

Contents

What is Custom Vision?	2
Create a new Project.....	3
Choose Training Images	6
Upload and Tag Images.....	7
Train the Classifier	13
Evaluate the Classifier	14
Test your Model	16
Improve Model (Optional – Best Practices)	19
Prediction API.....	21
Test Model using API.....	23

Custom Vision

What is Custom Vision?

Azure Custom Vision is an image recognition service that lets you build, deploy, and improve your own image identifier models. An image identifier applies labels (which represent classifications or objects) to images, according to their detected visual characteristics. Unlike the [Computer Vision](#) service, Custom Vision allows you to specify your own labels and train custom models to detect them.

The Custom Vision service uses a machine learning algorithm to analyze images. You, the developer, submit groups of images that feature and lack the characteristics in question. You label the images yourself at the time of submission. Then, the algorithm trains to this data and calculates its own accuracy by testing itself on those same images. Once you've trained the algorithm, you can test, retrain, and eventually use it in your image recognition app to [classify images](#). You can also [export the model](#) itself for offline use.

Custom Vision functionality can be divided into two features. [Image classification](#) applies one or more labels to an image. [Object detection](#) is similar, but it also returns the coordinates in the image where the applied label(s) can be found.

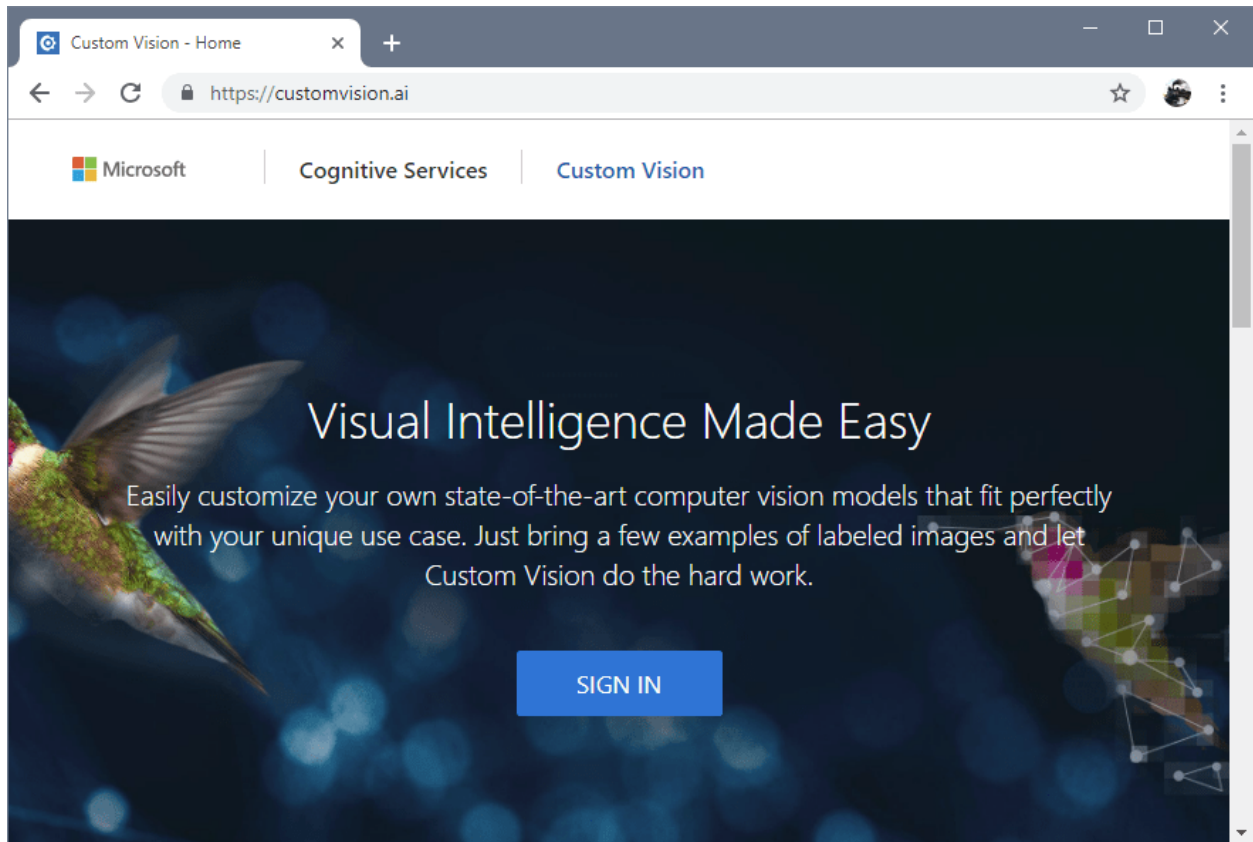
The Custom Vision service is optimized to quickly recognize major differences between images, so you can start prototyping your model with a small amount of data. 50 images per label are generally a good start. However, the service is not optimal for detecting subtle differences in images (for example, detecting minor cracks or dents in quality assurance scenarios).

Additionally, you can choose from several variations of the Custom Vision algorithm that are optimized for images with certain subject material—for example, landmarks or retail items. See [Select a domain](#) for more information.

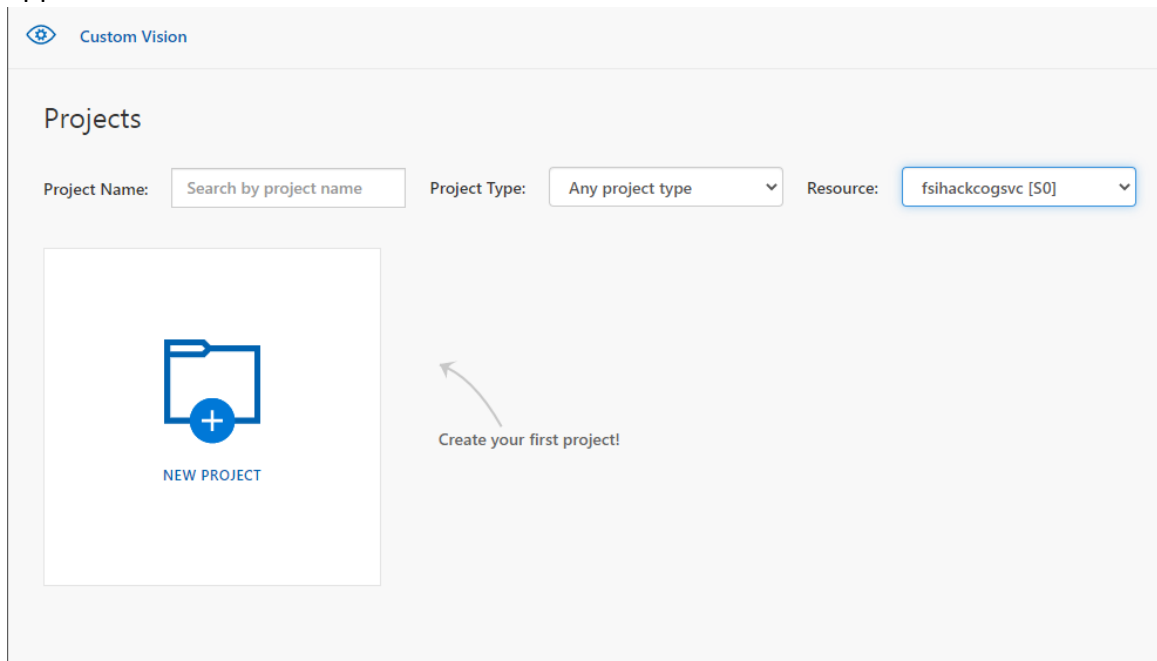
The Custom Vision Service is available as a set of native SDKs as well as through a web-based interface on the [Custom Vision website](#). You can create, test, and train a model through either interface or use both together.

Create a new Project

In your web browser, navigate to the [Custom Vision web page](https://customvision.ai) and select Sign in. Sign in with the same account you used to sign into the Azure portal.



1. To create your first project, select New Project. The Create new project dialog box will appear.

The 'Create new project' dialog box is shown. It has a title bar with a close button. The form includes:

- Name***: A text input field with the placeholder 'Enter project name'.
- Description**: A text input field with the placeholder 'Enter project description'.
- Resource**: A dropdown menu with a 'create new' link to its right.
- Manage Resource Permissions**: A link below the resource dropdown.
- Project Types**: Two radio buttons: 'Classification' (selected) and 'Object Detection'.
- Classification Types**: Two radio buttons: 'Multilabel (Multiple tags per image)' and 'Multiclass (Single tag per image)' (selected).
- Domains**: A list of radio buttons including 'General [A2]' (selected), 'General [A1]', 'General', 'Food', 'Landmarks', 'Retail', 'General (compact) [S1]', 'General (compact)', 'Food (compact)', 'Landmarks (compact)', and 'Retail (compact)'.
- Footer text**: 'Pick the domain closest to your scenario. Compact domains are lightweight models that can be exported to iOS/Android and other platforms. [Learn More](#)'.

2. Enter a name and a description for the project. Then select a Resource Group. If your signed-in account is associated with an Azure account, the Resource Group dropdown will display all of your Azure Resource Groups that include a Custom Vision Service Resource.

Note: If no resource group is available, please confirm that you have logged into customvision.ai with the same account as you used to log into the Azure portal. Also, please confirm you have selected the same "Directory" in the Custom Vision website as the directory in the Azure portal where your Custom Vision resources are located. In both sites, you may select your directory from the drop down account menu at the top right corner of the screen.

3. Select Classification under Project Types. Then, under Classification Types, choose either Multilabel or Multiclass, depending on your use case. Multilabel classification applies any number of your tags to an image (zero or more), while multiclass classification sorts images into single categories (every image you submit will be sorted into the most likely tag). You'll be able to change the classification type later if you want to.
4. Next, select one of the available domains. Each domain optimizes the classifier for specific types of images, as described in the following table. You will be able to change the domain later if you wish.

Domain	Purpose
Generic	Optimized for a broad range of image classification tasks. If none of the other domains are appropriate, or you're unsure of which domain to choose, select the Generic domain.
Food	Optimized for photographs of dishes as you would see them on a restaurant menu. If you want to classify photographs of individual fruits or vegetables, use the Food domain.
Landmarks	Optimized for recognizable landmarks, both natural and artificial. This domain works best when the landmark is clearly visible in the photograph. This domain works even if the landmark is slightly obstructed by people in front of it.
Retail	Optimized for images that are found in a shopping catalog or shopping website. If you want high precision classifying between dresses, pants, and shirts, use this domain.
Compact domains	Optimized for the constraints of real-time classification on mobile devices. The models generated by compact domains can be exported to run locally.

5. Finally, select Create project.

Choose Training Images

As a minimum, we recommend you use at least 30 images per tag in the initial training set. You'll also want to collect a few extra images to test your model once it's trained.

In order to train your model effectively, use images with visual variety. Select images that vary by:

- camera angle
- lighting
- background
- visual style
- individual/grouped subject(s)
- size
- type

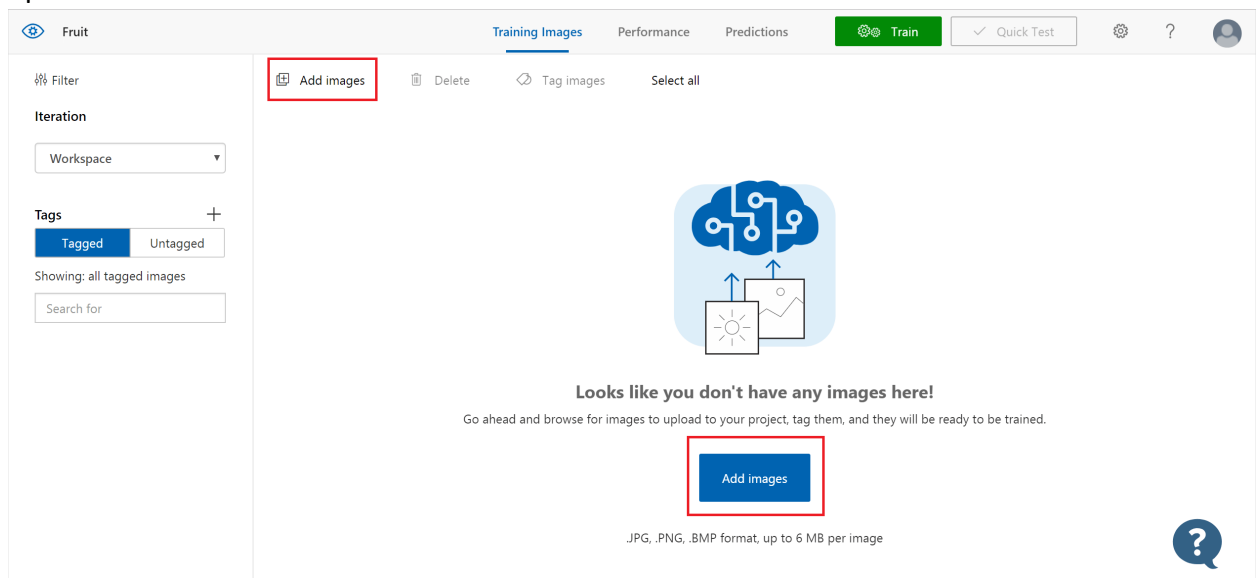
Additionally, make sure all of your training images meet the following criteria:

- .jpg, .png, .bmp, or .gif format
- no greater than 6MB in size (4MB for prediction images)
- no less than 256 pixels on the shortest edge; any images shorter than this will be automatically scaled up by the Custom Vision Service

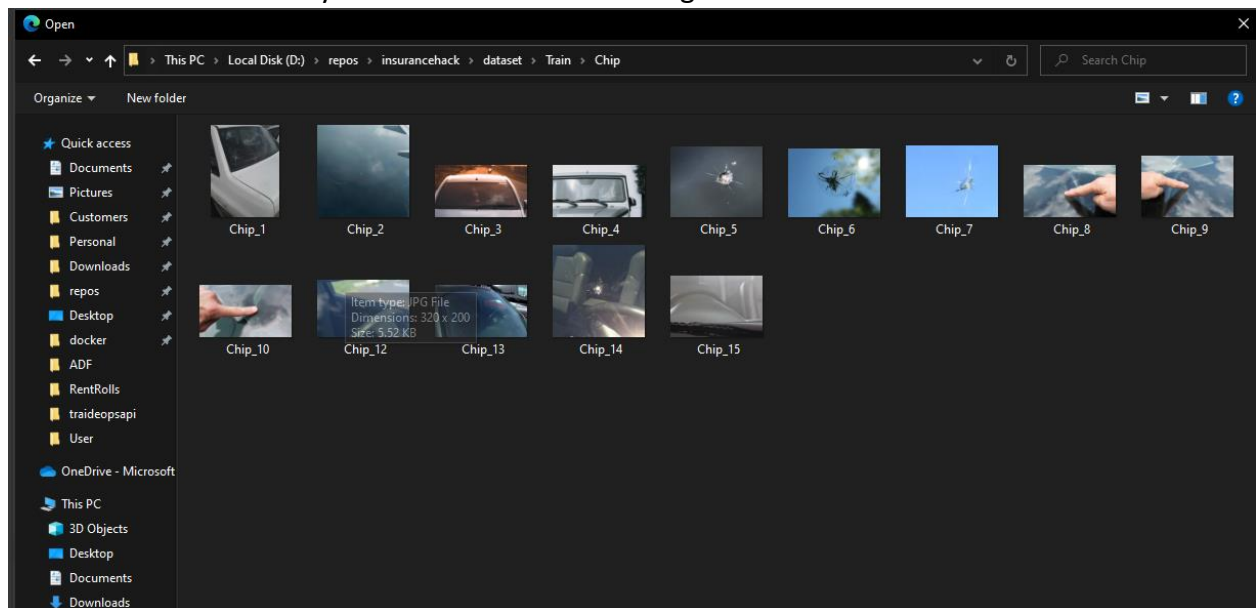
Note : Do you need a broader set of images to complete your training? Trove, a Microsoft Garage project, allows you to collect and purchase sets of images for training purposes. Once you've collected your images, you can download them and then import them into your Custom Vision project in the usual way. Visit the [Trove page](#) to learn more.

Upload and Tag Images

1. To add images, select **Add images** and then select **Browse local files**. Select **Open** to move to tagging. Your tag selection will be applied to the entire group of images you've selected to upload, so it's easier to upload images in separate groups according to their applied tags. You can also change the tags for individual images after they've been uploaded



2. Select the folder where you downloaded the training dataset



3. To create a tag, enter text in the **My Tags** field and press Enter. If the tag already exists, it will appear in a dropdown menu. In a multilabel project, you can add more than one

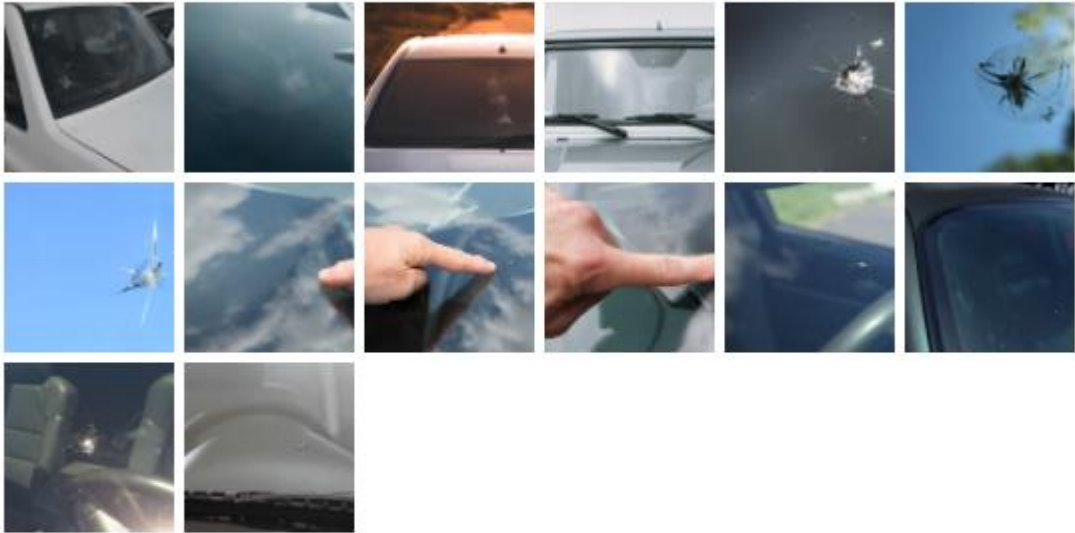
tag to your images, but in a multiclass project you can add only one. To finish uploading the images, use the **Upload [number] files** button

Image upload

Add Tags

Uploading

Summary



14 images will be added...

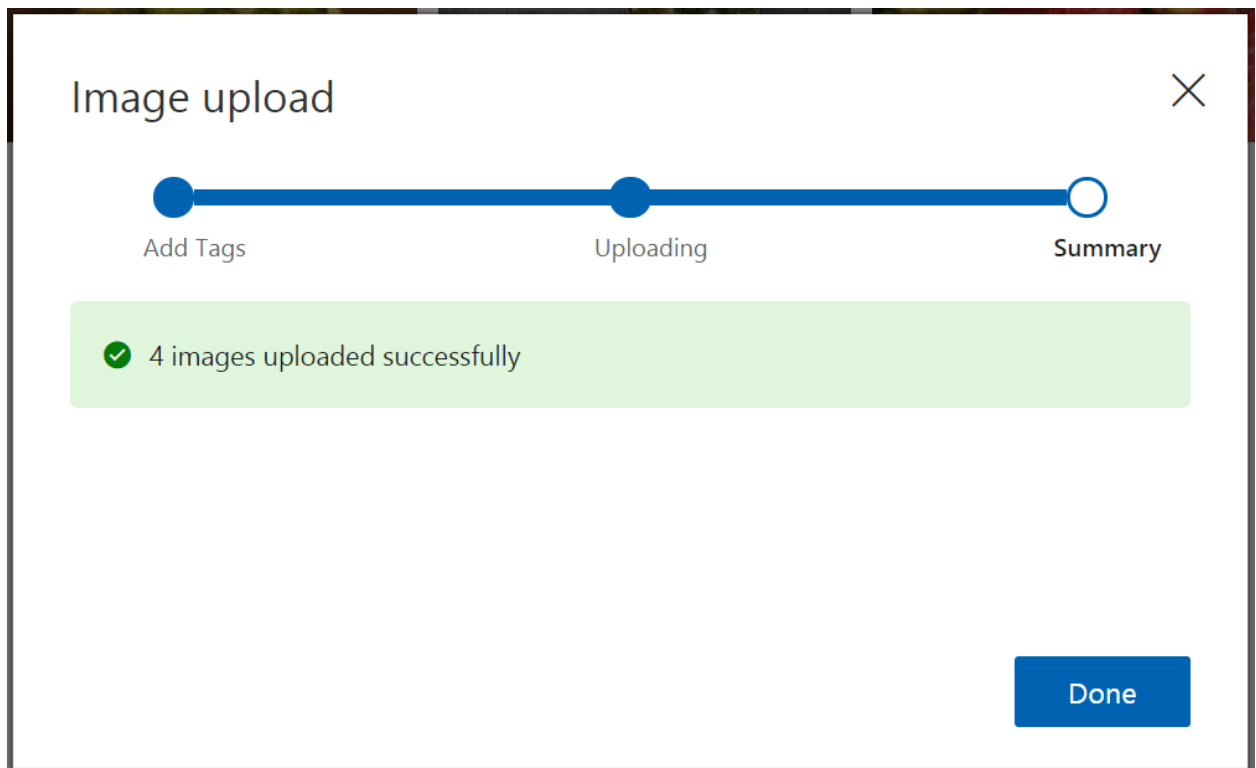
Add some tags to this batch of images...

My Tags

Chip

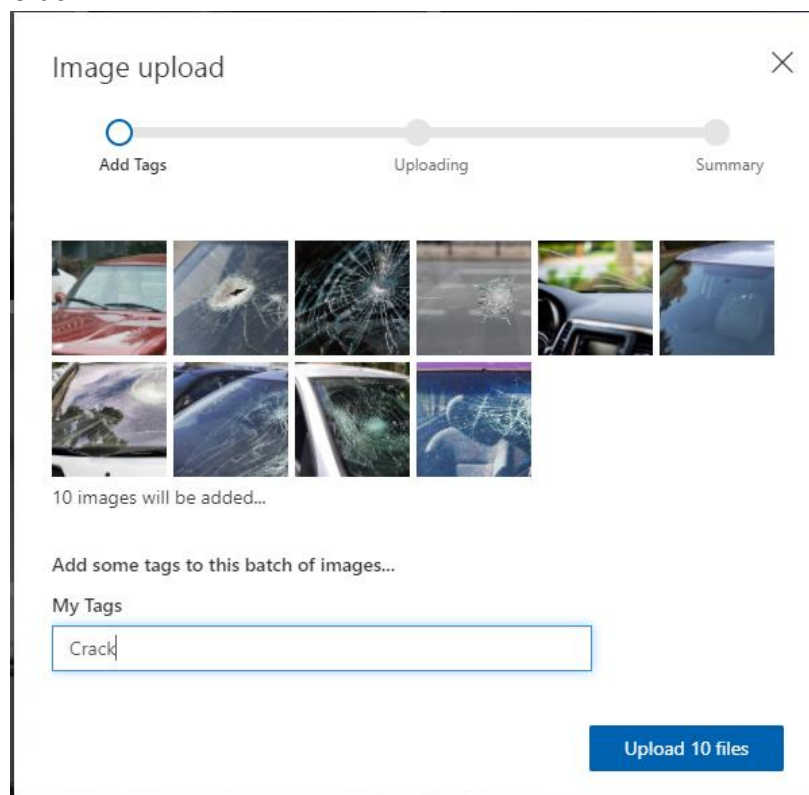
Upload 14 files

4. Select **Done** once the images have been uploaded.



5. Repeat the Steps 1-4 for following tags:

1. Crack



2. Insurance

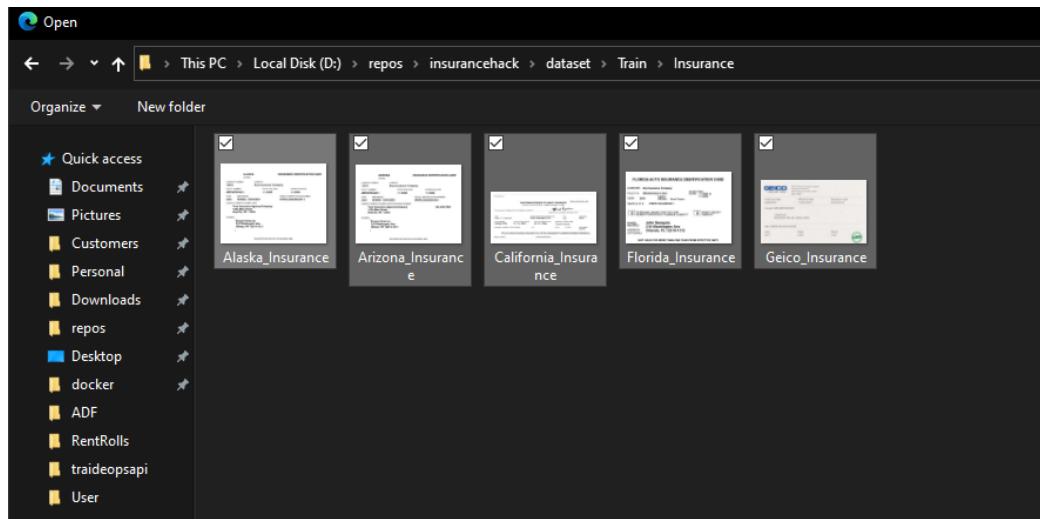


Image upload



Add Tags



Uploading



Summary



5 images will be added...

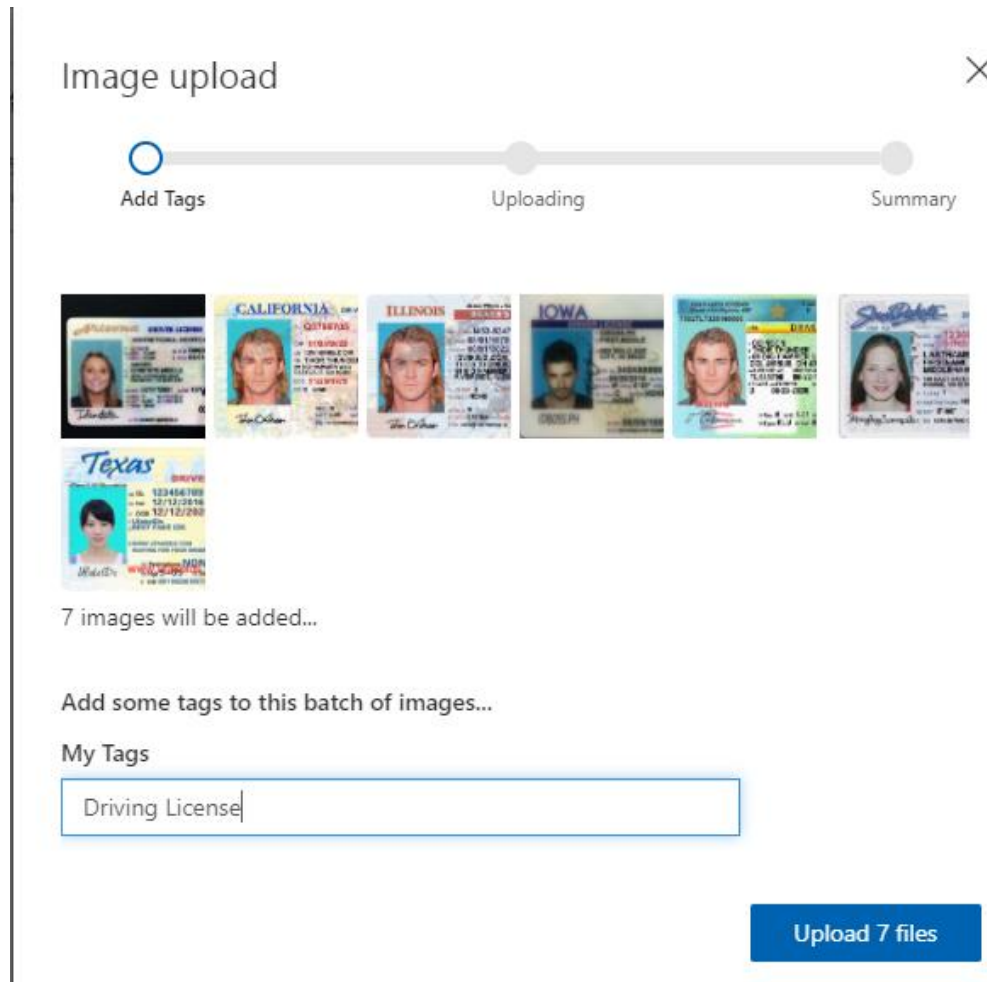
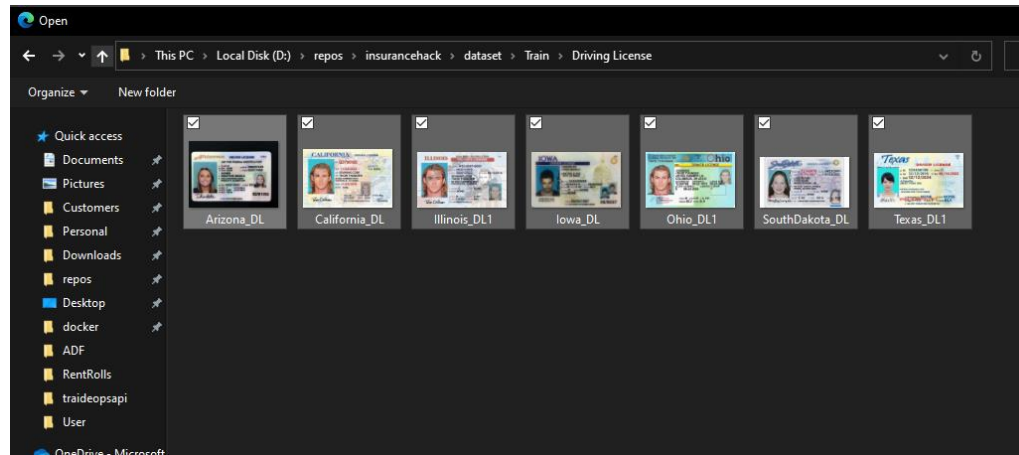
Add some tags to this batch of images...

My Tags

Insurance

Upload 5 files

3. Driving License



4. Service Estimates

Image upload



Add Tags



Uploading



Summary



5 images will be added...

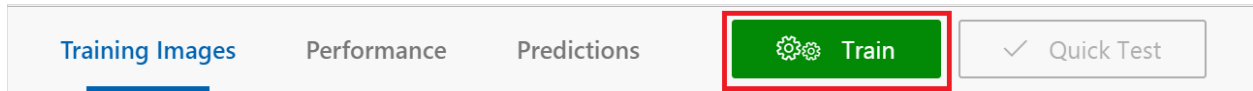
Add some tags to this batch of images...

My Tags

Upload 5 files

Train the Classifier

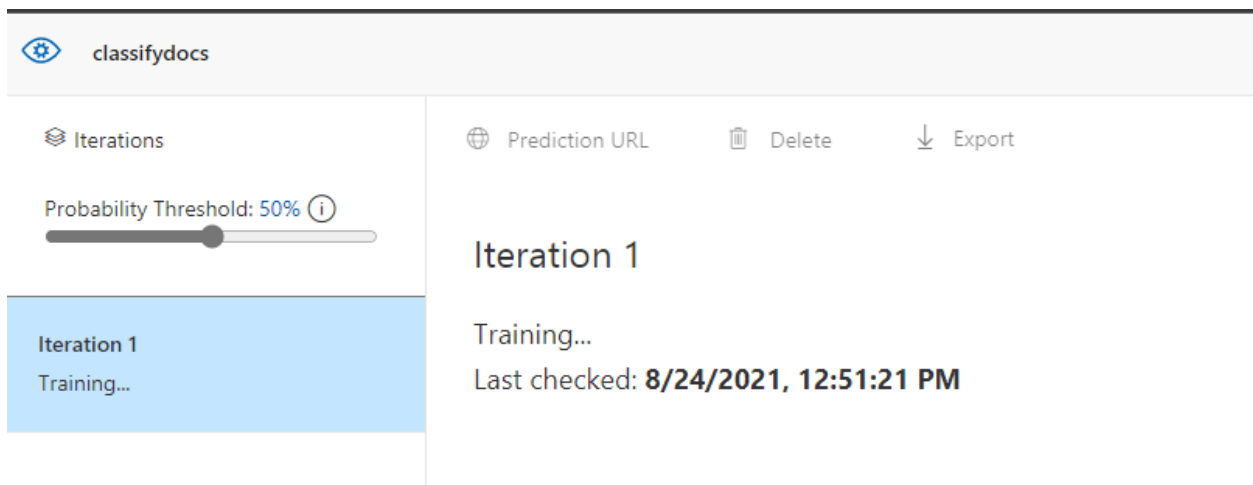
To train the classifier, select the Train button. The classifier uses all of the current images to create a model that identifies the visual qualities of each tag



Choose “Quick Training” in training type option and click Train



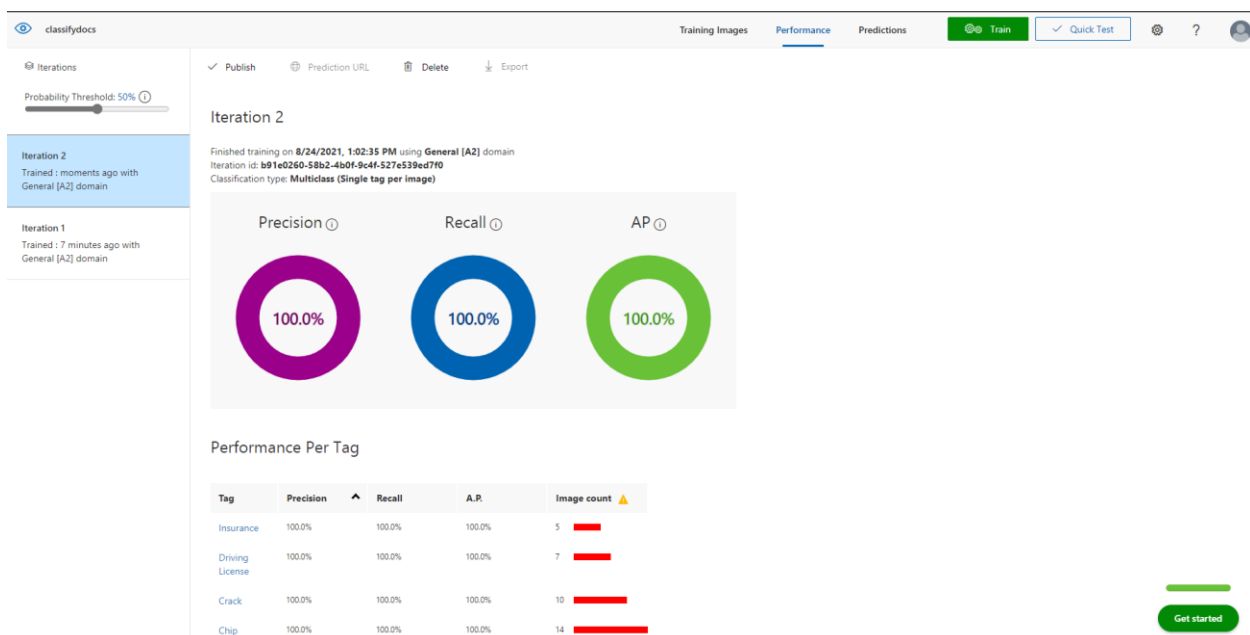
The training process should only take a few minutes. During this time, information about the training process is displayed in the **Performance** tab.



Evaluate the Classifier

After training has completed, the model's performance is estimated and displayed. The Custom Vision Service uses the images that you submitted for training to calculate precision and recall, using a process called [k-fold cross validation](#). Precision and recall are two different measurements of the effectiveness of a classifier:

- 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 actually 100 images of apples, and the model identified 80 as apples, the recall would be 80%.



Note the Probability Threshold slider on the left pane of the Performance tab. This is the level of confidence that a prediction needs to have in order to be considered correct (for the purposes of calculating precision and recall).

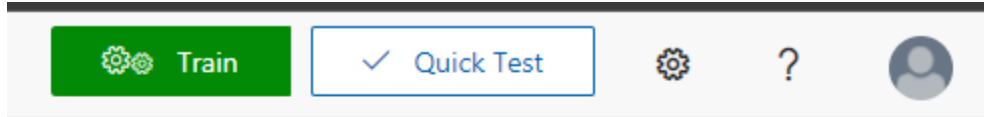
When you interpret prediction calls with a high probability threshold, they tend to return results with high precision at the expense of recall—the detected classifications are correct, but many remain undetected. A low probability threshold does the opposite—most of the actual classifications are detected, 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.

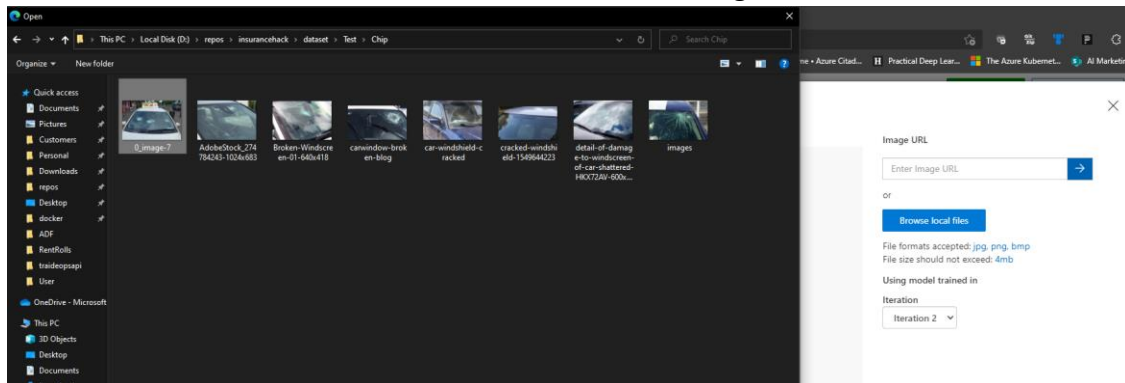
Each time you train your classifier, you create a new *iteration* with its own updated performance metrics. You can view all of your iterations in the left pane of the Performance tab. You'll also find the Delete button, which you can use to delete an iteration if it's obsolete. When you delete an iteration, you delete any images that are uniquely associated with it.

Test your Model

1. From the Custom Vision web page, select your project. Select **Quick Test** on the right of the top menu bar. This action opens a window labeled Quick Test.



2. In the **Quick Test** window, click in the **Submit Image** field and enter the URL of the image you want to use for your test. If you want to use a locally stored image instead, click the **Browse local files** button and select a local image file



Quick Test

Image URL

Enter Image URL

or

Browse local files

File formats accepted: jpg, png, bmp
File size should not exceed: 4mb

Using model trained in

Iteration

Iteration 2

Predictions

Tag	Probability
Crack	60.3%
Chip	35.8%
Driving License	3.3%
Insurance	0.4%

Quick Test

×

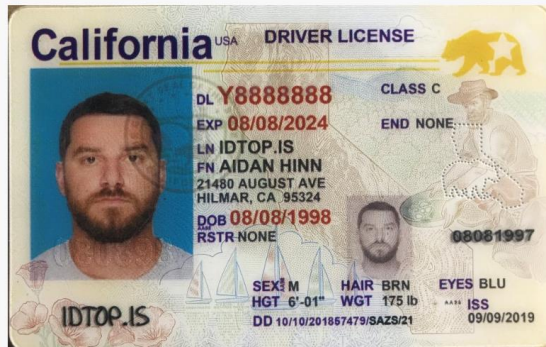


Image URL

Enter Image URL



or

Browse local files

File formats accepted: jpg, png, bmp
File size should not exceed: 4mb

Using model trained in

Iteration

Iteration 2

Predictions

Tag	Probability
Driving License	97.1%
Insurance	2.5%
Crack	0.2%
Chip	0%

Quick Test

×

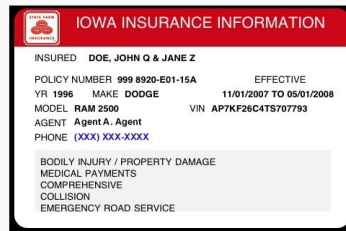


Image URL

Enter Image URL



or

Browse local files

File formats accepted: jpg, png, bmp
File size should not exceed: 4mb

Using model trained in

Iteration

Iteration 2

Predictions

Tag	Probability
Insurance	92.7%
Driving License	6.5%
Crack	0.4%
Chip	0.2%

- The image you select appears in the middle of the page. Then the results appear below the image in the form of a table with two columns, labeled **Tags** and **Confidence**. After you view the results, you may close the **Quick Test** window.
- You can now add this test image to your model and then retrain your model.

To use the image submitted previously for training, use the following steps:

- To view images submitted to the classifier, open the Custom Vision web page and select the **Predictions** tab.

Tip : The default view shows images from the current iteration. You can use the **Iteration** drop down field to view images submitted during previous iterations.

2. Hover over an image to see the tags that were predicted by the classifier.

Tip : Images are ranked, so that the images that can bring the most gains to the classifier are at the top. To select a different sorting, use the **Sort** section.

3. To add an image to your training data, select the image, select the tag, and then select **Save and close**. The image is removed from **Predictions** and added to the training images. You can view it by selecting the **Training Images** tab.
4. Use the Train button to retrain the classifier.

Improve Model (Optional – Best Practices)

The quality of your classifier or object detector depends on the amount, quality, and variety of the labeled data you provide it and how balanced the overall dataset is. A good model has a balanced training dataset that is representative of what will be submitted to it. The process of building such a model is iterative; it's common to take a few rounds of training to reach expected results.

The following is a general pattern to help you train a more accurate model:

1. First-round training
2. Add more images and balance data; retrain
3. Add images with varying background, lighting, object size, camera angle, and style; retrain
4. Use new image(s) to test prediction
5. Modify existing training data according to prediction results

Prevent Overfitting

Sometimes, a model will learn to make predictions based on arbitrary characteristics that your images have in common. For example, if you are creating a classifier for apples vs. citrus, and you've used images of apples in hands and of citrus on white plates, the classifier may give undue importance to hands vs. plates, rather than apples vs. citrus.

To correct this problem, provide images with different angles, backgrounds, object size, groups, and other variations. The following sections expand upon these concepts.

Data Quantity

The number of training images is the most important factor for your dataset. We recommend using at least 50 images per label as a starting point. With fewer images, there's a higher risk of overfitting, and while your performance numbers may suggest good quality, your model may struggle with real-world data.

Data Balance

It's also important to consider the relative quantities of your training data. For instance, using 500 images for one label and 50 images for another label makes for an imbalanced training dataset. This will cause the model to be more accurate in predicting one label than another. You're likely to see better results if you maintain at least a 1:2 ratio between the label with the fewest images and the label with the most images. For example, if the label with the most images has 500 images, the label with the least images should have at least 250 images for training.

Data Variety


Be sure to use images that are representative of what will be submitted to the classifier during normal use. Otherwise, your model could learn to make predictions based on arbitrary characteristics that your images have in common. For example, if you are creating a classifier for apples vs. citrus, and you've used images of apples in hands and of citrus on white plates, the classifier may give undue importance to hands vs. plates, rather than apples vs. citrus.


To correct this problem, include a variety of images to ensure that your model can generalize well. Below are some ways you can make your training set more diverse:


- **Background:** Provide images of your object in front of different backgrounds. Photos in natural contexts are better than photos in front of neutral backgrounds as they provide more information for the classifier.
- **Lighting:** Provide images with varied lighting (that is, taken with flash, high exposure, and so on), especially if the images used for prediction have different lighting. It is also helpful to use images with varying saturation, hue, and brightness.
- **Object Size:** Provide images in which the objects vary in size and number (for example, a photo of bunches of bananas and a closeup of a single banana). Different sizing helps the classifier generalize better.
- **Camera Angle:** Provide images taken with different camera angles. Alternatively, if all of your photos must be taken with fixed cameras (such as surveillance cameras), be sure to assign a different label to every regularly-occurring object to avoid overfitting—interpreting unrelated objects (such as lampposts) as the key feature.
- **Style:** Provide images of different styles of the same class (for example, different varieties of the same fruit). However, if you have objects of drastically different styles (such as Mickey Mouse vs. a real-life mouse), we recommend you label them as separate classes to better represent their distinct features.

Prediction API

From the Custom Vision web page, select your project and then select the Performance tab. To submit images to the Prediction API, you will first need to publish your iteration for prediction, which can be done by selecting Publish and specifying a name for the published iteration. This will make your model accessible to the Prediction API of your Custom Vision Azure resource.


 classifydocs

 Iterations

Probability Threshold: 50% 


Iteration 2

Trained : moments ago with General [A2] domain

 Publish

Publish this iteration so that it is accessible from the Prediction API.

Delete

 Export

Iteration 2

Finished training on **8/24/2021, 1:02:35 PM** using **General [A2]** domain
Iteration id: **b91e0260-58b2-4b0f-9c4f-527e539ed7f0**
Classification type: **Multiclass (Single tag per image)**

Publish Model

We only support publishing to a prediction resource in the same region as the training resource the project resides in.

Please check if you have a prediction resource and if the prediction resource is in the same region as the training resource.

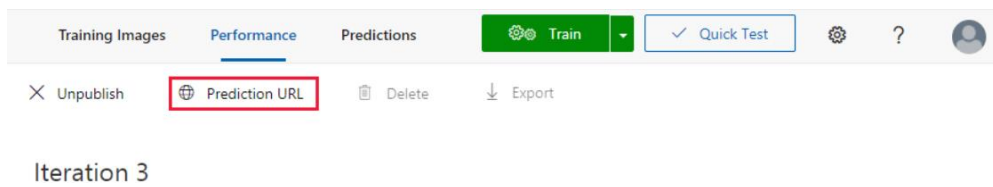
Model name

Prediction resource

Once your model has been successfully published, you'll see a "Published" label appear next to your iteration in the left-hand sidebar, and its name will appear in the description of the iteration.

Once your model has been published, you can retrieve the required information by selecting **Prediction URL**. This will open up a dialog with information for using the Prediction API, including the **Prediction URL** and **Prediction-Key**.

Note : Make sure the model name is latest



How to use the Prediction API ✕

If you have an image URL:

```
https://southcentralus.api.cognitive.microsoft.com/customvision/v3.0/Prediction/70:
```

Set **Prediction-Key** Header to : **<Your-Prediction-Key>**
Set **Content-Type** Header to : **application/json**
Set Body to : **{"Url": "https://example.com/image.png"}**

If you have an image file:

```
https://southcentralus.api.cognitive.microsoft.com/customvision/v3.0/Prediction/70:
```

Set **Prediction-Key** Header to : **<Your-Prediction-Key>**
Set **Content-Type** Header to : **application/octet-stream**
Set Body to : **<image file>**

Got it!

We will use the Prediction API (image URL) and the prediction key. **Record the ProjectId, URL and the Prediction Key as they will be used when in the logicapp workflow**

Test Model using API

- Go to [Online API Testing Tool | Test Your API Online \(reqbin.com\)](https://reqbin.com)
- Select REST API POST Example
- Enter the URL that you copied from Predict popup window above. Ensure that the HTTP method is POST
- Go to Content and copy following JSON
- {"Url": "https://www.dol.wa.gov/driverslicense/images/DLsample-New-CDL-Standard-2018.png"}
- Go to Headers Tab and enter
- Content-Type: application/json
- Prediction-Key: **<replace yourkey>**

How do I post JSON to a REST API end

To post JSON to a REST API endpoint, you must send an HTTP POST request to the You also need to specify the data type in the body of the POST message using the example, we also send the Accept: application/json request header to tell the REST

The screenshot shows the reqbin.com API testing tool interface. At the top, there are tabs for 'File', 'Generate Code', and 'Tools'. Below these, there is a URL input field containing 'https://fsihackcogsvcpred.cognitiv', a dropdown menu for the HTTP method set to 'POST', and a dropdown menu for the content type set to 'US'. A blue 'Send' button is to the right of the content type dropdown. Below the URL and method fields, there are tabs for 'Authorization', 'Content (1)', 'Headers (3)', and 'Raw (7)'. The 'Content (1)' tab is selected, showing a text area with the JSON string '{\"/driverslicense/images/DLsample-New-CDL-Standard-2018.png\"}'.

- Hit Send and you should see the response with Prediction and probabilities

[Share](#) [</> Generate Code](#) [↗ Debug API](#)

Status: **200 (OK)** Time: **3600 ms** Size: **0.67 kb**

Content (27)

Headers (9)

Raw (11)

JSON

Timings

```
{
  "id": "72870d63-d292-485b-bc90-bc560807939c",
  "project": "00efc47c-9c30-4d92-a21d-2caac17b8eba",
  "iteration": "dbd9eb92-3951-4dec-ad0d-78113d7a1a79",
  "created": "2021-10-30T16:38:24.479Z",
  "predictions": [{
    "probability": 0.8992402,
    "tagId": "a7946180-95c5-4516-b1cb-f86544469c06",
    "tagName": "Driving License"
  }, {
    "probability": 0.09991067,
    "tagId": "73017ffd-8ec8-4e4f-876f-0c50cf73b018",
```