IBM Watson Solutions

Train Custom Images using Visual Recognition Service

Prepared by Armen Pischdotchian

Version 7.0 June 2016

Overview

What is Bluemix you ask? **Bluemix** is an implementation of IBM's Open Cloud Architecture, leveraging Cloud Foundry to enable developers to rapidly build, deploy, and manage their cloud applications, while tapping a growing ecosystem of available services and runtime frameworks. You can view a short introductory video here: **http://www.ibm.com/developerworks/cloud/library/cl-bluemix-dbarnes-ny/index.html**

The purpose of this workshop is not to introduce you to Bluemix, that foundational knowledge is a prerequisite study on your part and you can obtain it from the link mentioned above. This workshop guides you through how to classify, detect faces in, and recognize text in an image, and how to train and create a custom classifier using the Visual Recognition service on Bluemix. The following workshop is literally a copy/paste from the DOCS sections of the Visual Recognition service, slightly better organized with a nice twist at the end.

https://www.ibm.com/smarterplanet/us/en/ibmwatson/developercloud/doc/visual-recognition/index.shtml

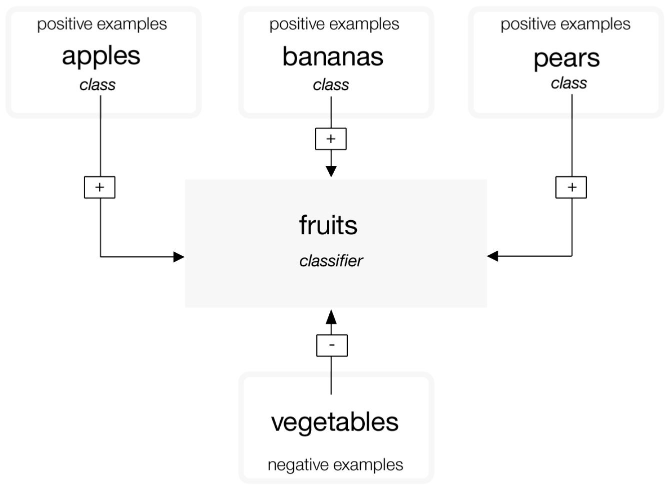
Using your own data

After you classify an image and create, train, and query a custom classifier with the example data in this workshop, you are guided to classify your own data or create your own custom classifier. This is task 7 and you complete this task at the end of the workshop.

Structure of the training data

A custom classifier is a group of classes that are trained against each other. This allows you to create a multi-faceted classifier that can identify highly specialized subjects, while also providing a score for each individual class.

During classifier training, classes are created when you upload separate compressed (.zip) files of positive examples for each class. For example, to create a classifier called **fruit**, you might upload a .zip file of images of pears, a .zip file of images of apples, and a .zip file of images of bananas in a single training call. You also provide a .zip file of negative examples in the same training call to further hone your classifier. Negative example files are not used to create a class. For the custom classifier fruit, for example, you might provide a .zip file with images of various vegetables.



After training completes, when the service identifies fruit in an image, it will return the classifier fruit as an array containing the classes pears, apples, bananas with their respective confidence scores.

The `POST /v3/classifiers` training call requires that you provide at least two example .zip files: two positive examples files, or one positive examples file and one negative examples file. Because custom classifiers are trained with only the images provided in a single call, you cannot update an existing classifier; you can only create new classifiers or delete existing classifiers.

Custom classifiers are only accessible to the specific service instance where they were created and cannot be shared with other Bluemix® users who do not have access to your instance of the service.

Size limitations

There are size limitations for training calls and data:

- The service accepts a maximum of 10,000 images or 100 MB per .zip file
- The service requires a minimum of 10 images per .zip file.
- The service accepts a maximum of 256 MB per training call.

There are also size limitations for classification calls:

- The `POST /v3/classify` methods accept a maximum of 20 images per batch.
- The `POST /v3/detect_faces` methods accept a maximum of 15 images per batch.
- The `POST /v3/recognize_text` methods accept a maximum of 10 images per batch.

Guidelines for good training

The following guidelines are not enforced by the API; however, the service tends to perform better when the training data adheres to them:

- A minimum of 50 images is recommended in each .zip file, as fewer than 50 images can decrease the quality of the trained classifier.
- If the quality and content of training data is the same, then classifiers that are trained on more images will generally be more accurate than classifiers that are trained on fewer images. The benefits of training a classifier on more images plateaus at around 5000 images, and this can take a while to process. You can train a classifier on more than 5000 images, but it may not significantly increase that classifier's accuracy.
- Uploading a total of 150-200 images per .zip file gives you the best balance between the time it takes to train and the improvement to classifier accuracy. More than 200 images increase the time, and it does increase the accuracy, but with diminishing returns for the amount of time it takes.
- Include approximately the same number of images in each examples file. Including an unequal number of images can cause the quality of the trained classifier to decline.
- The accuracy of your custom classifier can be affected by the kinds of images you provide to train it. Provide example images that are similar to the images you plan to analyze. For example, if you are training the classifier "tiger", your classifier might be less accurate if you provide only images of tigers in a zoo taken by a mobile phone to train the classifier, but you want to test the classifier on images of tigers in the wild taken by professional photographers.

Guidelines for high volume classifying

If you want to classify many images, submitting one image at a time can take a long time. You can maximize efficiency and performance of the service in the following ways:

- Resize images to be no larger than 320 pixels in either width or height. Images do not need to be high resolution.
- Submit images in batches as compressed (.zip) files.
- Specify only the classifiers you want results for in the `classifier_ids` parameter. If you do not specify a value for this parameter, the service classifies the images against the default classifier and takes longer to return a response.

Tasks that you complete in this workshop

You complete the following tasks in this workshop. This workshop is an adaptation of the tutorial found in the Visual Recognition service in IBM Bluemix.

1. Getting service credentials
2. Classifying an image
3. Detecting faces in an image
4. Recognizing text in an image
5. Creating a custom classifier
6. Classifying an image with a custom classifier
7. Training your images

Before you begin, you must already have created a Bluemix account.

Task 1: Obtaining your service credentials

Before you can work with a Watson Service, you need service credentials in Bluemix. If you already have credentials for this service, you can skip this Task.

To get your service credentials, follow these steps:

1. Log in to Bluemix at console.ng.bluemix.net.
2. Create an instance of the **Visual Recognition** service:
    1. Click the Catalog tab
    2. Scroll down until you see the Watson services
    3. Click the Visual Recognition tile (service)
    4. Accept all default values
    5. Once the service has completed loading, click CREATE.
    6. On the left side of the page, click **Service Credentials** to view your service credentials.
    7. Copy the `api-key` from these service credentials to a text editor.
    8. Keep that text editor handy and easy to access. You will use that as your sandbox to copy/paste code and edit it there before running the commands in a terminal.

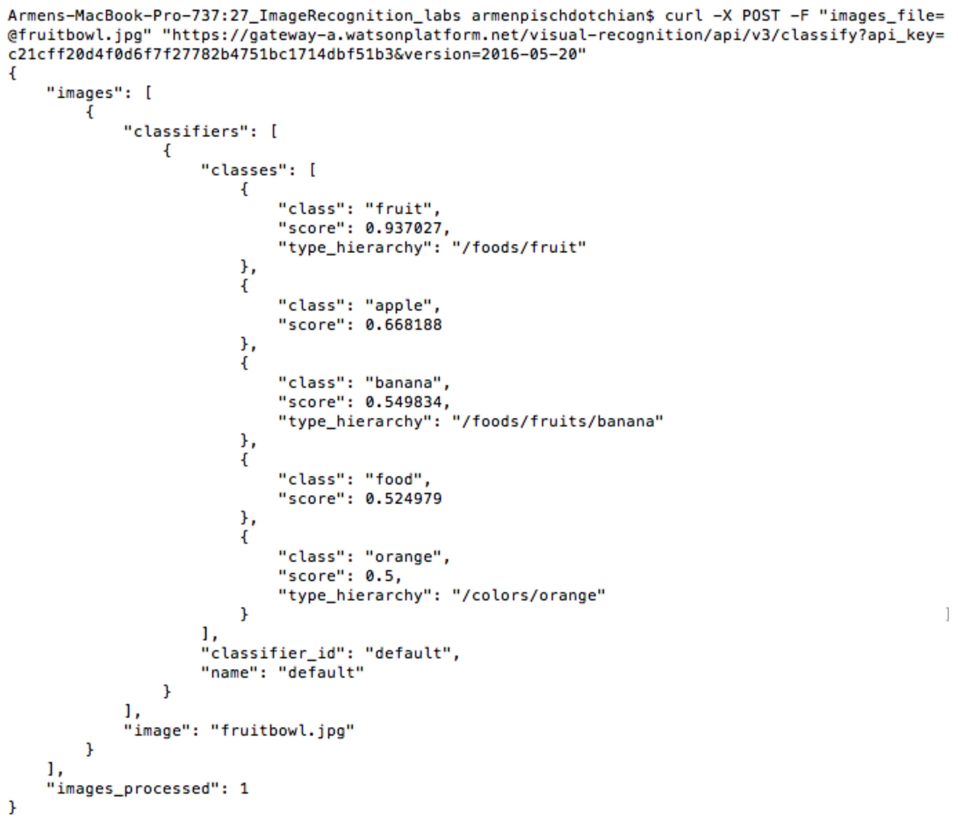
Task 2: Classifying an image

The Visual Recognition service comes with a powerful set of built-in classifiers that you can use to analyze an image quickly and with minimal effort. You can submit an image URL, a webpage URL to extract the main image from, or upload an image file directly. Maximum image size is 2 MB. To classify an uploaded image against the default classes:

1. Use the `POST /v3/classify` method to submit an image to be analyzed. Replace <api-key> with the credentials you copied from the service onto your text pad. The following example uploads the image fruitbowl.jpg and classifies it against all built-in classifiers.

   ```
   curl -X POST -F "images_file=@fruitbowl.jpg" "https://gateway-
   a.watsonplatform.net/visual-recognition/api/v3/classify?api_key=<api-
   key>&version=2016-05-20"
   ```

   The API returns a response that includes the default classifier, the classes identified in the image, and a score for each class. Scores range from 0 to 1, with a higher score indicating greater correlation. The `/v3/classify` calls omit low-scoring classes by default. You can show low-scoring classes by setting the `show_low_confidence` parameter to `1` in your call.

   ```
   Armens-MacBook-Pro-737:27_ImageRecognition_labs armenpischdotchian$ curl -X POST -F "images_file=
   @fruitbowl.jpg" "https://gateway-a.watsonplatform.net/visual-recognition/api/v3/classify?api_key=
   c21cff20d4f0d6f7f27782b4751bc1714dbf51b3&version=2016-05-20"
   {
       "images": [
           {
               "classifiers": [
                   {
                       "classes": [
                           {
                               "class": "fruit",
                               "score": 0.937027,
                               "type_hierarchy": "/foods/fruit"
                           },
                           {
                               "class": "apple",
                               "score": 0.668188
                           },
                           {
                               "class": "banana",
                               "score": 0.549834,
                               "type_hierarchy": "/foods/fruits/banana"
                           },
                           {
                               "class": "food",
                               "score": 0.524979
                           },
                           {
                               "class": "orange",
                               "score": 0.5,
                               "type_hierarchy": "/colors/orange"
                           }
                       ],
                       "classifier_id": "default",
                       "name": "default"
                   }
               ],
               "image": "fruitbowl.jpg"
           }
       ],
       "images_processed": 1
   }
   ```

2. Review your results.

You classified your first image! Move on to Task 3 and 4 to learn how to analyze images for faces and text, or jump to Task 5 to create a custom classifier.

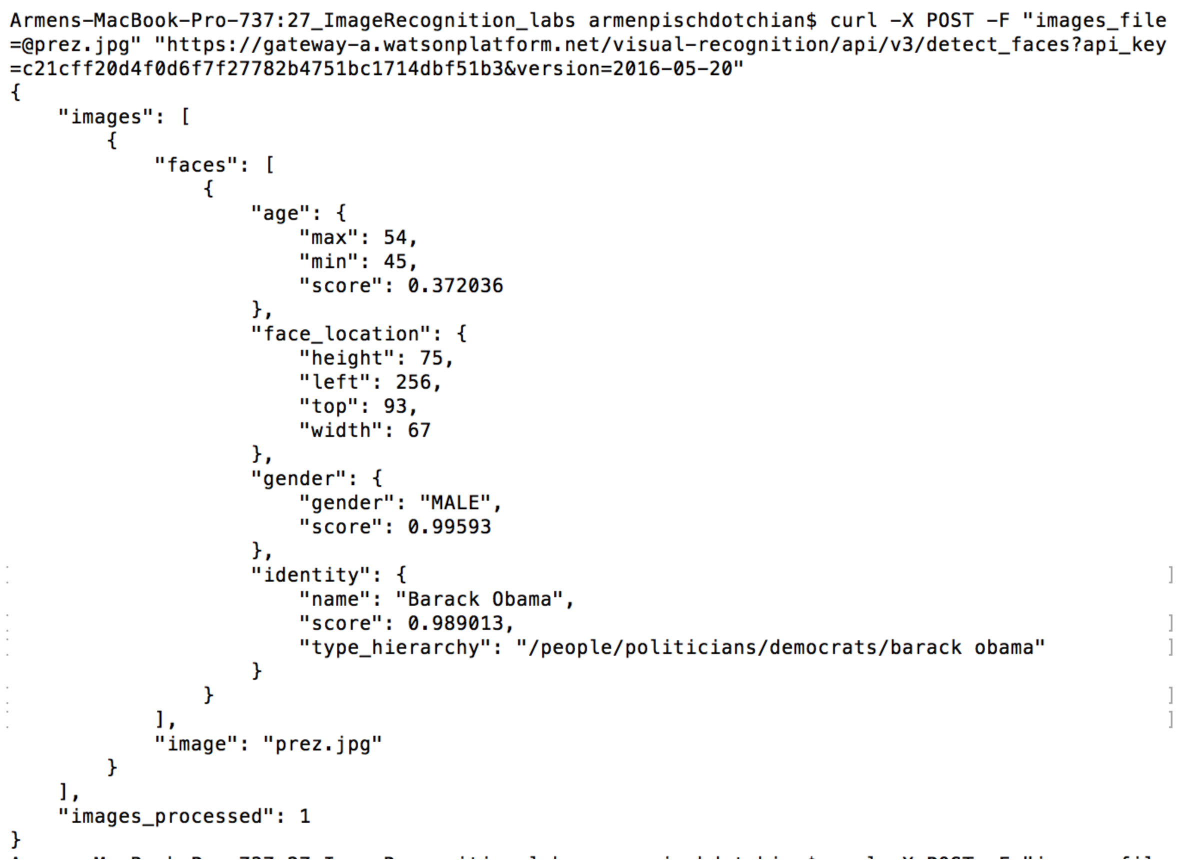
Task 3: Detecting faces in an image

The Visual Recognition service can detect faces in images and give information about them, such as where the face is located in the image and the estimated age range and gender for each face. The service can also identify many celebrities by name and can provide a knowledge graph so that you can perform interesting aggregations into higher-level concepts.

1. Download the sample file prez.jpg.
2. Call the `POST /v3/detect_faces` method with the following cURL command, which uploads the image to analyze (maximum image size is 2 MB):
   - Replace `<api-key>` with the service credentials you copied in Task 1.
   - Modify the location of the `images_file` to point to where you saved the .jpg file.

   ```
   curl -X POST -F "images_file=@prez.jpg" "https://gateway-
   a.watsonplatform.net/visual-recognition/api/v3/detect_faces?api_key=<api-
   key>&version=2016-05-20"
   ```

   The API returns a response that includes a location, age estimate, gender, identity and type hierarchy (if the service recognizes that face), and a score for each. Scores range from 0 to 1, with a higher score indicating greater correlation.



```
Armens-MacBook-Pro-737:27_ImageRecognition_labs armenpischdotchian$ curl -X POST -F "images_file
=@prez.jpg" "https://gateway-a.watsonplatform.net/visual-recognition/api/v3/detect_faces?api_key
=c21cff20d4f0d6f7f27782b4751bc1714dbf51b3&version=2016-05-20"
{
    "images": [
        {
            "faces": [
                {
                    "age": {
                        "max": 54,
                        "min": 45,
                        "score": 0.372036
                    },
                    "face_location": {
                        "height": 75,
                        "left": 256,
                        "top": 93,
                        "width": 67
                    },
                    "gender": {
                        "gender": "MALE",
                        "score": 0.99593
                    },
                    "identity": {
                        "name": "Barack Obama",
                        "score": 0.989013,
                        "type_hierarchy": "/people/politicians/democrats/barack obama"
                    }
                }
            ],
            "image": "prez.jpg"
        }
    ],
    "images_processed": 1
}
```

Tasks 4: Recognizing text in an image

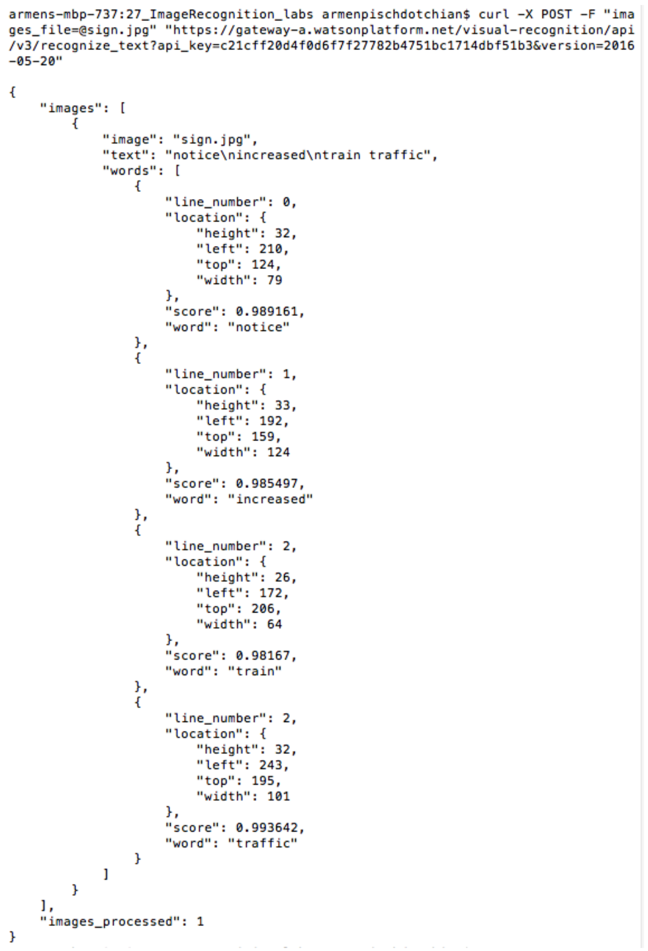
The Visual Recognition service has a functionality that can recognize text in an image, and provides information about that text, such as location and line number.

Note: The `/v3/recognize_text` endpoint is in beta and only supports English language text recognition.

1. Download the sample file sign.jpg or you can use an image with words that you find from Google images as you search for signs.
2. Call the `POST /v3/recognize_text` method with the following cURL command, which uploads the image to analyze (maximum image size is 2 MB):
    - o Replace <api-key> with the service credentials you copied in Task 1.
    - o Modify the location of the `images_file` to point to where you saved the .jpg file.

```
curl -X POST -F "images_file=@sign.jpg" "https://gateway-
a.watsonplatform.net/visual-
recognition/api/v3/recognize_text?api_key=c21cff20d4f0d6f7f27782b4751bc1714dbf5
1b3&version=2016-05-20"
```

The API returns a response that includes an aggregated text string of all words found in the image, and the location, line number, and score for each identified word. Scores range from 0 to 1, with a higher score indicating greater accuracy.



```
armens-mbp-737:27_ImageRecognition_labs armenpischdotchian$ curl -X POST -F "ima
ges_file=@sign.jpg" "https://gateway-a.watsonplatform.net/visual-recognition/api
/v3/recognize_text?api_key=c21cff20d4f0d6f7f27782b4751bc1714dbf51b3&version=2016
-05-20"

{
    "images": [
        {
            "image": "sign.jpg",
            "text": "notice\nincreased\ntrain traffic",
            "words": [
                {
                    "line_number": 0,
                    "location": {
                        "height": 32,
                        "left": 210,
                        "top": 124,
                        "width": 79
                    },
                    "score": 0.989161,
                    "word": "notice"
                },
                {
                    "line_number": 1,
                    "location": {
                        "height": 33,
                        "left": 192,
                        "top": 159,
                        "width": 124
                    },
                    "score": 0.985497,
                    "word": "increased"
                },
                {
                    "line_number": 2,
                    "location": {
                        "height": 26,
                        "left": 172,
                        "top": 206,
                        "width": 64
                    },
                    "score": 0.98167,
                    "word": "train"
                },
                {
                    "line_number": 2,
                    "location": {
                        "height": 32,
                        "left": 243,
                        "top": 195,
                        "width": 101
                    },
                    "score": 0.993642,
                    "word": "traffic"
                }
            ]
        }
    ],
    "images_processed": 1
}
```

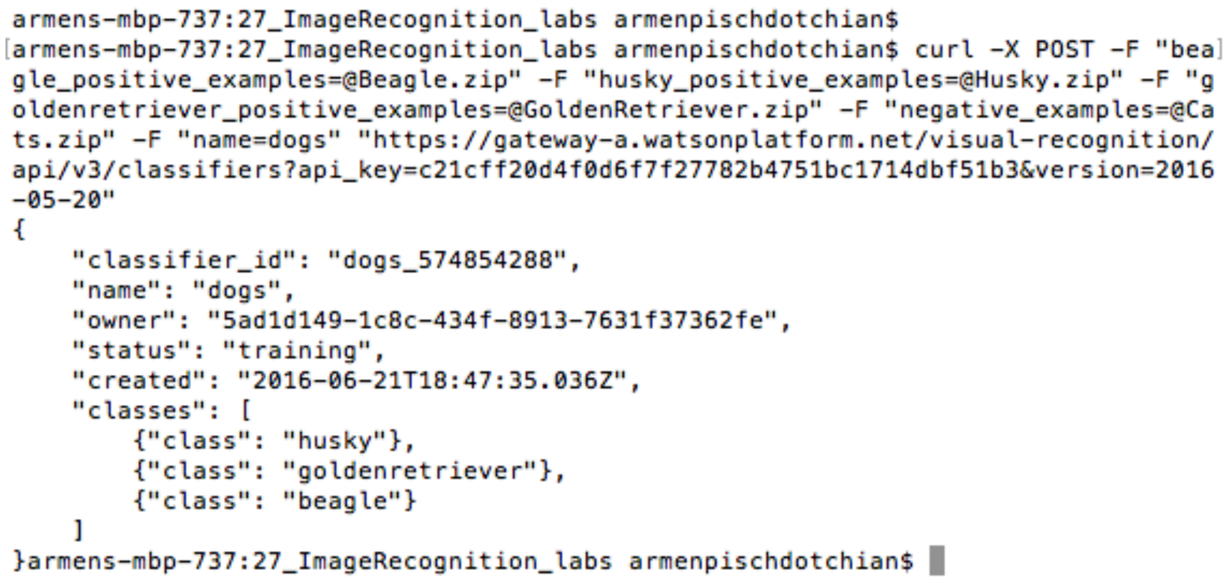
Task 5: Creating a custom classifier

The Visual Recognition service can learn from example images you upload to create a new, multi-faceted classifier. Each example file is trained against the other files in that call, and positive examples are stored as classes. (Negative example files are not stored as classes.) These classes are grouped to define a single classifier, but return their own scores.

1. Download the sample training files Husky.zip, GoldenRetriever.zip, Beagle.zip, and Cats.zip.
2. Call the `POST /v3/classifiers` method with the following cURL command, which uploads the training data and creates the classifier "dogs":
   - Specify a name for your classifier; in this example it is: dogs
   - Replace `<api-key>` with the service credentials you copied in the first Task.
   - For positive example files, the `_positive_examples` suffix is required. The prefix you choose (for example, Husky) becomes the name of that class.
   - Modify the location of the `{class}_positive_examples` to point to where you saved the .zip files.

```
curl -X POST -F "beagle_positive_examples=@Beagle.zip" -F
"husky_positive_examples=@Husky.zip" -F
"goldenretriever_positive_examples=@GoldenRetriever.zip" -F
"negative_examples=@Cats.zip" -F "name=dogs" "https://gateway-
a.watsonplatform.net/visual-
recognition/api/v3/classifiers?api_key=c21cff20d4f0d6f7f27782b4751bc1714dbf51b3
&version=2016-05-20"
```

The response includes a new classifier ID and status. For example:

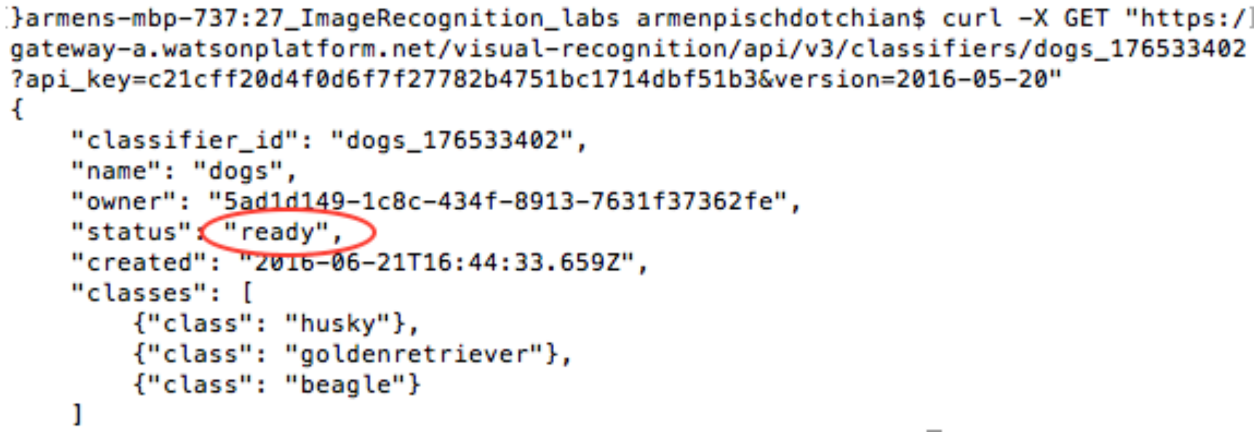


```
armens-mbp-737:27_ImageRecognition_labs armenpischdotchian$
armens-mbp-737:27_ImageRecognition_labs armenpischdotchian$ curl -X POST -F "bea
gle_positive_examples=@Beagle.zip" -F "husky_positive_examples=@Husky.zip" -F "g
oldenretriever_positive_examples=@GoldenRetriever.zip" -F "negative_examples=@Ca
ts.zip" -F "name=dogs" "https://gateway-a.watsonplatform.net/visual-recognition/
api/v3/classifiers?api_key=c21cff20d4f0d6f7f27782b4751bc1714dbf51b3&version=2016
-05-20"
{
    "classifier_id": "dogs_574854288",
    "name": "dogs",
    "owner": "5ad1d149-1c8c-434f-8913-7631f37362fe",
    "status": "training",
    "created": "2016-06-21T18:47:35.036Z",
    "classes": [
        {"class": "husky"},
        {"class": "goldenretriever"},
        {"class": "beagle"}
    ]
}armens-mbp-737:27_ImageRecognition_labs armenpischdotchian$
```

Training begins immediately and must finish before you can query the classifier.

3. Check the training status periodically until you see a status of "ready":
    - Issue a `GET /classifiers/{classifier_id}` call to retrieve the status of the classifier. In the following example, replace `{api-key}` and `{classifier_id}` with your information:

```
curl -X GET "https://gateway-a.watsonplatform.net/visual-
recognition/api/v3/classifiers/dogs_176533402?api_key=c21cff20d4f0d6f7f27782b47
51bc1714dbf51b3&version=2016-05-20"
```



```
}armens-mbp-737:27_ImageRecognition_labs armenpischdotchian$ curl -X GET "https:/
gateway-a.watsonplatform.net/visual-recognition/api/v3/classifiers/dogs_176533402
?api_key=c21cff20d4f0d6f7f27782b4751bc1714dbf51b3&version=2016-05-20"
{
    "classifier_id": "dogs_176533402",
    "name": "dogs",
    "owner": "5ad1d149-1c8c-434f-8913-7631f37362fe",
    "status": "ready",
    "created": "2016-06-21T16:44:33.659Z",
    "classes": [
        {"class": "husky"},
        {"class": "goldenretriever"},
        {"class": "beagle"}
    ]
```

Task 6: Classifying an image with a custom classifier

When the new classifier completes training, you can call it to see how it performs.

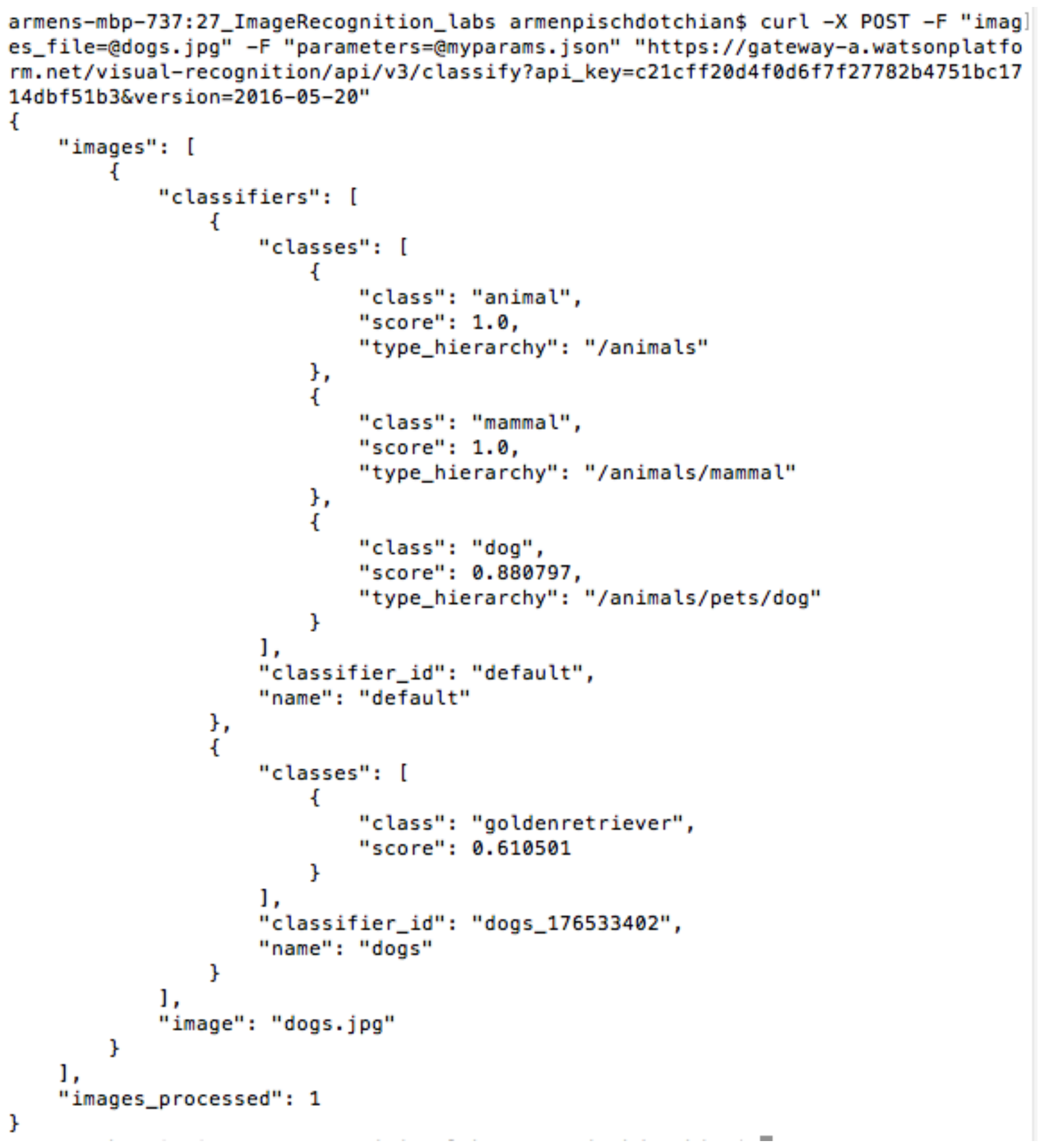
1. Create a JSON file called "myparams.json" that includes the parameters for your call, such as the `classifier_id` of your new classifier and the built-in classifier. A simple JSON file might look like the following:

```
{
     "classifier_ids": ["dogs_1941945966", "default"]
}
```

2. Use the `POST /v3/classify` method to test your custom classifier. The following example uploads the image dogs.jpg and classifies it against the "fruits" classifier.
    - Replace `{api-key}` with the service credentials you copied in the first Task.
    - To classify against built-in classes, omit the `parameters` parameter.

```
curl -X POST -F "images_file=@dogs.jpg" -F "parameters=@myparams.json"
"https://gateway-a.watsonplatform.net/visual-
recognition/api/v3/classify?api_key=c21cff20d4f0d6f7f27782b4751bc1714dbf51b3&ve
rsion=2016-05-20"
```

The API returns a response that includes classifiers, their classes, and a score for each. Scores range from 0 â€“ 1, with a higher score indicating greater correlation. The `/v3/classify` calls omit low-scoring classes by default if high scoring classes are identified. You can set a minimum score to display by specifying a floating point value for the `threshold` parameter in your call.



```
armens-mbp-737:27_ImageRecognition_labs armenpischdotchian$ curl -X POST -F "imag
es_file=@dogs.jpg" -F "parameters=@myparams.json" "https://gateway-a.watsonplatfo
rm.net/visual-recognition/api/v3/classify?api_key=c21cff20d4f0d6f7f27782b4751bc17
14dbf51b3&version=2016-05-20"
{
    "images": [
        {
            "classifiers": [
                {
                    "classes": [
                        {
                            "class": "animal",
                            "score": 1.0,
                            "type_hierarchy": "/animals"
                        },
                        {
                            "class": "mammal",
                            "score": 1.0,
                            "type_hierarchy": "/animals/mammal"
                        },
                        {
                            "class": "dog",
                            "score": 0.880797,
                            "type_hierarchy": "/animals/pets/dog"
                        }
                    ],
                    "classifier_id": "default",
                    "name": "default"
                },
                {
                    "classes": [
                        {
                            "class": "goldenretriever",
                            "score": 0.610501
                        }
                    ],
                    "classifier_id": "dogs_176533402",
                    "name": "dogs"
                }
            ],
            "image": "dogs.jpg"
        }
    ],
    "images_processed": 1
}
```

3. Review your results.

You're done! You created, trained, and queried a custom classifier with the Visual Recognition service.

Deleting your custom classifier

You might want to delete your custom classifier to begin developing your application with a clean instance. Or, because retraining classifiers is not possible at this time, you might want to delete your classifier, tweak your training data, and create a new one.

To delete the classifier, call the `DELETE /v3/classifiers/{classifier_id}` method.

Issue the following cURL command to delete the classifier. Replace `{api_key)` with your information. Include the `classifier_id` for the classifier you want to delete:

```
curl -X DELETE "https://gateway-a.watsonplatform.net/visual-
recognition/api/v3/classifiers/{classifier_id}?api_key=<api-key>&version=2016-05-20"
```

Task 7: Training your custom images

The zip file that you downloaded from Github contains two other folders each contain images of tanks. This sample is the work that Chris Madison from IBM has done and you can find further details on it in this link: https://cmadison.me/ and you can find his images here: https://github.com/tankcdr/blog-tanks-visual_recognition

In the meantime, in this section, without guiding you step by step, draws from your work thus far and invites you to custom train your own images. Of course, all responses are in JSON format. UI design and working with DoJo is not part of this workshop.

Notice that Chris has about 50 images per folder: each folder contains images of tanks. So conceivably you can provide your own images compressed in a zip file. Please note the guidelines of image preparation as stated in the beginning of this guide.

There is yet a third folder that Chris has named: test. After you have trained the classifier, one group of tanks being positive and another group being negative, then you upload a very similar looking tank from the test folder and it should identify it as the proper tank (or not).

You then reverse the positive and negative assumptions and see how well it identifies other similar looking tanks.

Basically, you have to perform Task 5 and Task 6, creating two distinct classifiers. Typically you would assign the classifier name to be type of images that are positive.

Enjoy this section, since it is all about your images!