869% [0.331231415, 0.61405015, 0.6362891, 0.385939837]
              For Tag 'Muffin': 3.135% [0.174345419, 0.409970373, 0.458655715, 0.218689173]
              For Tag 'Muffin': 2.991% [0.008214988, 0.03092061, 0.181516379, 0.257676244]
              For Tag 'Muffin': 2.613% [0.12349081, 0.6451644, 0.681633651, 0.354825616]
              For Tag 'Muffin': 2.177% [0.3288841, 0.47192055, 0.164117157, 0.100035369]
              For Tag 'Muffin': 1.830% [0.1604278, 0, 0.4830678, 0.0725843]
              For Tag 'Muffin': 1.697% [0, 0.005163975, 0.2836524, 0.197787941]
              For Tag 'Muffin': 1.292% [0.452365875, 0.336768657, 0.500448346, 0.5399126]
              For Tag 'Muffin': 1.246% [0.637812257, 0, 0.36217773, 0.06886031]
              For Tag 'Muffin': 1.100% [0.0481789559, 0.474761367, 0.190935969, 0.122284174]
              For Tag 'Strawberry': 1.082% [0, 0.0141699323, 0.07569957, 0.0172690973]
------------------------------
```

```
       Image: Test02.jpg
              For Tag 'Apple': 92.286% [0.66393584, 0.322840065, 0.3145402, 0.551750064]
              For Tag 'Apple': 82.698% [0.215984285, 0.409043372, 0.327929437, 0.421557963]
              For Tag 'Apple': 82.100% [0.43457824, 0.07026026, 0.3172269, 0.502035737]
              For Tag 'Apple': 62.339% [0.0130720735, 0.09492104, 0.176995143, 0.456798434]
              For Tag 'Apple': 10.023% [0.202900678, 0.138997942, 0.542853832, 0.8015692]
              For Tag 'Apple': 4.737% [0.9709588, 0.455433547, 0.0290311575, 0.362548053]
              For Tag 'Apple': 3.711% [0, 0, 0.2653275, 0.656832933]
              For Tag 'Apple': 2.553% [0.352338046, 0, 0.593947768, 0.890323639]
              For Tag 'Apple': 1.455% [0.9459655, 0.293046474, 0.0540244579, 0.49333632]
              For Tag 'Apple': 1.390% [0.120855212, 0.227188244, 0.06773448, 0.303626239]
------------------------------
```

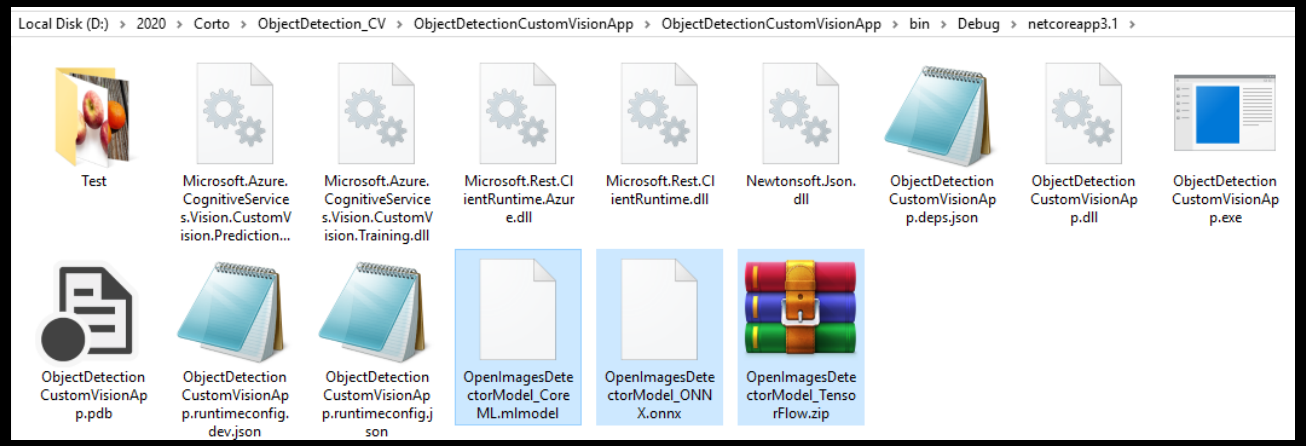
Proyecto open-source disponible en GitHub: https://github.com/icebeam7/ObjectDetectionCustomVisionApp

Exportando el modelo a varias plataformas

```
----- Do you want to export the model? (Y/N) -----
Y       Options:
        1) TensorFlow
        2) CoreML
        3) Other platform
        E) End program
1-----------------------------
        Exporting to TensorFlow...
        Status: Exporting
        Status: Exporting
        Status: Exporting
        Status: Exporting
        Status: Exporting
        Status: Exporting
        Status: Exporting
        Status: Exporting
        Status: Exporting
        Status: Exporting
        Status: Exporting
        Status: Done
Status: Done
        Downloading TensorFlow model
        Model successfully exported. You can find it here:
        D:\2020\Corto\ObjectDetection_CV\ObjectDetectionCustomVisionApp\ObjectDetectionCustomVisionApp\
bin\Debug\netcoreapp3.1\OpenImagesDetectorModel_TensorFlow.zip.
```



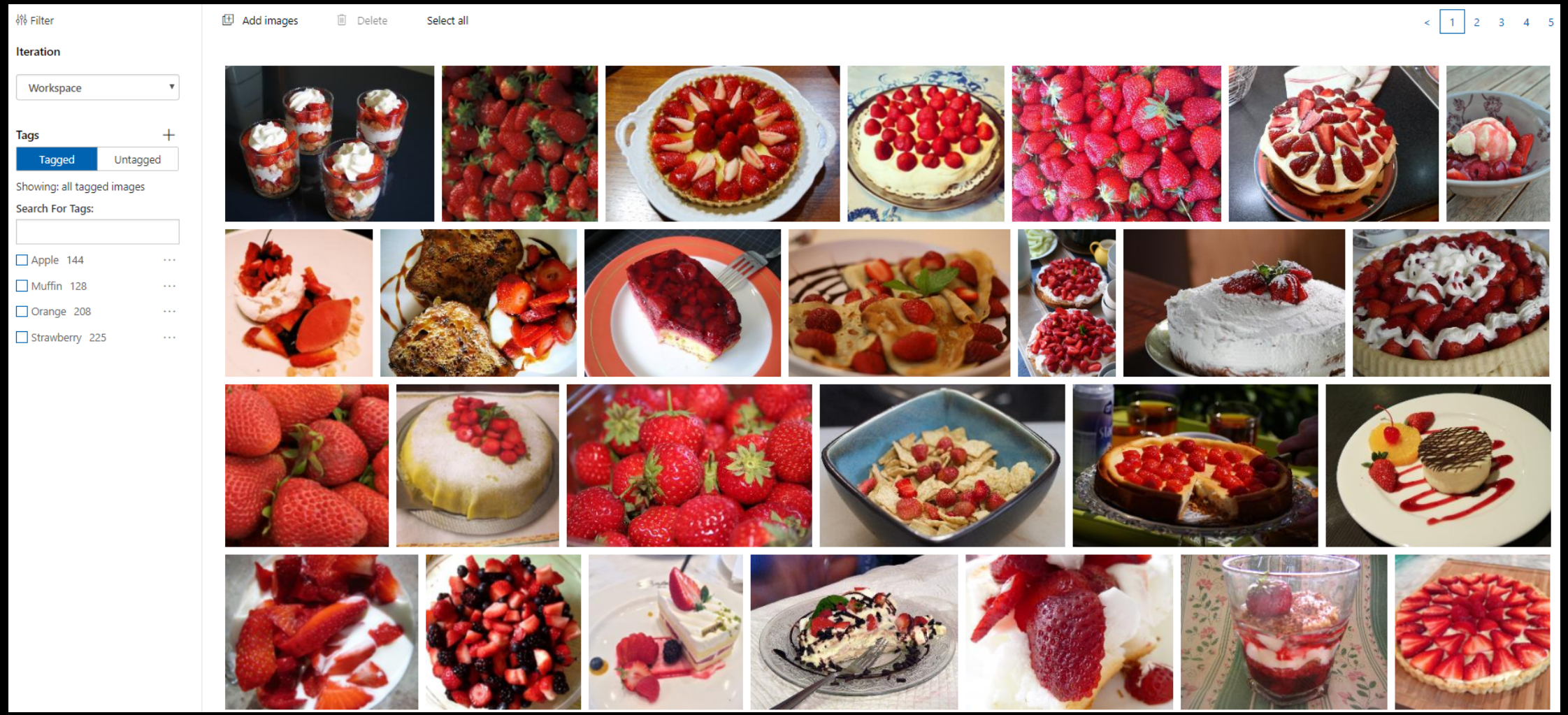

```
        Options:
        1) TensorFlow
        2) CoreML
        3) Other platform
        E) End program
2-----------------------------
        Exporting to CoreML...
        Status: Exporting
        Status: Exporting
        Status: Exporting
        Status: Exporting
        Status: Exporting
        Status: Exporting
        Status: Exporting
        Status: Exporting
        Status: Exporting
        Status: Exporting
        Status: Exporting
        Status: Exporting
        Status: Exporting
        Status: Exporting
        Status: Exporting
        Status: Exporting
        Status: Exporting
        Status: Exporting
        Status: Exporting
        Status: Done
Status: Done
        Downloading CoreML model
        Model successfully exported. You can find it here:
        D:\2020\Corto\ObjectDetection_CV\ObjectDetectionCustomVisionApp\ObjectDetectionCustomVisionApp\
bin\Debug\netcoreapp3.1\OpenImagesDetectorModel_CoreML.mlmodel.
```




```
----- Do you want to export the model? (Y/N) -----
y       Options:
        1) TensorFlow
        2) CoreML
        3) Other platform
        E) End program
3       Type the platform name
ONNX
        Now type the file extension for the ONNX exported model.
onnx
-----------------------------
        Exporting to ONNX...
        Status: Done
Status: Done
        Downloading ONNX model
        Model successfully exported. You can find it here:
        D:\2020\Corto\ObjectDetection_CV\ObjectDetectionCustomVisionApp\ObjectDetectionCustomVisionApp\
bin\Debug\netcoreapp3.1\OpenImagesDetectorModel_ONNX.onnx.
```

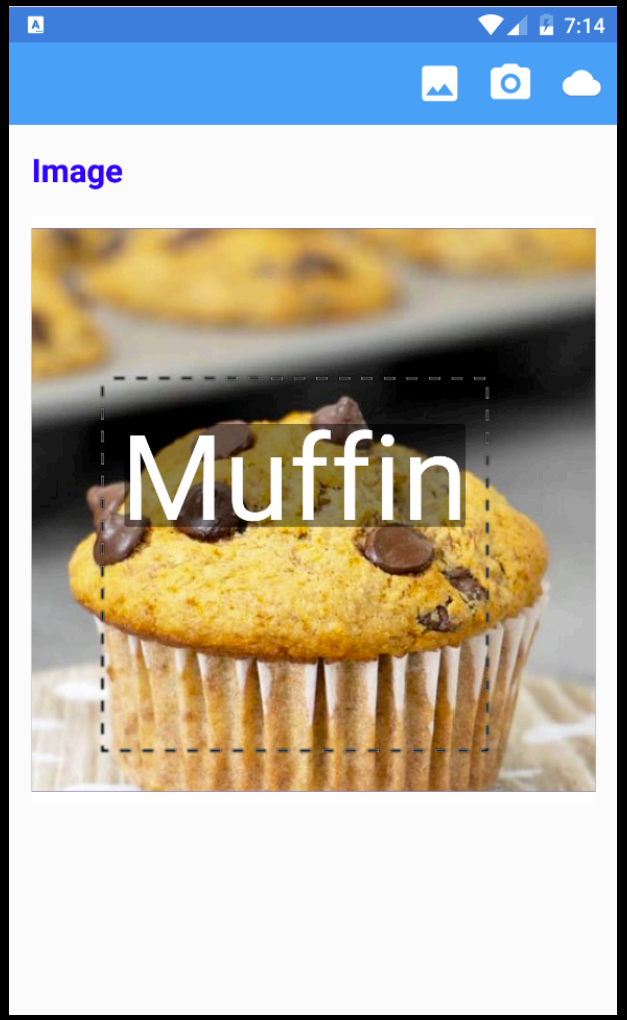
Proyecto open-source disponible en GitHub: https://github.com/icebeam7/ObjectDetectionCustomVisionApp

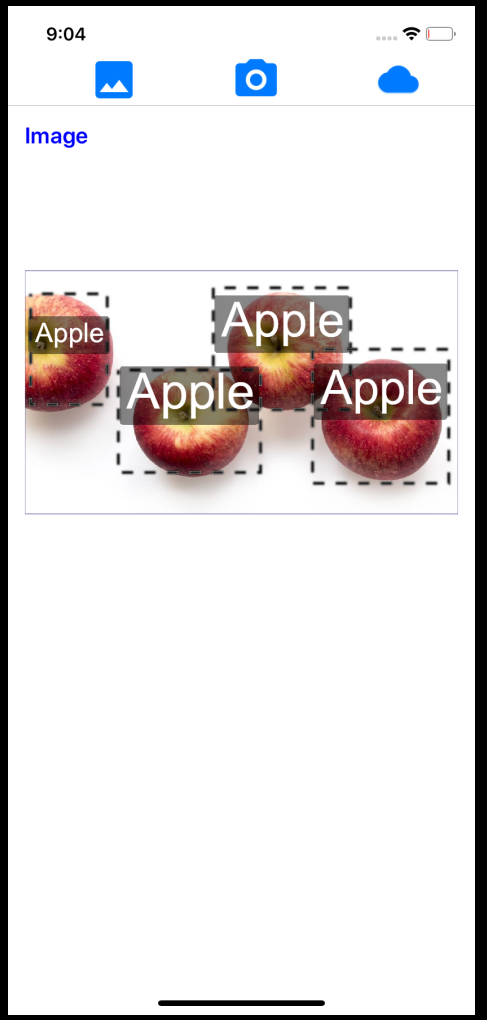
Estado del proyecto en el portal de Custom Vision después de crear el modelo entrenado



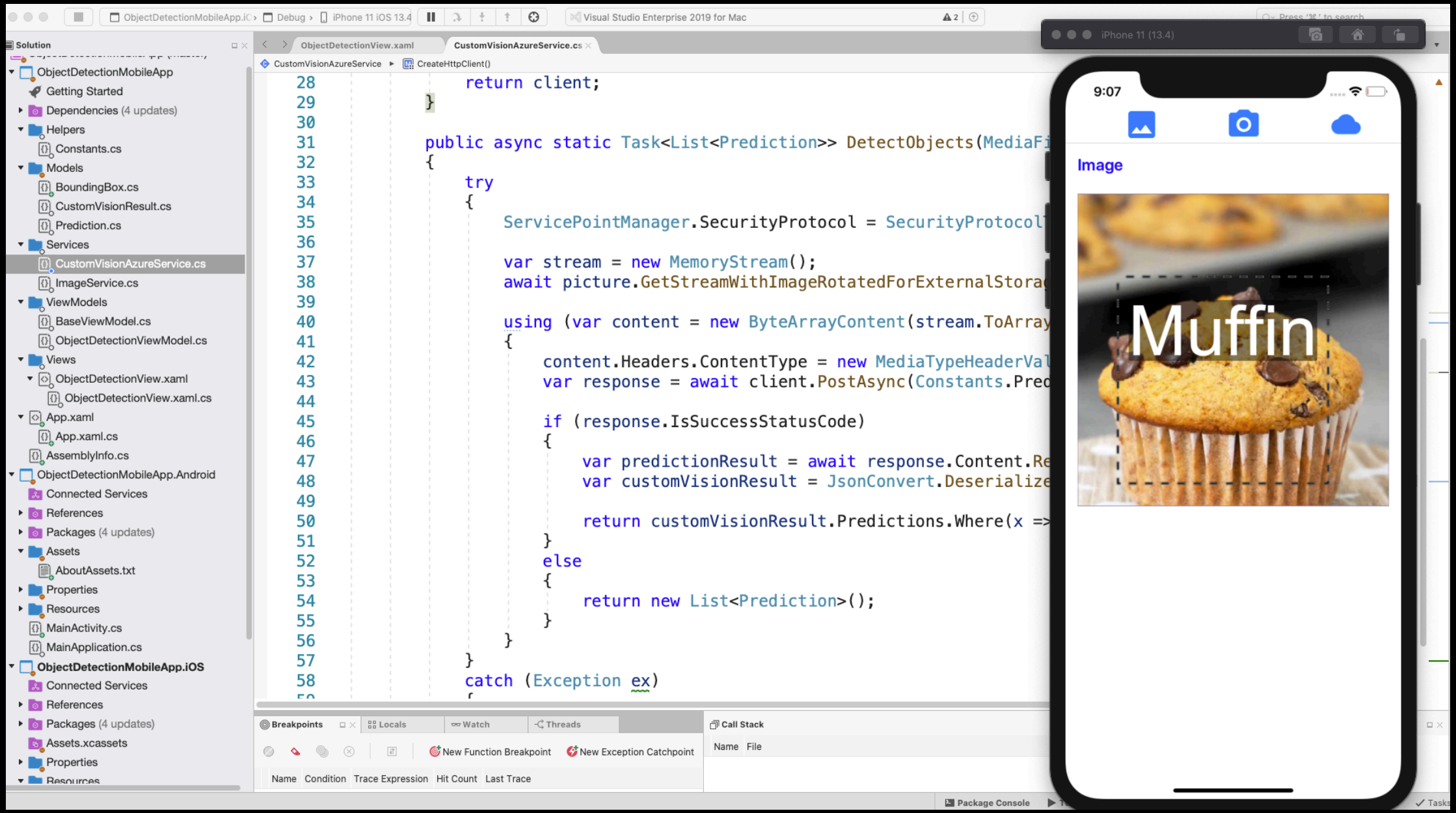
Proyecto open-source disponible en GitHub: https://github.com/icebeam7/ObjectDetectionCustomVisionApp

Demo: Detección de objetos en apps móviles con Custom Vision y Xamarin





Proyecto open-source disponible en GitHub: https://github.com/icebeam7/ObjectDetectionMobileApp



Proyecto open-source disponible en GitHub: https://github.com/icebeam7/ObjectDetectionMobileApp

Tips de optimización

- La mejor forma de tener un detector de objetos de calidad es **agregando imágenes diversas** (diferentes fondos, ángulos, tamaño de objeto, grupos de fotos y variantes de tipos).

- **Incluye imágenes representativas de lo que el detector encontrará en el mundo real.** Por tanto, las fotos en contexto son mejores que las de objetos frente a fondos neutros, por ejemplo.

- Siempre entrena el modelo después de haber agregado más imágenes.

- Usa **al menos 30 imágenes** por etiqueta.

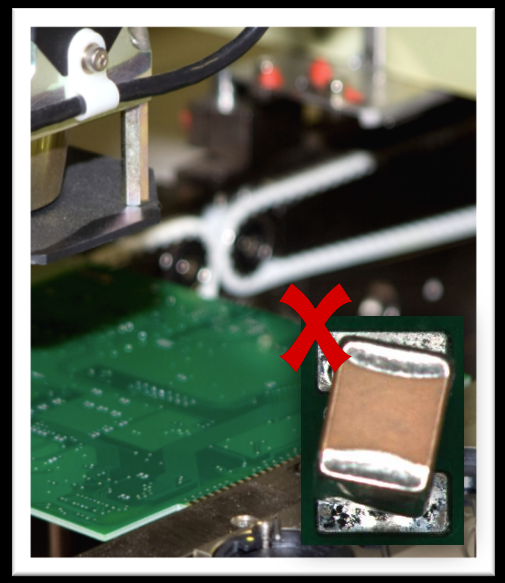
Precio

Pricing details

The below pricing reflects a preview discount.

INSTANCE	TRANSACTIONS PER SECOND (TPS)	FEATURES	PRICE
Free	2 TPS	Upload, training, and prediction transactions Up to 2 projects Up to 1 hour training per month	5,000 training images free per project 10,000 predictions per month
Standard	10 TPS	Upload and prediction transactions Up to 100 projects	$2 per 1,000 transactions
		Training	$20 per compute hour
		Image Storage Up to 6 MB each	$0.70 per 1,000 images

Escenarios de uso



Detección de categoría en una línea de producción

Detección de la salud de plantas en agricultura y ganadería

Detección de productos en una tienda en línea

Escenarios adicionales

- Clasificar imágenes enviadas por el usuario al sitio web
- Identificar elementos: conteo de objetos, identificación de animales, etc.
- Detección de riesgos / seguridad industrial

Call to Action

Servicio de Custom Vision

https://azure.microsoft.com/es-mx/services/cognitive-services/custom-vision-service/

Documentación de Custom Vision

https://docs.microsoft.com/es-mx/azure/cognitive-services/custom-vision-service/home

Quickstart: Create an object detection project with the Custom Vision SDK

https://docs.microsoft.com/en-us/azure/cognitive-services/custom-vision-service/quickstarts/object-detection

Custom Vision y TensorFlow

https://docs.microsoft.com/es-mx/azure/cognitive-services/custom-vision-service/export-model-python

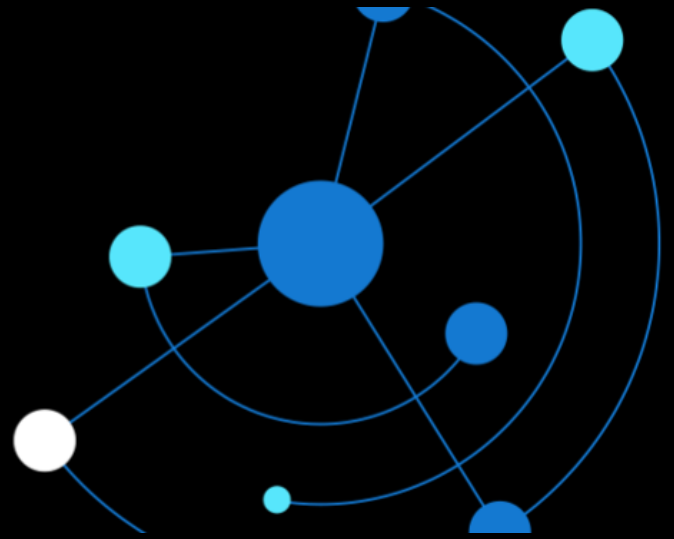

Q
A

Azure Tech Communities Live Days - Video Indexer & Azure Custom Vision

Combinando el poder de Video Indexer y Azure Custom Vision para la detección de objetos en videos con IA

Mayo 8 - 2020 / LATAM -

Ver Agenda



En este evento exploraremos juntos cómo combinar el poder de Video Indexer y Azure Custom Vision para la detección de objetos en videos implementando IA presentado por Luis Antonio Beltran – MVP Mexico (Desarrollador móvil certificado en Xamarin MS (C#, Azure), MCSD en UWP, MCPS, MTA).

Tendrás la oportunidad de preguntar todo lo que necesitas para continuar aprendiendo y elevando tu crecimiento profesional.

El evento comienza a las 02:00 PM EST (Eastern Standard Time). Los horarios locales por país son:
- 02:00 - 03:30 PM: República Dominicana, Puerto Rico, Trinidad y Tobago, Paraguay, Bolivia y Chile.
- 11:00 - 12:30 PM: Costa Rica, Guatemala, El Salvador, Honduras.
- 01:00 - 02:30 PM: Colombia, Jamaica, Ecuador, Perú, Panamá, México.
- 03:00 - 04:30 PM: Argentina, Uruguay.

Agenda

08/05
Azure Tech Communities Live Days

02:00 PM	Combinando el poder de Video Indexer y Azure Custom Vision para la detección de objetos en videos con IA
03:00 PM	Preguntas y respuestas (Q&A)

https://www.microsoft.com/es-xl/events-hub/latam/azure-tech-communities-live-days-video-indexer-amp-azure-custom-vision/

Global AI On Tour Latinoamérica (Online)

May 18th–23rd, 2020 | Worldwide (Online event) | View schedule

Global AI On Tour Latinoamérica es un evento en línea dirigido a entusiastas de la Inteligencia Artificial en Microsoft Azure.

Aprende a implementar soluciones de IA utilizando servicios previamente entrenados, tales como los Cognitive Services y Bot Framework, o construye tus propios modelos de machine learning con Azure ML y frameworks de código abierto como PyTorch y ML.NET. Al final del día, ¡serás capaz de incorporar inteligencia en tus aplicaciones!

https://ti.to/comunidad-xamarin-en-espanol/global-ai-on-tour-latinoamerica-2020

Comunidad Xamarin en Español

Aprendiendo Azure

¡Gracias por tu atención!

GitHub:
https://github.com/icebeam7

LinkedIn:
https://linkedin.com/in/luisantoniobeltran

YouTube:
https://youtube.com/user/darkicebeam

SlideShare:
https://slideshare.net/icebeam

About Me:
https://about.me/luis-beltran

Luis Beltrán
Tomás Bata University in Zlín
Tecnológico Nacional de México en Celaya

Microsoft® MVP Most Valuable Professional

luis@luisbeltran.mx luisbeltran.mx @darkicebeam