Lab - Explore Data Sets and Models

Lab: Getting Started with the Model Asset Exchange and the Data Asset Exchange

In this lab you will explore the Model Asset Exchange (MAX) and the Data Asset Exchange (DAX), which are two open source Data Science resources on IBM Developer.

Upon completion of part 1 of this lab ("Explore deep learning models") you will be able to:

- Find ready-to-use deep learning models on the Model Asset Exchange
- Locate resources that guide you through deployment in a local or cloud environment
- Explore the deep learning model-serving microservice API using your web browser
- Articulate how developers can consume those microservices

Upon completion of part 2 ("Explore deep learning data sets") you will know:

- Where to find open data sets on IBM Developer
- How to explore those data sets

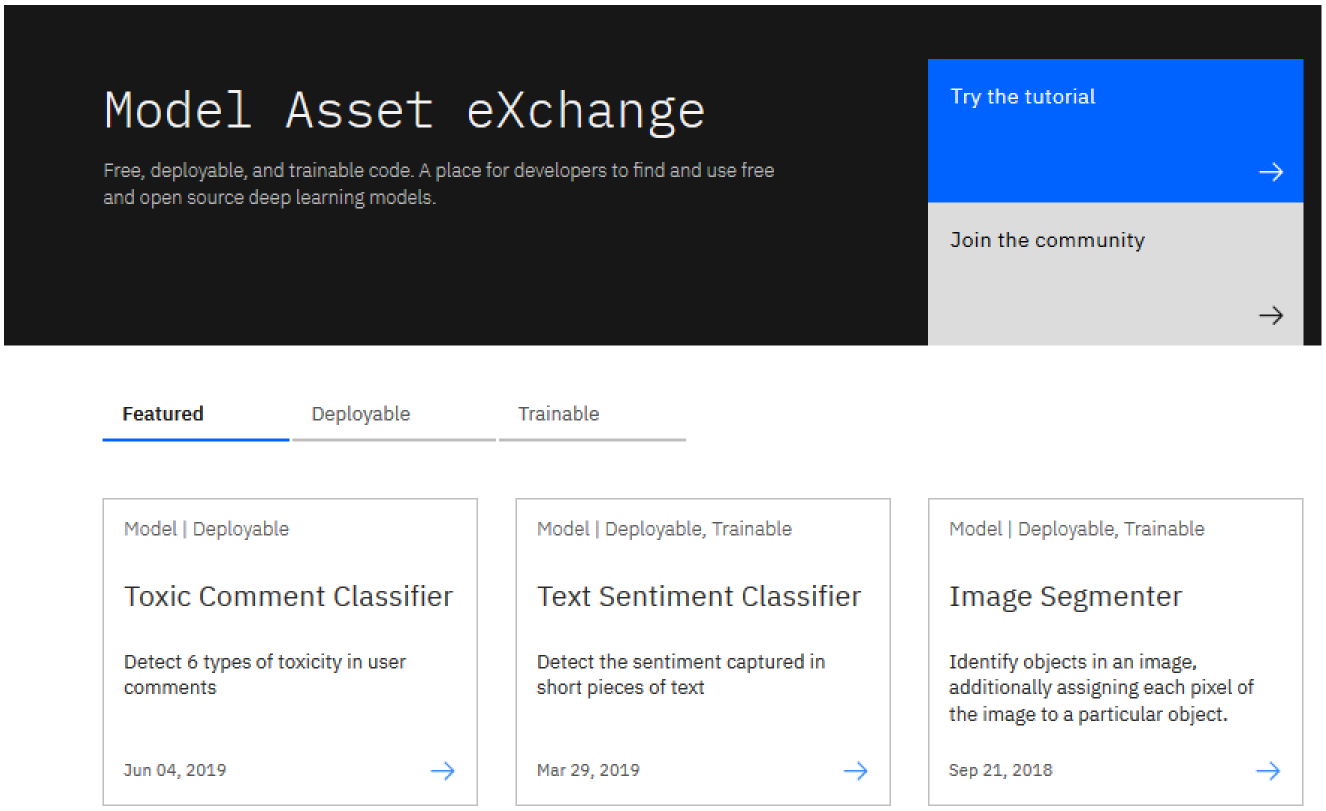
It will take you approximately 30 minutes to complete the lab. Only a web browser is required to complete the tasks.

Part 1: Explore deep learning models

The Model Asset Exchange is a curated repository of open source deep learning models for a variety of domains, such as text, image, audio, and video processing.

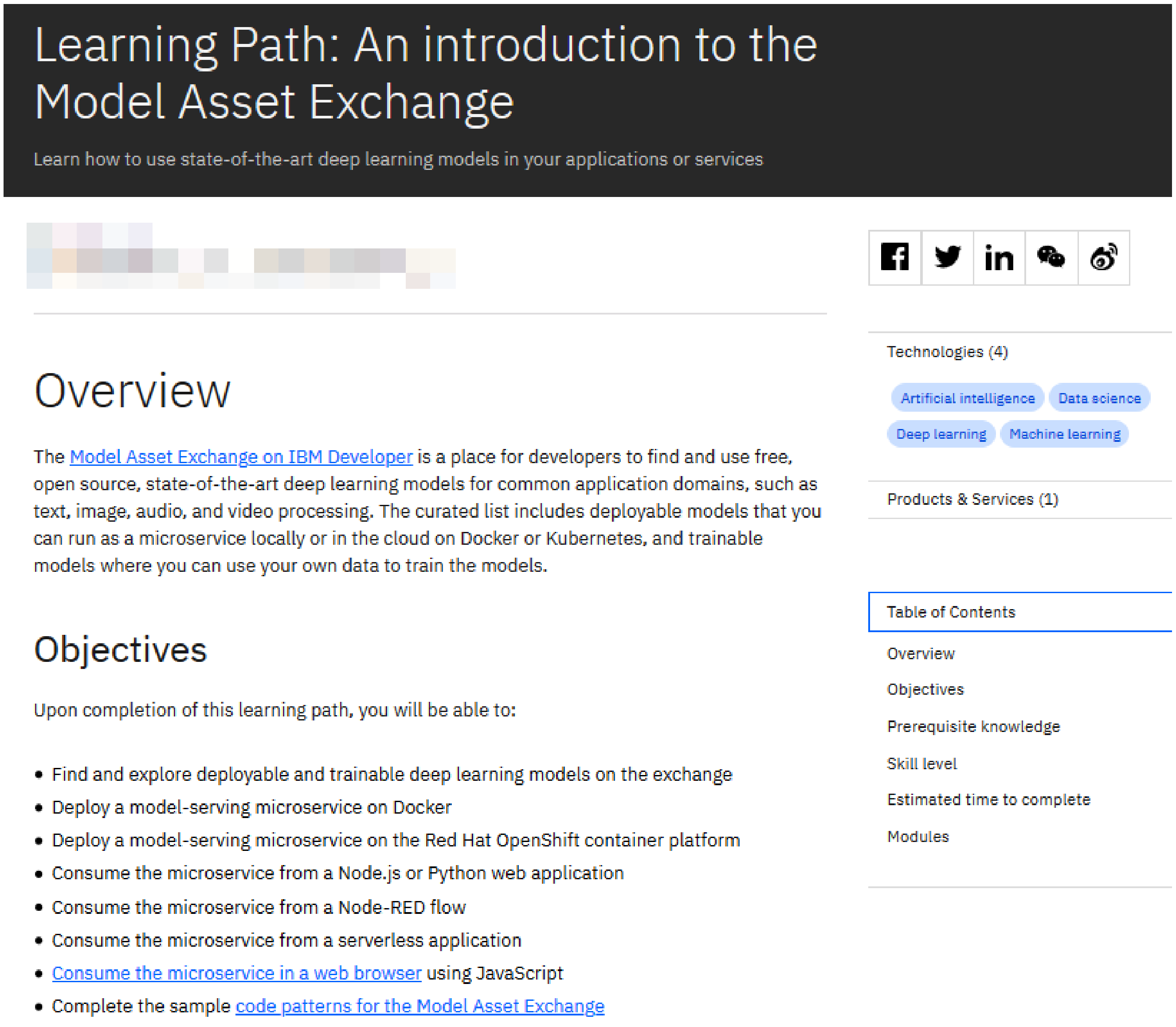
1. Open https://developer.ibm.com/ in your web browser.

2. From the main menu select **Open Source at IBM** > **Model Asset eXchange**. The MAX home page is displayed.


Model Asset eXchange
Free, deployable, and trainable code. A place for developers to find and use free and open source deep learning models.
Try the tutorial
Join the community
Featured
Deployable
Trainable
Model | Deployable
Toxic Comment Classifier
Detect 6 types of toxicity in user comments
Jun 04, 2019
Model | Deployable, Trainable
Text Sentiment Classifier
Detect the sentiment captured in short pieces of text
Mar 29, 2019
Model | Deployable, Trainable
Image Segmenter
Identify objects in an image, additionally assigning each pixel of the image to a particular object.
Sep 21, 2018

In this introductory lab exercise, we are going to focus on a few MAX key features. More detailed information is available in the Learning Path, which covers common deployment and consumption scenarios.

3. Open the Learning Path (**Try the tutorial**) in a new browser window (or tab) and briefly review the objectives and modules.


Learning Path: An introduction to the Model Asset Exchange

Learn how to use state-of-the-art deep learning models in your applications or services

Overview

The Model Asset Exchange on IBM Developer is a place for developers to find and use free, open source, state-of-the-art deep learning models for common application domains, such as text, image, audio, and video processing. The curated list includes deployable models that you can run as a microservice locally or in the cloud on Docker or Kubernetes, and trainable models where you can use your own data to train the models.

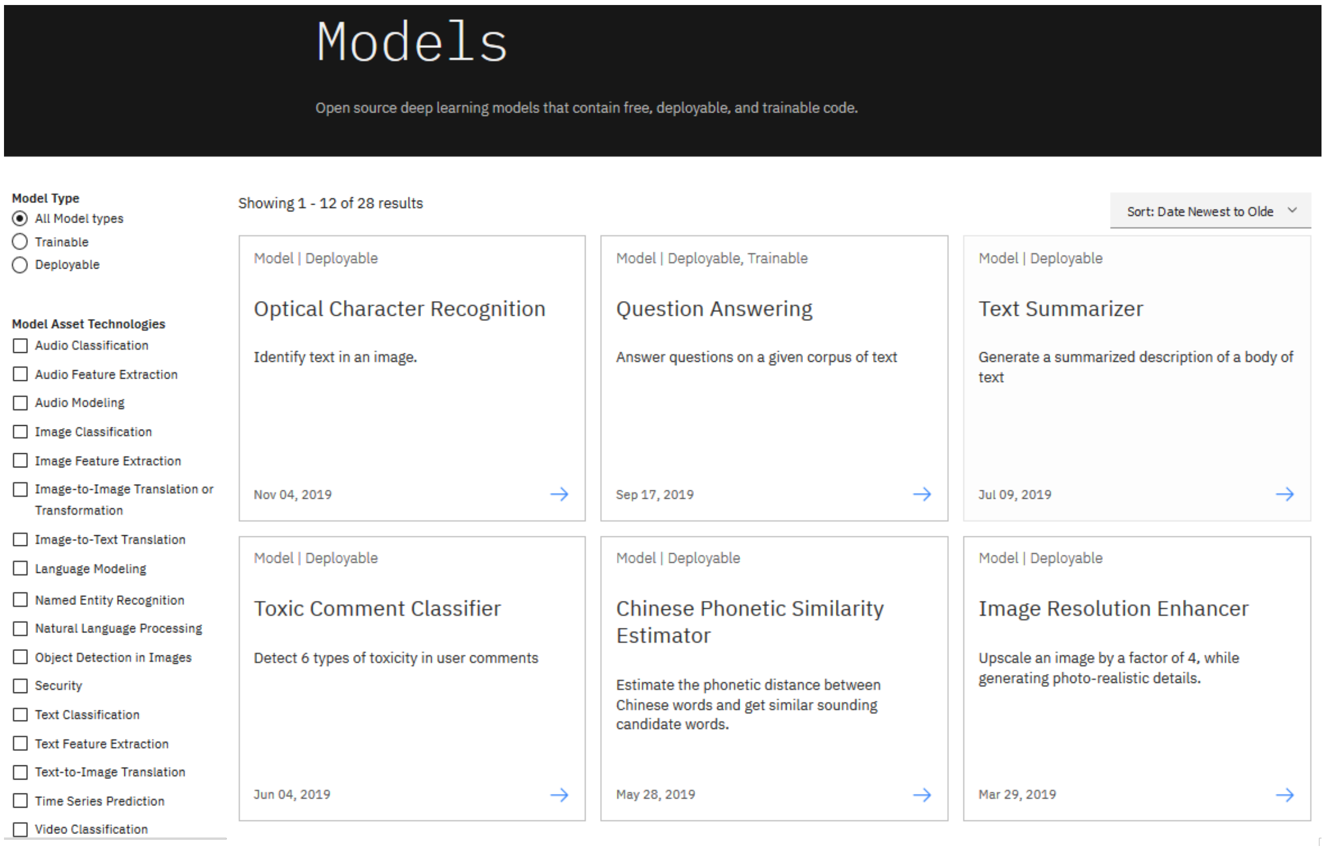
Objectives

Upon completion of this learning path, you will be able to:

- Find and explore deployable and trainable deep learning models on the exchange
- Deploy a model-serving microservice on Docker
- Deploy a model-serving microservice on the Red Hat OpenShift container platform
- Consume the microservice from a Node.js or Python web application
- Consume the microservice from a Node-RED flow
- Consume the microservice from a serverless application
- Consume the microservice in a web browser using JavaScript
- Complete the sample code patterns for the Model Asset Exchange

4. Close the Learning Path browser window (or tab).

5. The main page displays a small model selection. Scroll down and click **View all models** to review the complete list.



Models

Open source deep learning models that contain free, deployable, and trainable code.

Model Type
- ● All Model types
- ○ Trainable
- ○ Deployable

Model Asset Technologies
- ☐ Audio Classification
- ☐ Audio Feature Extraction
- ☐ Audio Modeling
- ☐ Image Classification
- ☐ Image Feature Extraction
- ☐ Image-to-Image Translation or Transformation
- ☐ Image-to-Text Translation
- ☐ Language Modeling
- ☐ Named Entity Recognition
- ☐ Natural Language Processing
- ☐ Object Detection in Images
- ☐ Security
- ☐ Text Classification
- ☐ Text Feature Extraction
- ☐ Text-to-Image Translation
- ☐ Time Series Prediction
- ☐ Video Classification

Showing 1 - 12 of 28 results

Sort: Date Newest to Olde

Model | Deployable
Optical Character Recognition
Identify text in an image.
Nov 04, 2019

Model | Deployable, Trainable
Question Answering
Answer questions on a given corpus of text
Sep 17, 2019

Model | Deployable
Text Summarizer
Generate a summarized description of a body of text
Jul 09, 2019

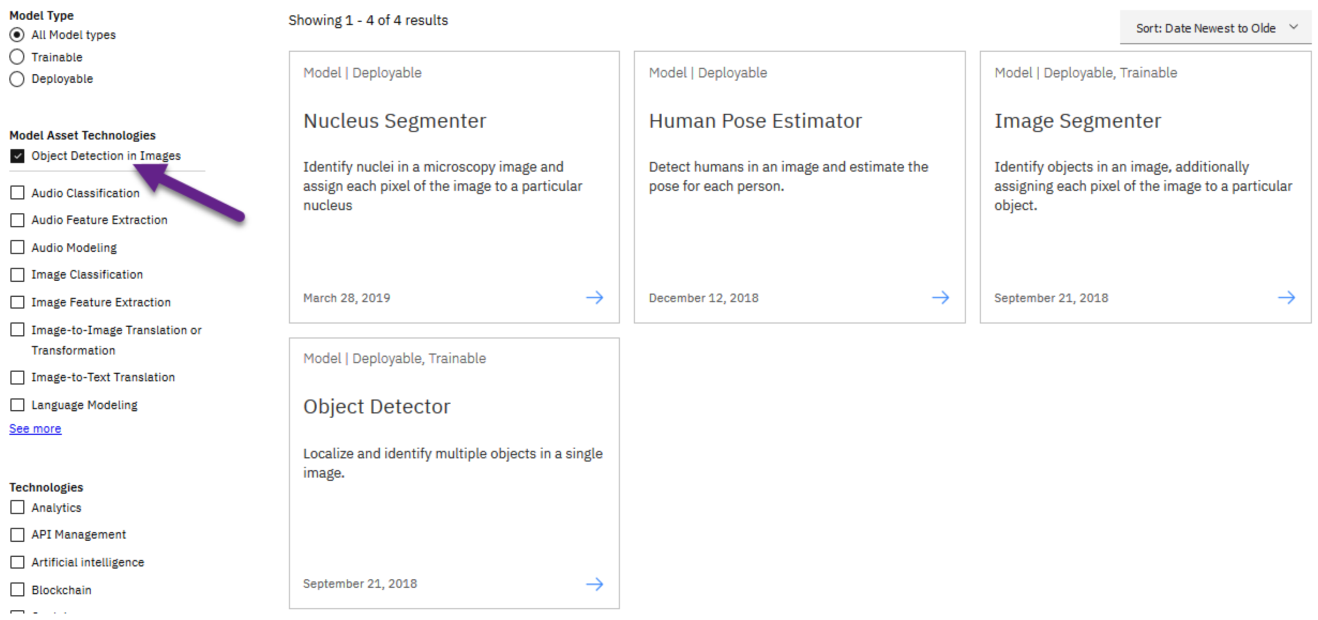
Model | Deployable
Toxic Comment Classifier
Detect 6 types of toxicity in user comments
Jun 04, 2019

Model | Deployable
Chinese Phonetic Similarity Estimator
Estimate the phonetic distance between Chinese words and get similar sounding candidate words.
May 28, 2019

Model | Deployable
Image Resolution Enhancer
Upscale an image by a factor of 4, while generating photo-realistic details.
Mar 29, 2019

You can filter models by type and domain using the filter on the left-hand side.

6. Select **Object Detection in Images** from the **Model Asset Technologies** list on the left side.



Model Type
- ● All Model types
- ○ Trainable
- ○ Deployable

Model Asset Technologies
- ☑ Object Detection in Images
- ☐ Audio Classification
- ☐ Audio Feature Extraction
- ☐ Audio Modeling
- ☐ Image Classification
- ☐ Image Feature Extraction
- ☐ Image-to-Image Translation or Transformation
- ☐ Image-to-Text Translation
- ☐ Language Modeling

See more

Technologies
- ☐ Analytics
- ☐ API Management
- ☐ Artificial intelligence
- ☐ Blockchain

Showing 1 - 4 of 4 results

Sort: Date Newest to Olde

Model | Deployable
Nucleus Segmenter
Identify nuclei in a microscopy image and assign each pixel of the image to a particular nucleus
March 28, 2019

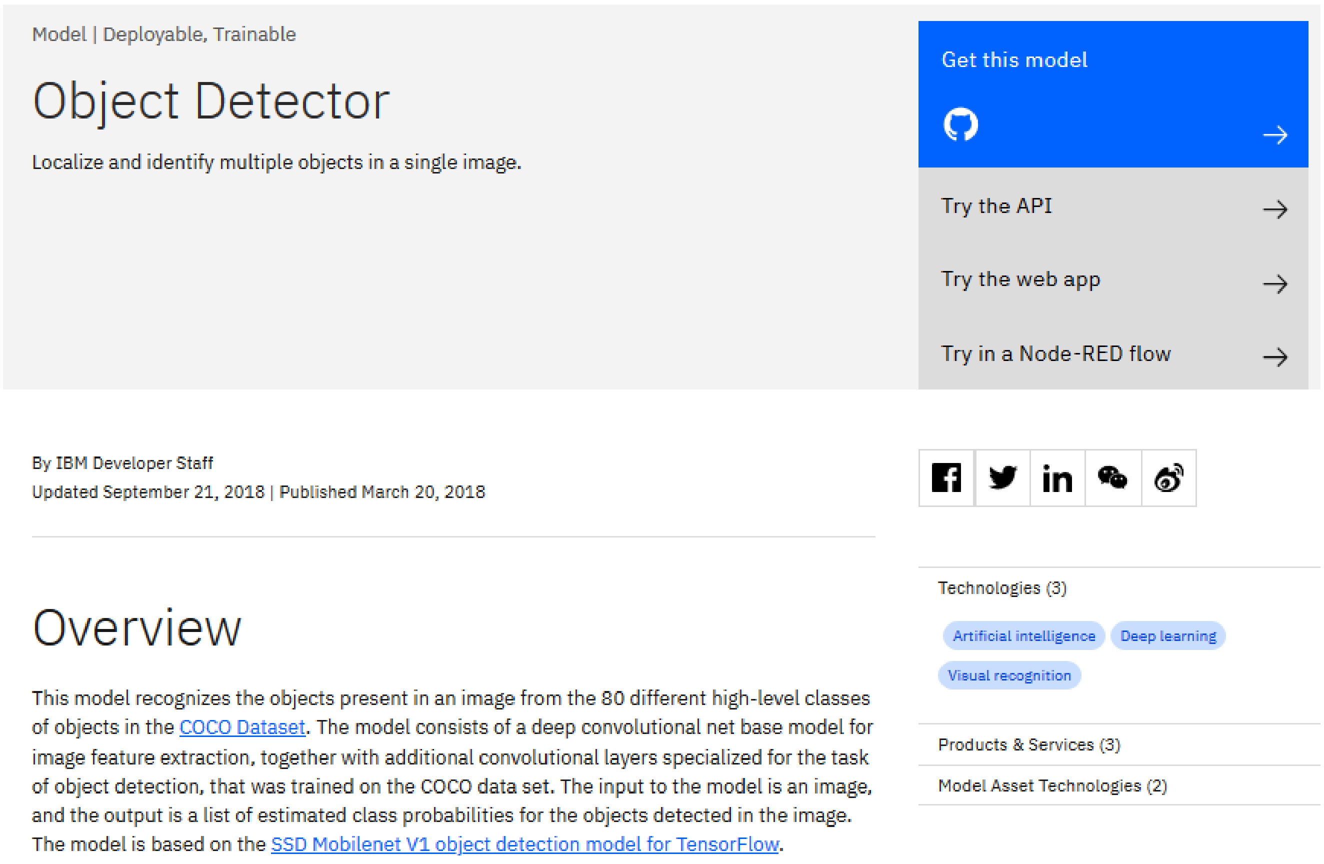
Model | Deployable
Human Pose Estimator
Detect humans in an image and estimate the pose for each person.
December 12, 2018

Model | Deployable, Trainable
Image Segmenter
Identify objects in an image, additionally assigning each pixel of the image to a particular object.
September 21, 2018

Model | Deployable, Trainable
Object Detector
Localize and identify multiple objects in a single image.
September 21, 2018

Four models should be displayed: Nucleus Segmenter, Human Pose Estimator, Image Segmenter, and Object Detector.

7. Select the **Object Detector** model.


Model | Deployable, Trainable

Object Detector

Localize and identify multiple objects in a single image.

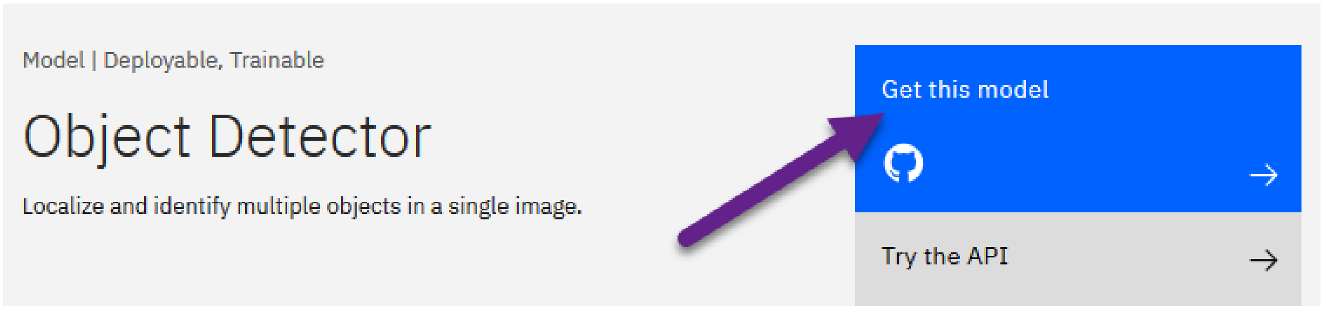
By IBM Developer Staff
Updated September 21, 2018 | Published March 20, 2018

Overview

This model recognizes the objects present in an image from the 80 different high-level classes of objects in the COCO Dataset. The model consists of a deep convolutional net base model for image feature extraction, together with additional convolutional layers specialized for the task of object detection, that was trained on the COCO data set. The input to the model is an image, and the output is a list of estimated class probabilities for the objects detected in the image. The model is based on the SSD Mobilenet V1 object detection model for TensorFlow.

On the model page you can learn about the underlying technology, read up on related research, and explore deployment and consumption options.

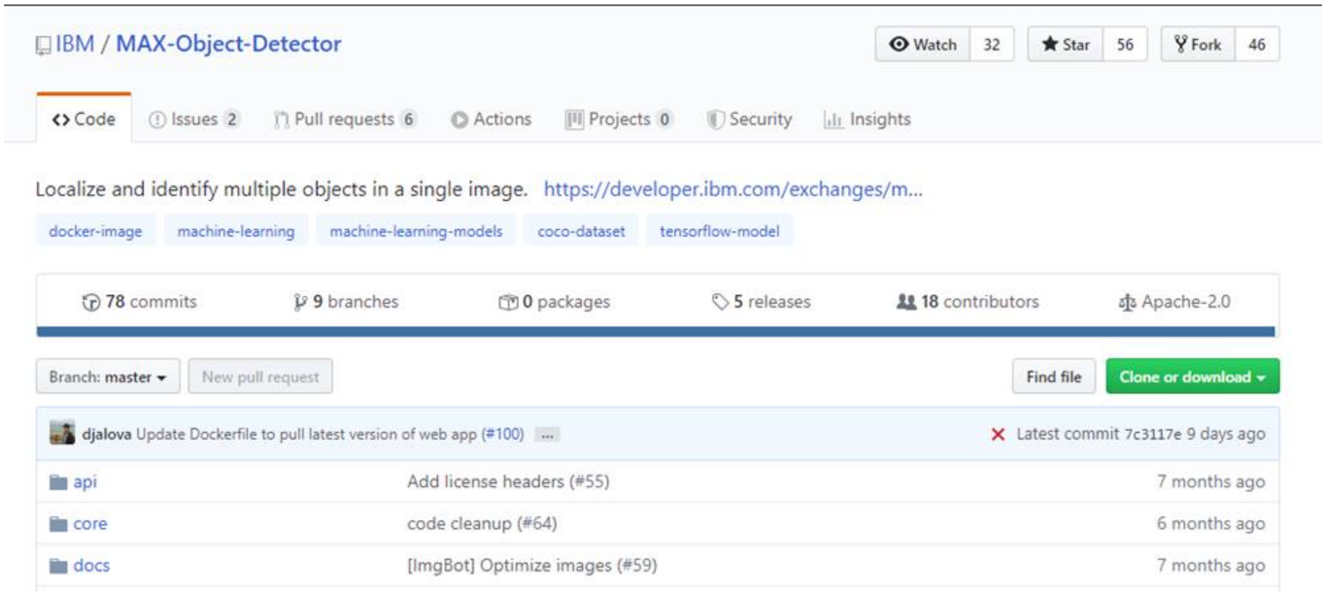
8. The model's source is published on GitHub and can be downloaded and modified if desired. Click **Get this model** to open the repository in a new browser tab (or window).


Model | Deployable, Trainable

Object Detector

Localize and identify multiple objects in a single image.

The repository contains everything a developer needs to manually build a customized version of the model-serving microservice.



Localize and identify multiple objects in a single image. https://developer.ibm.com/exchanges/m...

docker-image machine-learning machine-learning-models coco-dataset tensorflow-model

78 commits | 9 branches | 0 packages | 5 releases | 18 contributors | Apache-2.0

djalova Update Dockerfile to pull latest version of web app (#100) ... Latest commit 7c3117e 9 days ago

api	Add license headers (#55)	7 months ago
core	code cleanup (#64)	6 months ago
docs	[ImgBot] Optimize images (#59)	7 months ago

9. Close the GitHub browser tab (or window).

10. On the Object Detector page (https://developer.ibm.com/exchanges/models/all/max-object-detector/) navigate to the **Deployment** section. All MAX models are distributed as containerized applications, which can be deployed as a microservice in a local environment, a hybrid cloud environment or a cloud environment using Docker or Kubernetes.

- Deploy from Dockerhub:

  ```
  docker run -it -p 5000:5000 codait/max-object-detector
  ```

  Show more

- Deploy on Red Hat OpenShift:

  Follow the instructions for the OpenShift web console or the OpenShift Container Platform CLI in this tutorial and specify `codait/max-object-detector` as the image name.

- Deploy on Kubernetes:

  ```
  kubectl apply -f https://raw.githubusercontent.com/IBM/MAX-Object-Detector/master/n
  ```

  Show more

  A more elaborate tutorial on how to deploy this MAX model to production on IBM Cloud can be found here.

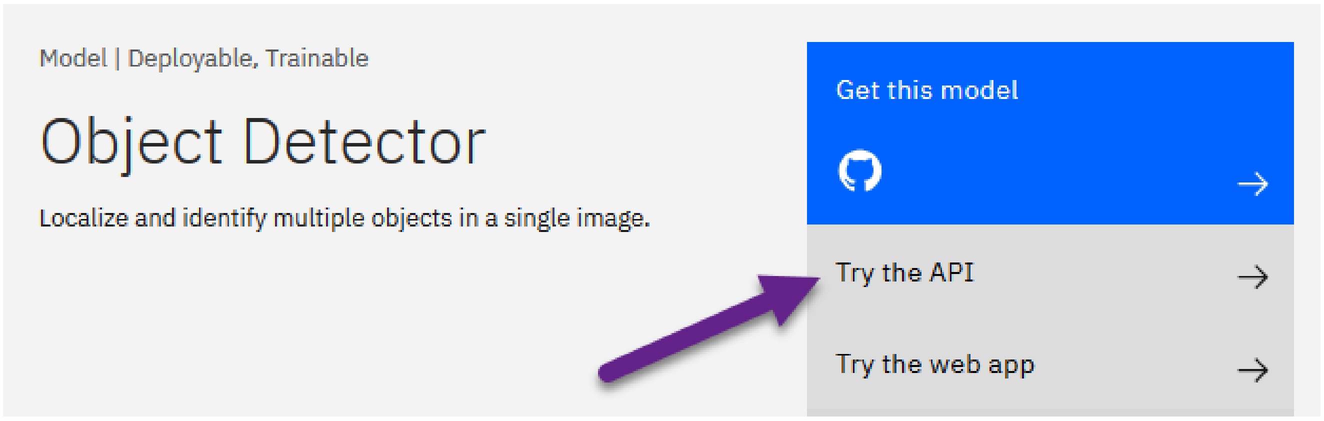
- Locally: follow the instructions in the model README on GitHub

It is beyond the scope of this lab to discuss the deployment steps. However, you can explore the model-serving microservice without having to deploy anything, which we'll discuss in the next section.

Explore a MAX model-serving microservice

To explore a microservice you can access an evaluation microservice instance that is hosted on IBM Cloud.

11. Scroll to the top of the page and click **Try the API**.

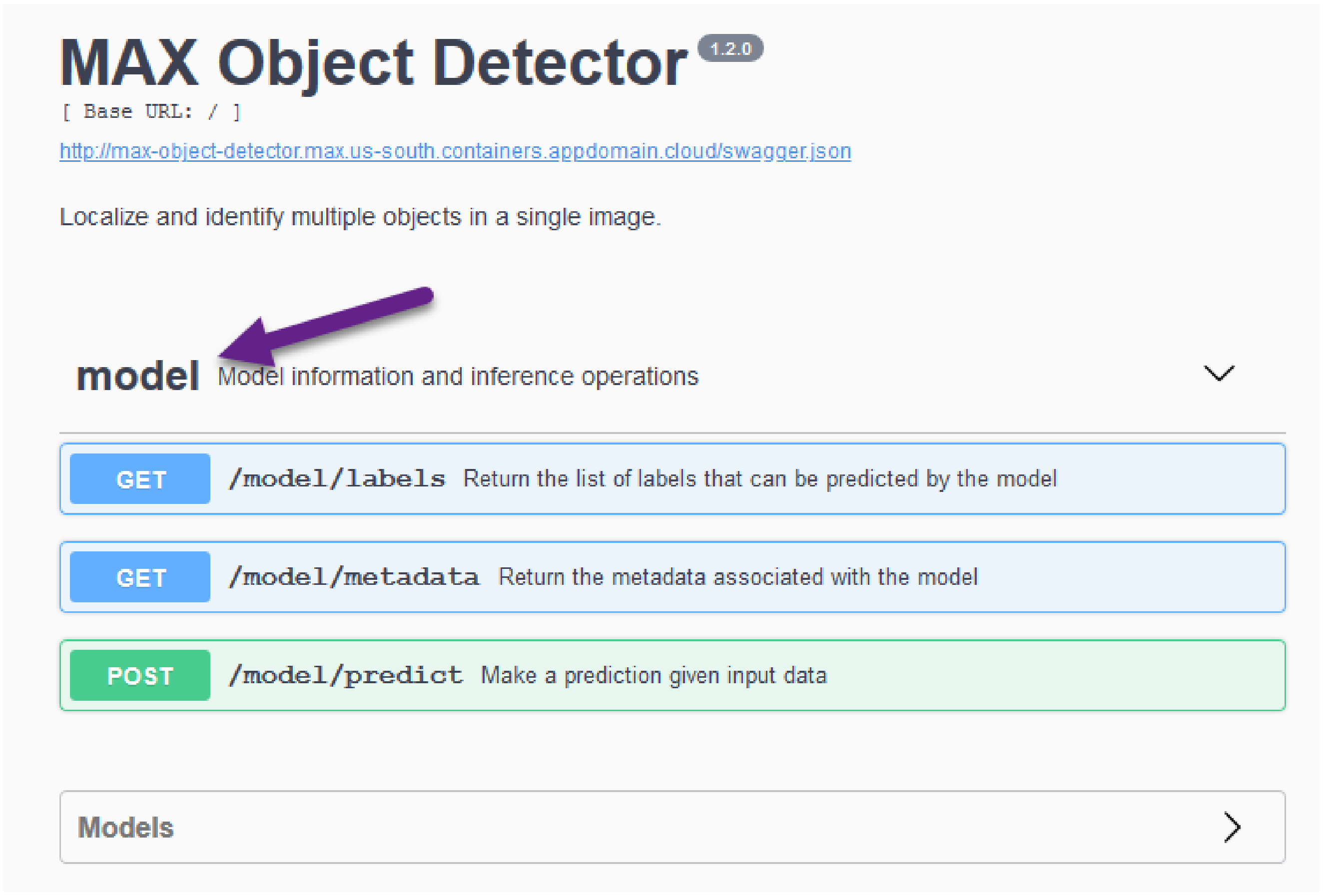

Model | Deployable, Trainable

Object Detector

Localize and identify multiple objects in a single image.

12. The microservice's API specification page opens in a new browser tab (or window).

13. Expand the **model** section. The microservice's REST endpoints are displayed.


MAX Object Detector 1.2.0

[Base URL: /]

http://max-object-detector.max.us-south.containers.appdomain.cloud/swagger.json

Localize and identify multiple objects in a single image.

model Model information and inference operations

GET /model/labels Return the list of labels that can be predicted by the model

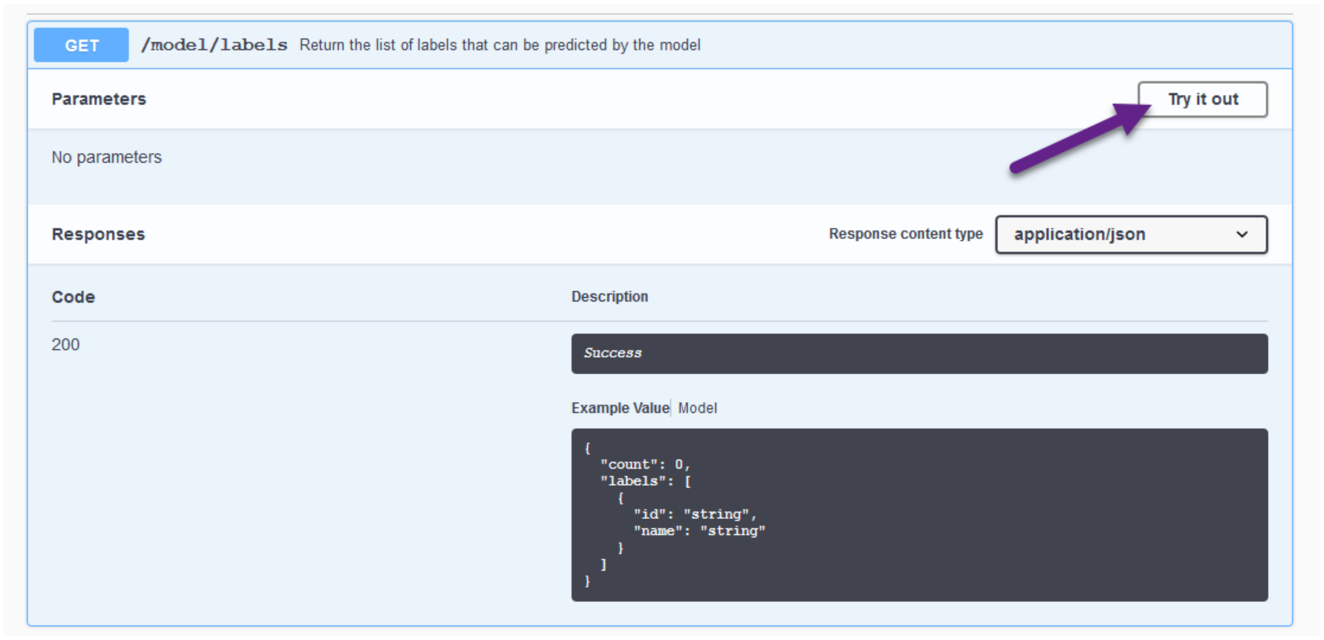
GET /model/metadata Return the metadata associated with the model

POST /model/predict Make a prediction given input data

The Object Detector microservice exposes three endpoints that applications can access: a *labels* endpoint, a *metadata* endpoint and a *predict* endpoint. Let's explore the *labels* endpoint.

14. Click **GET /model/labels**. This endpoint returns a list of objects that this model can detect.

15. Click **Try it out** and **Execute**.



GET /model/labels Return the list of labels that can be predicted by the model

Parameters

No parameters

Responses

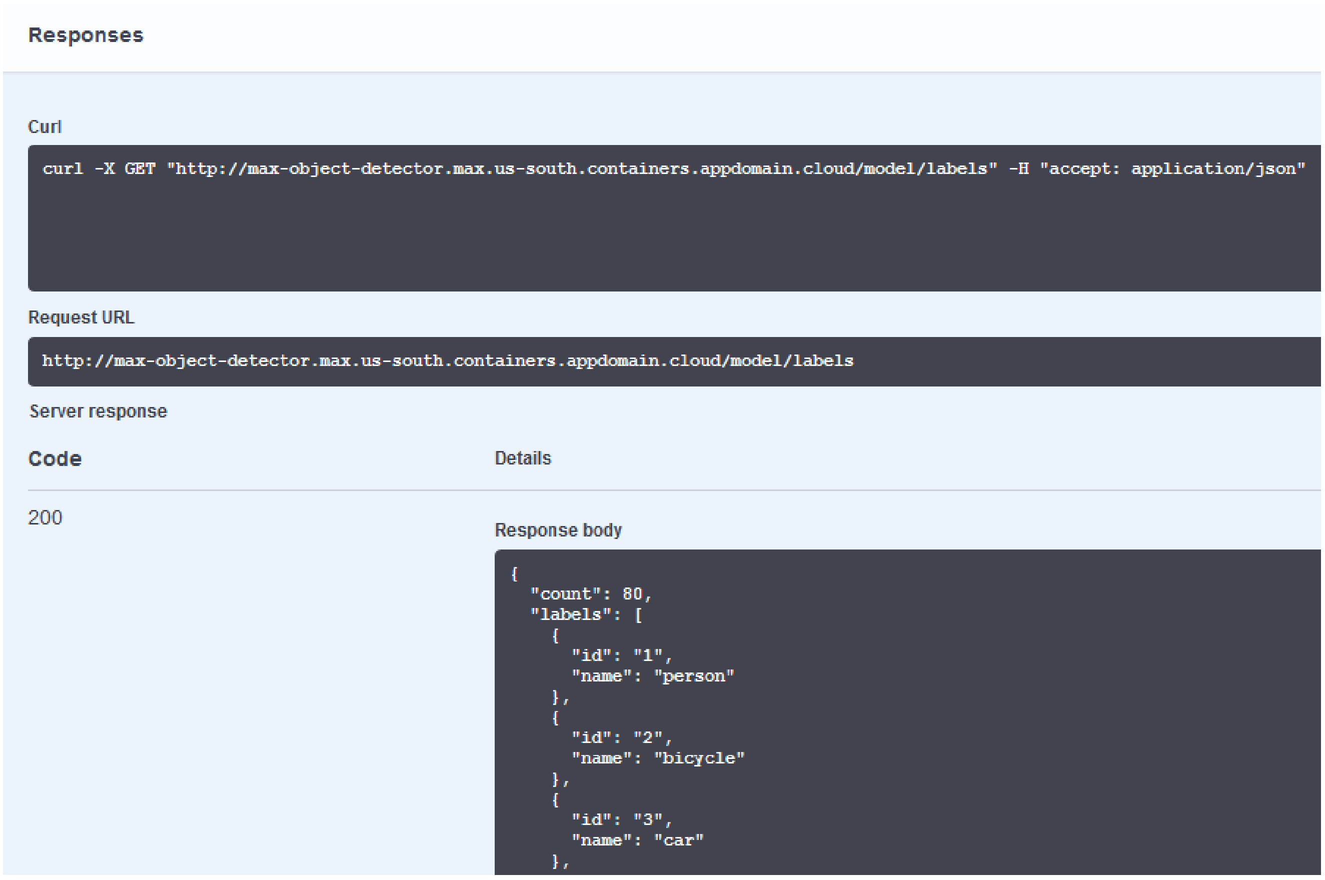
Response content type: application/json

Code	Description
200	Success

Example Value | Model

```
{
  "count": 0,
  "labels": [
    {
      "id": "string",
      "name": "string"
    }
  ]
}
```

The response indicates that this model was trained to identify 80 different types of objects, such as persons, bicycles, and cars.



Responses

Curl

```
curl -X GET "http://max-object-detector.max.us-south.containers.appdomain.cloud/model/labels" -H "accept: application/json"
```

Request URL

```
http://max-object-detector.max.us-south.containers.appdomain.cloud/model/labels
```

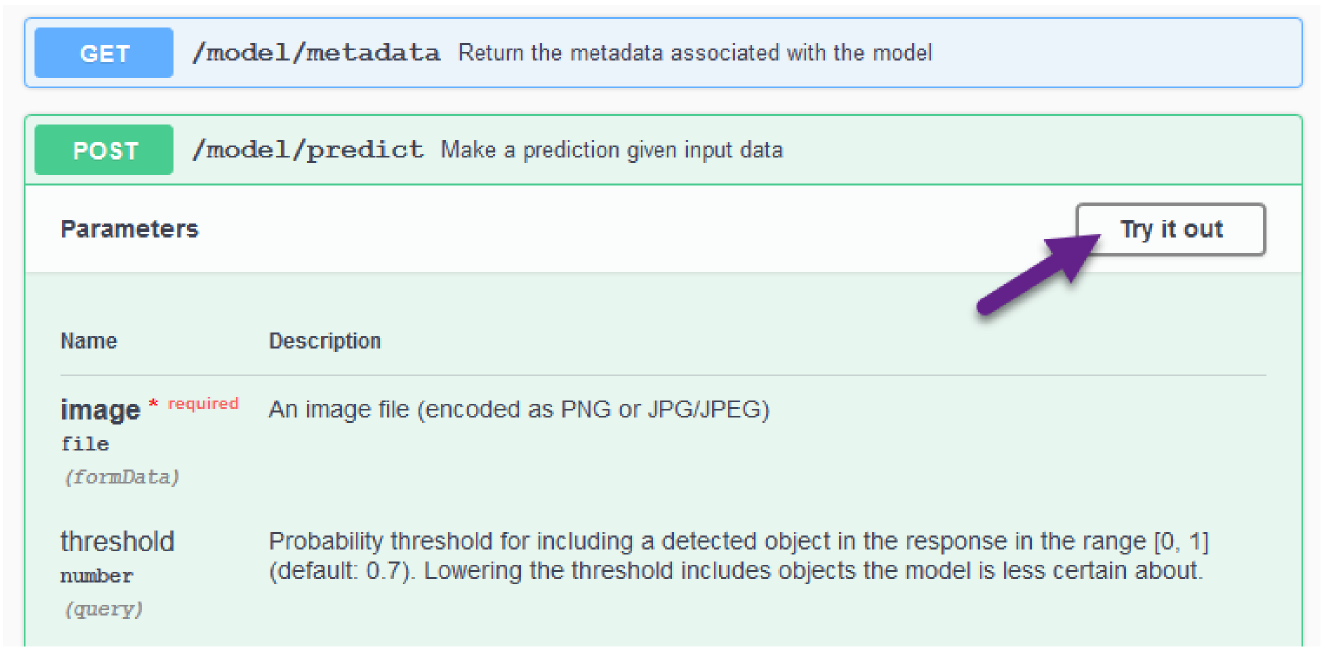
Server response

Code	Details
200	Response body

```
{
  "count": 80,
  "labels": [
    {
      "id": "1",
      "name": "person"
    },
    {
      "id": "2",
      "name": "bicycle"
    },
    {
      "id": "3",
      "name": "car"
    },
```

You can train some models, like the Object Detector, using your own data with the help of the Watson Machine Learning service on IBM Cloud. (You won't be doing that in this lab though.)

16. Click **POST /models/predict** and **Try it out** to test the model-serving endpoint. This endpoint is what your applications would use to analyze the content of an image.


GET /model/metadata Return the metadata associated with the model

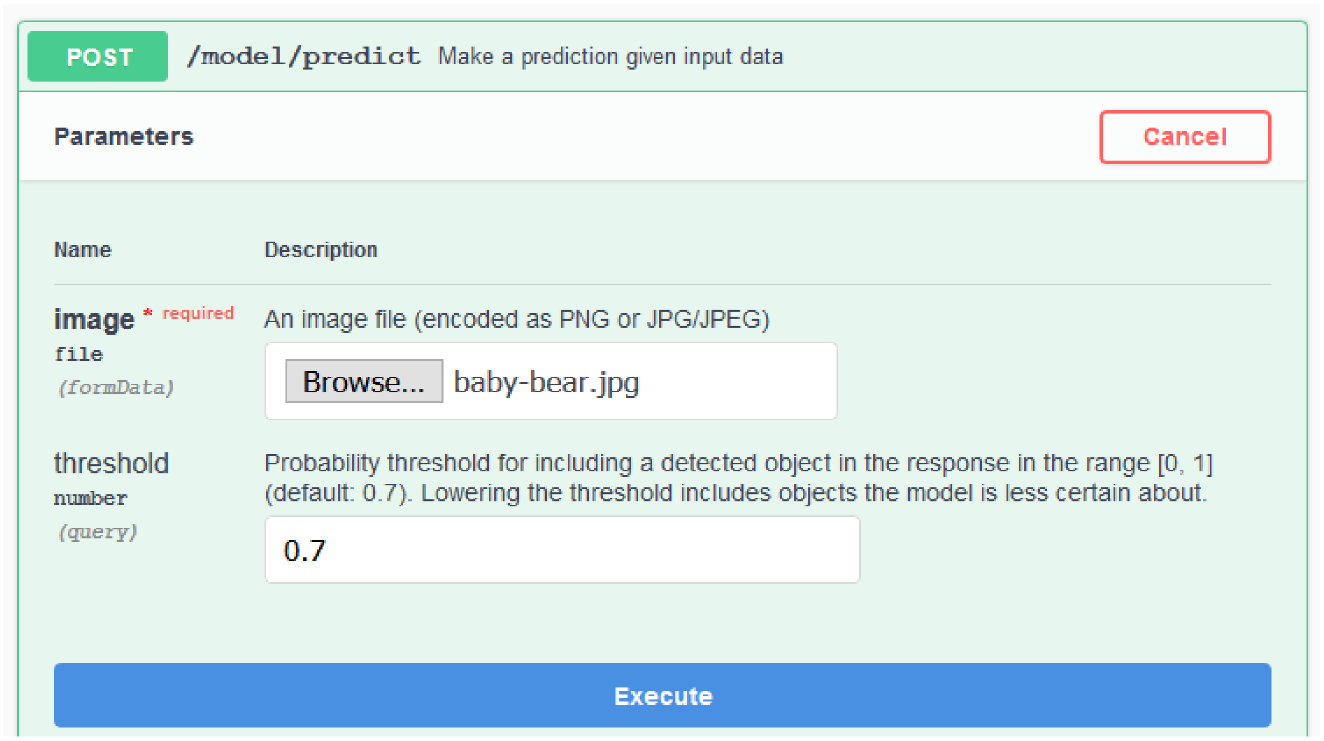
POST /model/predict Make a prediction given input data

Parameters

Try it out

Name	Description
image * required file *(formData)*	An image file (encoded as PNG or JPG/JPEG)
threshold number *(query)*	Probability threshold for including a detected object in the response in the range [0, 1] (default: 0.7). Lowering the threshold includes objects the model is less certain about.

17. Browse to a JPG or PNG file on your local machine, accept the default threshold parameter and click **Execute** to invoke the endpoint. The microservice analyzes the image and returns a list of identified objects.


POST /model/predict Make a prediction given input data

Parameters

Cancel

Name	Description
image * required file *(formData)*	An image file (encoded as PNG or JPG/JPEG) Browse... baby-bear.jpg
threshold number *(query)*	Probability threshold for including a detected object in the response in the range [0, 1] (default: 0.7). Lowering the threshold includes objects the model is less certain about. 0.7

Execute

18. Review the response. For each detected object an entry is returned in the predictions list, identifying the object class (e.g. person), the probability that the identified object class is correct (e.g. 0.9879012107849121), and the normalized coordinates where in the image

the object was detected. Probability is a numeric value between 0 and 1. The higher the value the more confident the model is about its prediction accuracy.

```
{
  "status": "ok",
  "predictions": [
    {
      "label_id": "88",
      "label": "teddy bear",
      "probability": 0.9896332025527954,
      "detection_box": [
        0.27832502126693726,
        0.5611844062805176,
        0.643224835395813,
        0.8432191610336304
      ]
    },
    {
      "label_id": "1",
      "label": "person",
      "probability": 0.9879012107849121,
      "detection_box": [
        0.24251864850521088,
        0.26926857233047485,
        0.6558930277824402,
        0.5768759846687317
      ]
    }
  ]
}
```

The threshold input value was used as a filter, eliminating objects from the result that the model is not confident enough about.

19. Lower (or increase) the threshold value from its default of 0.7 and invoke the endpoint again. The returned results might change depending on the value you've selected.

20. Close the browser tab (or window).

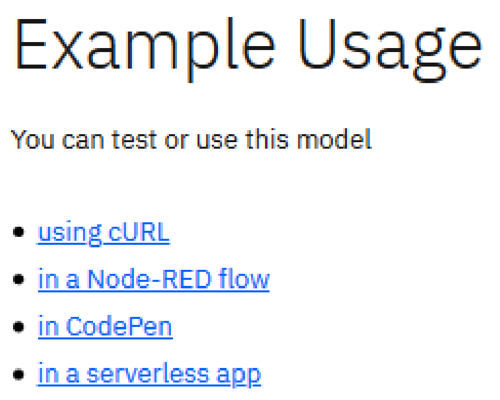
In the next section we'll briefly review how applications can consume the prediction endpoint to analyze the input.

Consume a MAX model-serving microservice

Since the microservice exposes a REST API, developers can implement applications and services that consume the prediction endpoint in any programming language.

1. On the Object Detector page (https://developer.ibm.com/exchanges/models/all/max-object-detector/) navigate to the **Example Usage** section.

2. To make it easy for developers to get started, each model comes with a set of examples. For the Object Detector model these samples include:

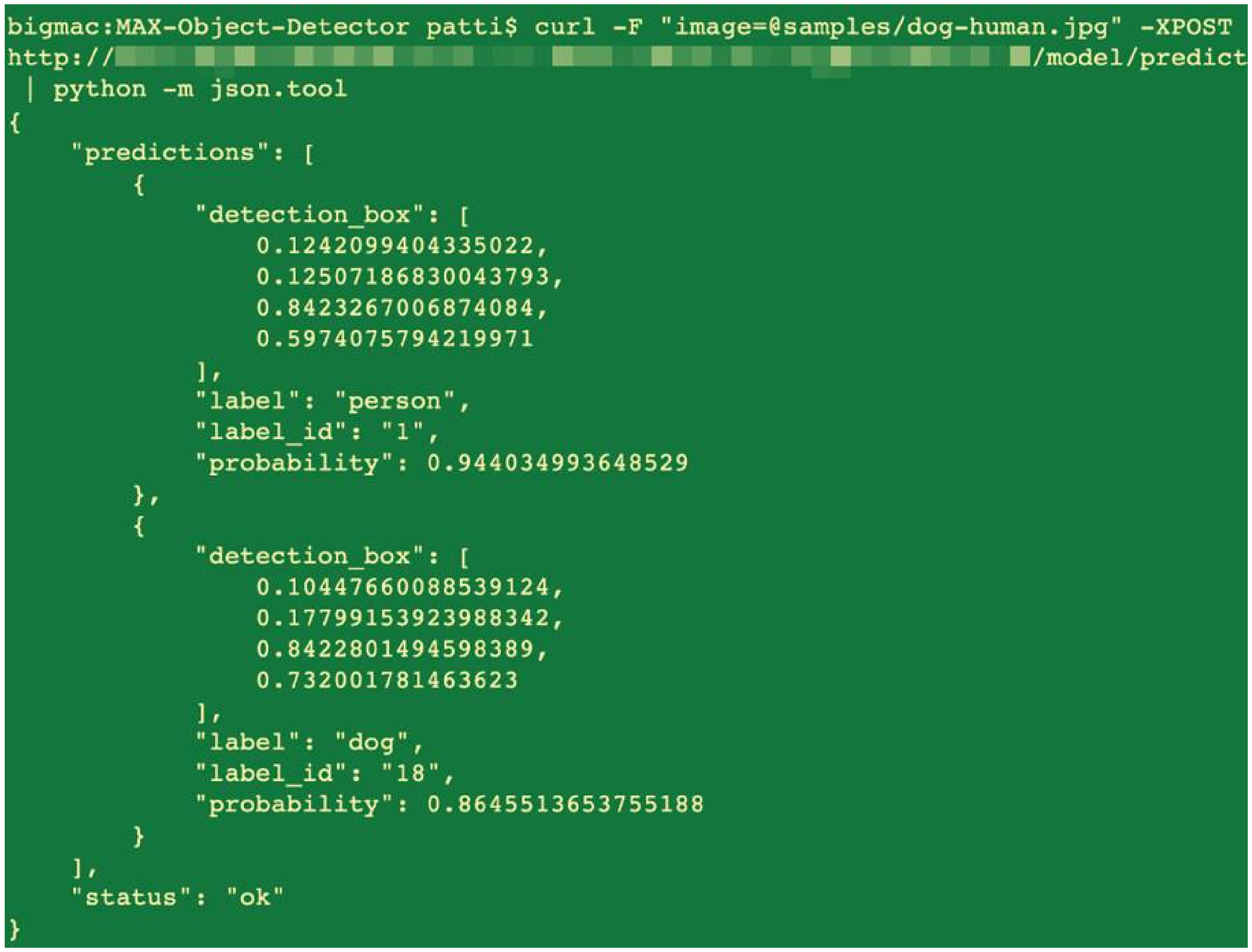


Example Usage

You can test or use this model

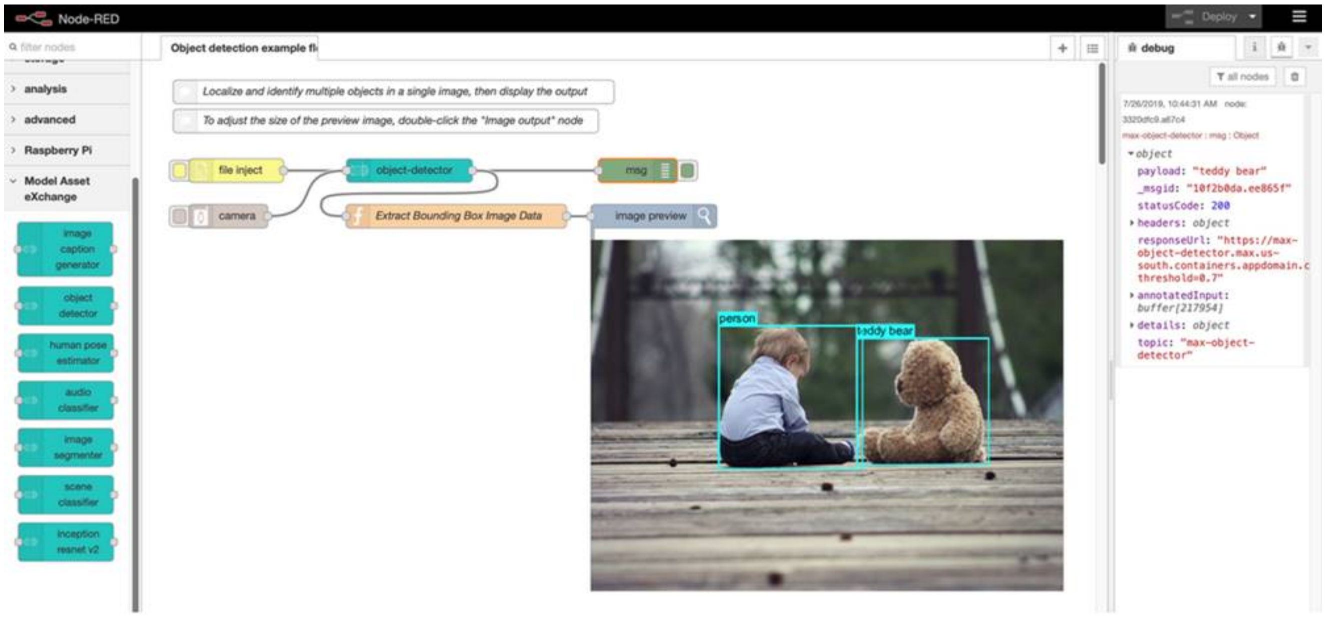
- using cURL
- in a Node-RED flow
- in CodePen
- in a serverless app

A command line example that uses curl (https://curl.haxx.se/), a popular open source tool to download and upload files from the web.



```
bigmac:MAX-Object-Detector patti$ curl -F "image=@samples/dog-human.jpg" -XPOST
http://[redacted]/model/predict
 | python -m json.tool
{
    "predictions": [
        {
            "detection_box": [
                0.1242099404335022,
                0.12507186830043793,
                0.8423267006874084,
                0.5974075794219971
            ],
            "label": "person",
            "label_id": "1",
            "probability": 0.944034993648529
        },
        {
            "detection_box": [
                0.10447660088539124,
                0.17799153923988342,
                0.8422801494598389,
                0.732001781463623
            ],
            "label": "dog",
            "label_id": "18",
            "probability": 0.8645513653755188
        }
    ],
    "status": "ok"
}
```

Node-RED flows that illustrate how to incorporate object detection into Internet of Things (IOT) applications.

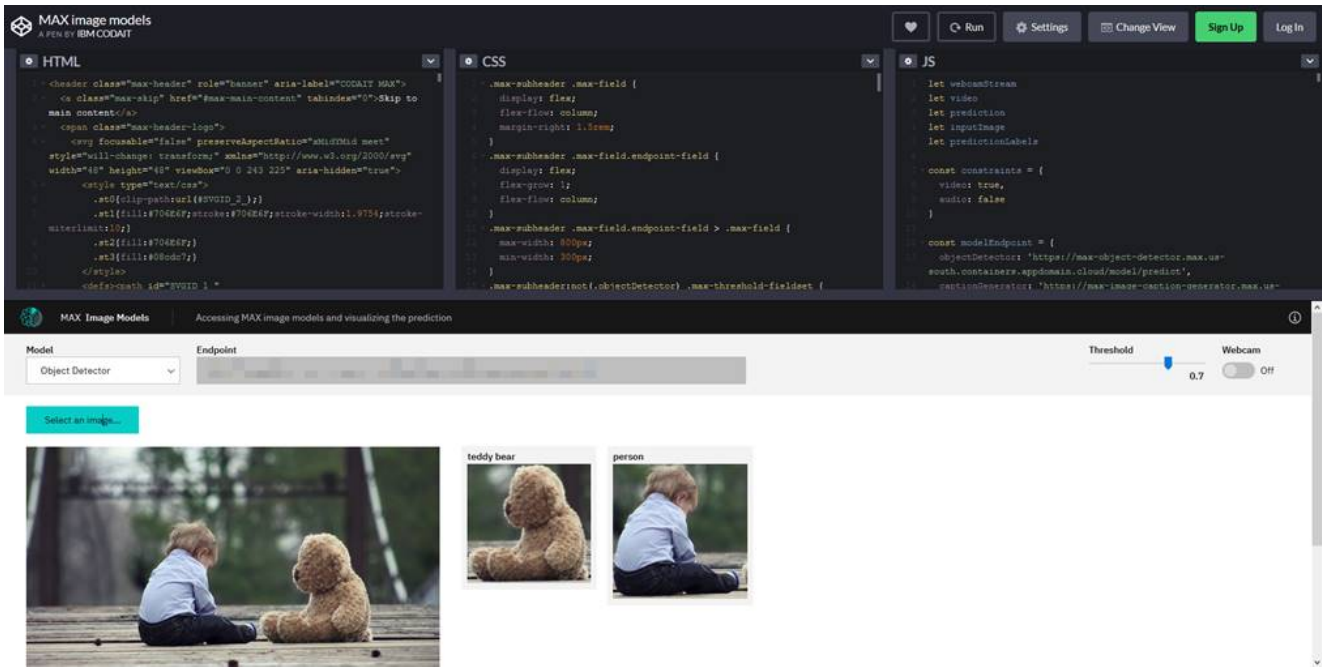


Localize and identify multiple objects in a single image, then display the output

To adjust the size of the preview image, double-click the "Image output" node

```
7/26/2019, 10:44:31 AM  node:
3320dfc9.a87c4
max-object-detector : msg : Object
▾object
  payload: "teddy bear"
  _msgid: "10f2b0da.ee865f"
  statusCode: 200
 ▸headers: object
  responseUrl: "https://max-
  object-detector.max.us-
  south.containers.appdomain.c
  threshold=0.7"
 ▸annotatedInput:
  buffer[217954]
 ▸details: object
  topic: "max-object-
  detector"
```

JavaScript code pens that illustrate how to utilize object detection in a web browser.



```
<header class="max-header" role="banner" aria-label="CODAIT MAX">
  <a class="max-skip" href="#max-main-content" tabindex="0">Skip to
main content</a>
  <span class="max-header-logo">
    <svg focusable="false" preserveAspectRatio="xMidYMid meet"
style="will-change: transform;" xmlns="http://www.w3.org/2000/svg"
width="48" height="48" viewBox="0 0 243 225" aria-hidden="true">
      <style type="text/css">
        .st0{clip-path:url(#SVGID_2_);}
        .st1{fill:#706E6F;stroke:#706E6F;stroke-width:1.9754;stroke-
miterlimit:10;}
        .st2{fill:#706E6F;}
        .st3{fill:#08cdc7;}
      </style>
      <defs><path id="SVGID_1_"
```

```
.max-subheader .max-field {
  display: flex;
  flex-flow: column;
  margin-right: 1.5rem;
}
.max-subheader .max-field.endpoint-field {
  display: flex;
  flex-grow: 1;
  flex-flow: column;
}
.max-subheader .max-field.endpoint-field > .max-field {
  max-width: 800px;
  min-width: 300px;
}
.max-subheader:not(.objectDetector) .max-threshold-fieldset {
```

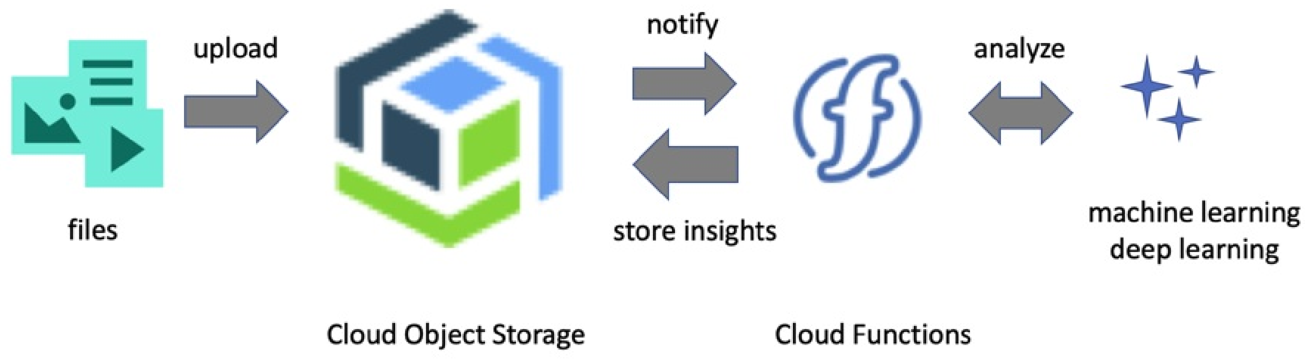
```
let webcamStream
let video
let prediction
let inputImage
let predictionLabels

const constraints = {
  video: true,
  audio: false
}

const modelEndpoint = {
  objectDetector: 'https://max-object-detector.max.us-
south.containers.appdomain.cloud/model/predict',
  captionGenerator: 'https://max-image-caption-generator.max.us-
```

Model: Object Detector
Endpoint: [redacted]
Threshold: 0.7
Webcam: Off

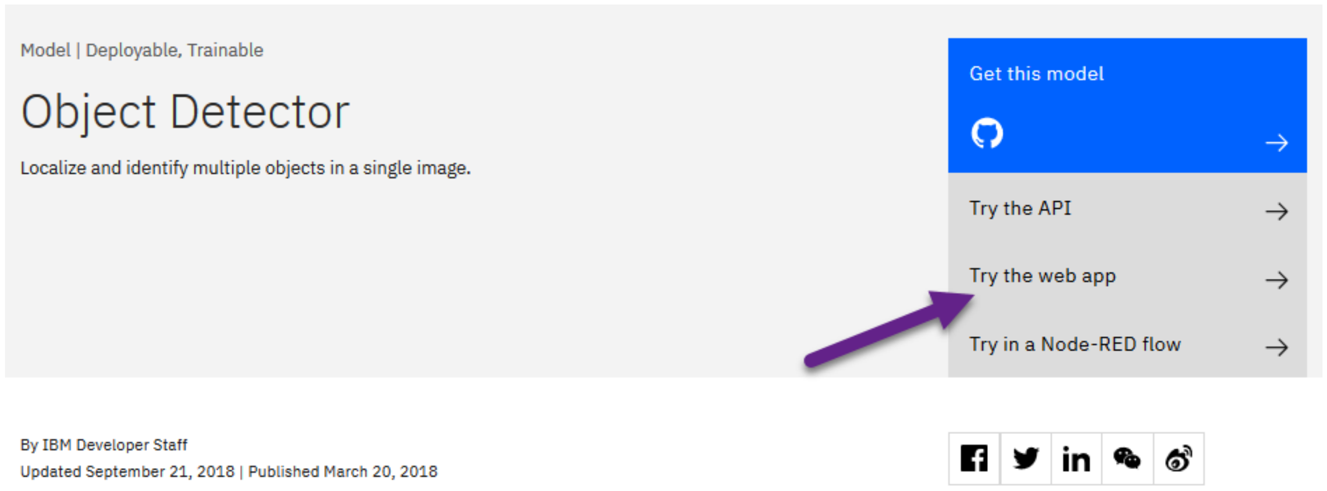
A tutorial that outlines how to build a serverless application using IBM Cloud Functions, which is a Functions as a Service platform based on Apache OpenWhisk.



Some models, like the Object Detector, also come with a sample application that you can try out without having to install anything.

3. On the Object Detector page (https://developer.ibm.com/exchanges/models/all/max-object-detector/) scroll to the top.

4. Click **Try the web app** to launch the demo application in your web browser.



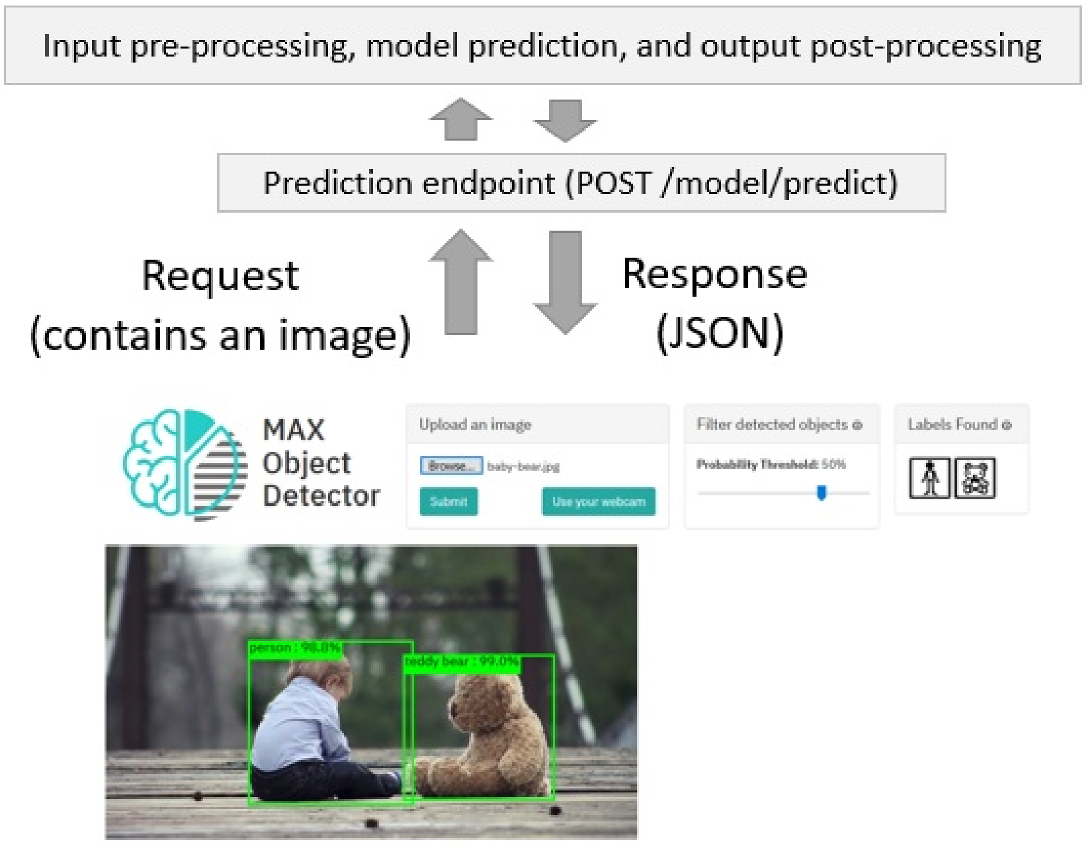
Model | Deployable, Trainable

Object Detector

Localize and identify multiple objects in a single image.

By IBM Developer Staff
Updated September 21, 2018 | Published March 20, 2018

The demo web application uploads an image (or takes a picture using the web cam), sends a request to an Object Detector microservice and visualizes the response by drawing bounding boxes around detected objects and attaching a label.


Input pre-processing, model prediction, and output post-processing
Prediction endpoint (POST /model/predict)
Request (contains an image)
Response (JSON)
MAX Object Detector
Upload an image
Browse...
baby-bear.jpg
Submit
Use your webcam
Filter detected objects
Probability Threshold: 50%
Labels Found
person : 98.8%
teddy bear : 99.0%

5. Upload an image (or take a picture using the web camera) and inspect the visualized results.

6. Change the filter conditions (e.g. lower the probability threshold) and observe how they impact the visualized results. Note the web application caches the response and applies the filter on the cached data. This was done for two reasons:

- It eliminates the need to re-process the image using the deep learning microservice (which significantly lowers the application's response time)
- It illustrates how client-side filtering can be applied to the results if the prediction endpoint doesn't support filtering by the desired criteria.

If you are interested in learning more about how the application was implemented, take a look at the code pattern: https://developer.ibm.com/patterns/create-a-web-app-to-interact-with-objects-detected-using-machine-learning/

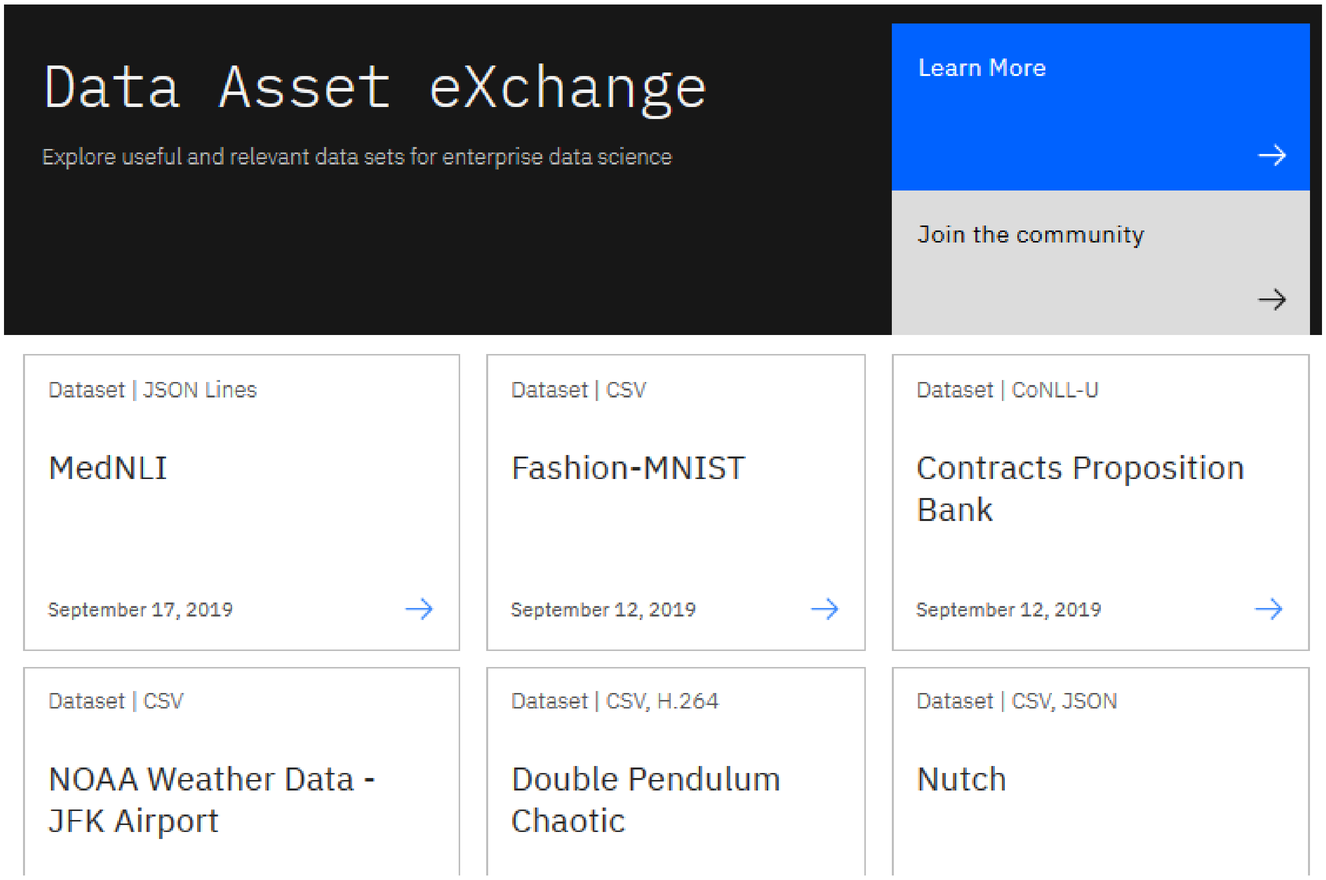
This concludes part 1 of this lab, which introduced the Model Asset Exchange.

Part 2: Explore deep learning data sets

The Data Asset Exchange is a curated collection of open data sets from IBM Research and 3rd parties that you can use to train models.

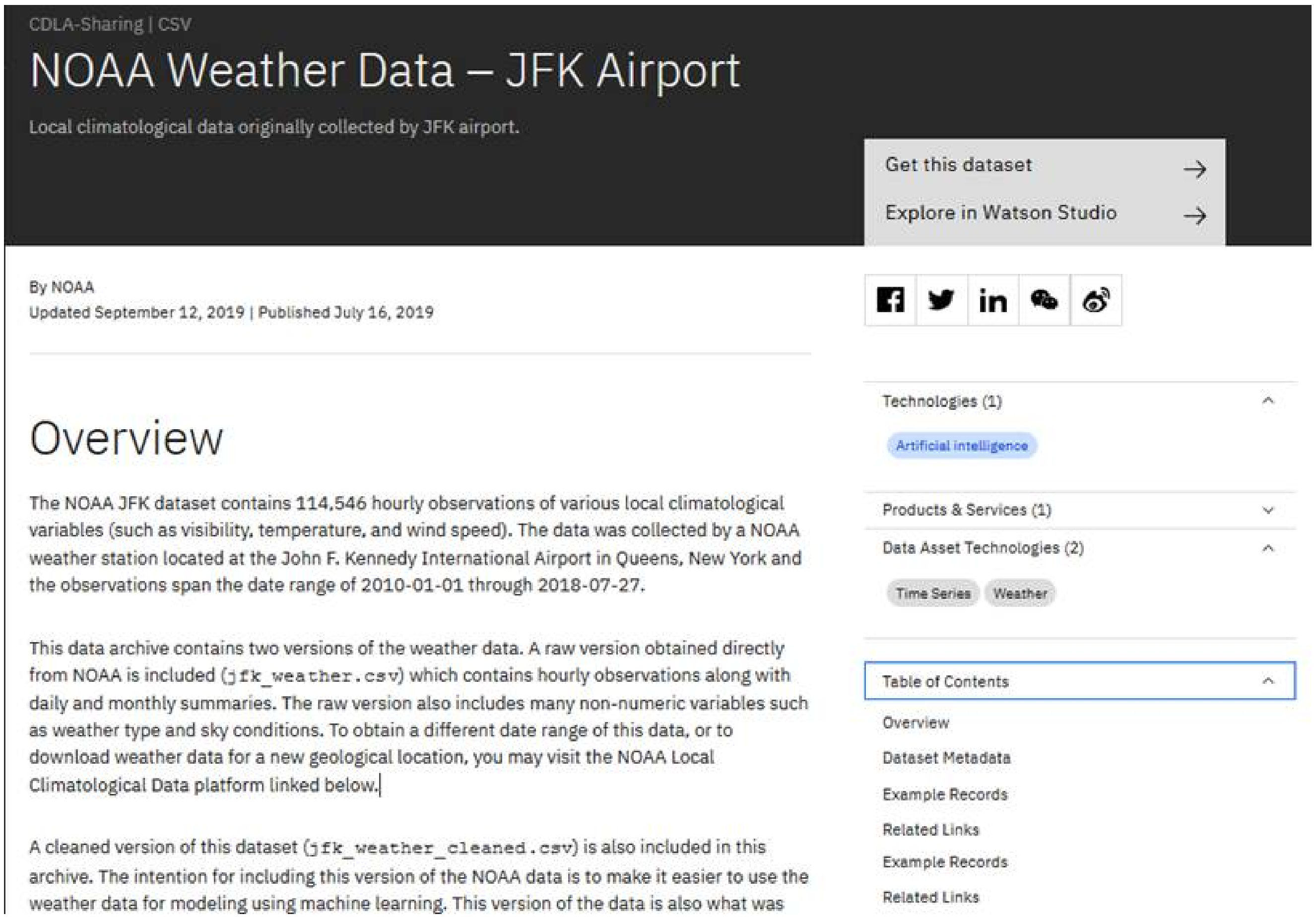
1. Open https://developer.ibm.com/ in your web browser.

2. From the main menu select **Open Source at IBM** > **Data Asset eXchange**. The DAX homepage is displayed.



The collection includes data sets from the Debater project (https://www.research.ibm.com/artificial-intelligence/project-debater/), data sets that can be used to train models to perform document layout analysis, natural language processing, time series analysis and more.

3. Open the NOAA Weather Data dataset (https://developer.ibm.com/exchanges/data/all/jfk-weather-data/), which contains data from a weather station at the John F. Kennedy Airport in New York spanning 8 years. This data set was used to train the weather forecaster model on MAX (https://developer.ibm.com/exchanges/models/all/max-weather-forecaster/).


CDLA-Sharing | CSV

NOAA Weather Data – JFK Airport

Local climatological data originally collected by JFK airport.

By NOAA
Updated September 12, 2019 | Published July 16, 2019

Overview

The NOAA JFK dataset contains 114,546 hourly observations of various local climatological variables (such as visibility, temperature, and wind speed). The data was collected by a NOAA weather station located at the John F. Kennedy International Airport in Queens, New York and the observations span the date range of 2010-01-01 through 2018-07-27.

This data archive contains two versions of the weather data. A raw version obtained directly from NOAA is included (`jfk_weather.csv`) which contains hourly observations along with daily and monthly summaries. The raw version also includes many non-numeric variables such as weather type and sky conditions. To obtain a different date range of this data, or to download weather data for a new geological location, you may visit the NOAA Local Climatological Data platform linked below.

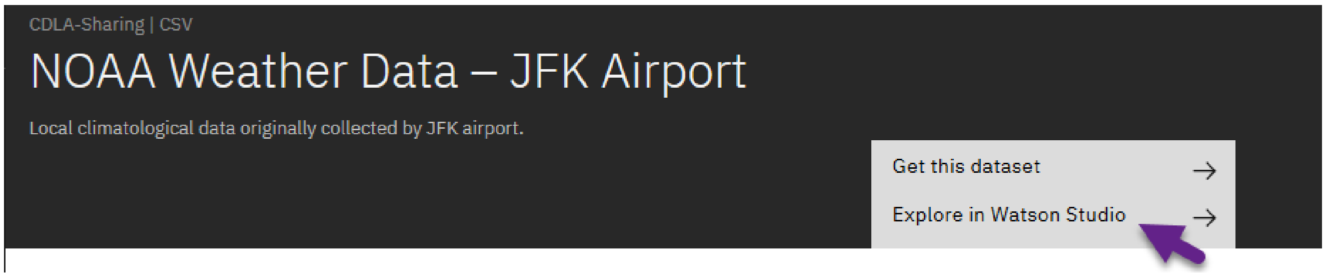
A cleaned version of this dataset (`jfk_weather_cleaned.csv`) is also included in this archive. The intention for including this version of the NOAA data is to make it easier to use the weather data for modeling using machine learning. This version of the data is also what was

You can download the data set using the **Get this dataset** link. Data sets are stored as compressed archives, which you can extract using any utility that supports the tar.gz format. If you are not familiar with this file format take a look at this short open source tutorial https://opensource.com/article/17/7/how-unzip-targz-file.

4. Inspect the data set's metadata.

- This data set is stored as tabular data and formatted as a comma separated value (CSV) file, which is a very popular basic data exchange format.
- The data set was published under the data science friendly CDLA-Sharing license (https://cdla.io/).
- The data set contains time-series data and can be used to predict weather trends.

5. Most data sets are complemented by Python notebooks that you can use to explore, pre-process, and analyze the data. You can access the notebook (or notebooks) by clicking the **Explore in Watson Studio** or **Try the notebook** link:

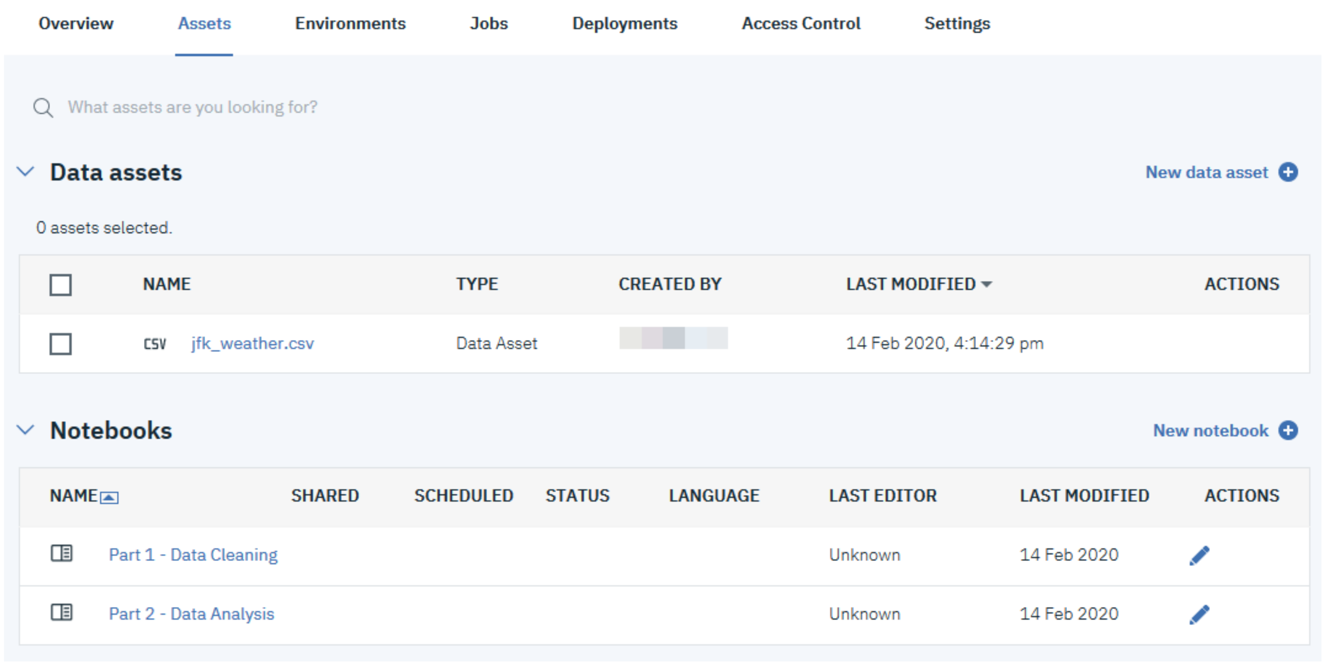

CDLA-Sharing | CSV

NOAA Weather Data – JFK Airport

Local climatological data originally collected by JFK airport.

The notebooks are hosted on Watson Studio, IBM's Data Science platform. Later in this course you'll learn more about Watson Studio, notebooks and how to run them.

6. [Optional] If you are already familiar with notebooks and Watson Studio feel free to open the link and import the project or notebook. The example below depicts the weather data set project assets, which include the raw data file and two notebooks.



Data assets

0 assets selected.

NAME	TYPE	CREATED BY	LAST MODIFIED	ACTIONS
CSV jfk_weather.csv	Data Asset		14 Feb 2020, 4:14:29 pm	

Notebooks

NAME	SHARED	SCHEDULED	STATUS	LANGUAGE	LAST EDITOR	LAST MODIFIED	ACTIONS
Part 1 - Data Cleaning					Unknown	14 Feb 2020	
Part 2 - Data Analysis					Unknown	14 Feb 2020	

This concludes part 2 of this lab, which introduced the Data Asset Exchange.