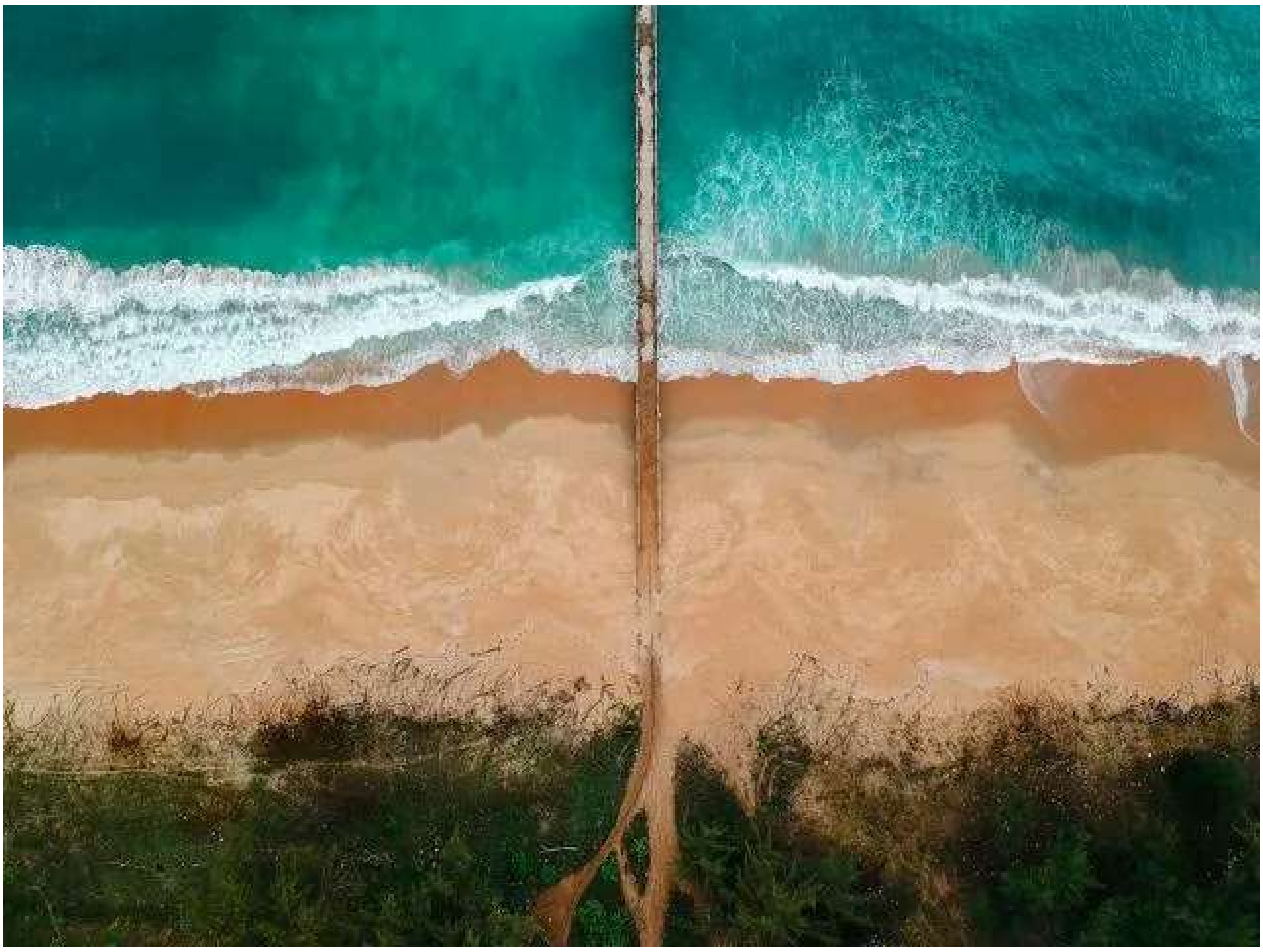
Post API Image Pipeline

Max Kleiner
Published in Nerd For Tech
4 min read · 5 days ago



The Object Detection API (**API**) provides fast and accurate image object recognition using advanced neural networks developed by machine learning experts and pretrained models.

First we send an input image by Post (**PostMultipartFormDataStream**), return a list of detected objects labels, confidence percentages and bounding boxes. Objects with confidence less than 0.3 (30%) are filtered out. The image we get with the first step of the pipeline:

```
function GEO_to_text_API2_randimage2(AURL, url_name, aApikey: string): string;
```

The Random Image API generates random images for all your placeholder and design needs. It Returns a random, base64-encoded image in JPEG format. Don't forget to set the Accept Header otherwise you have to decode with ALMimeBase64decodeStream. The (required) — header indicating the content type to accept in the result. Must be set to the following: `image/jpg`.

Second step is to post the image for object-detection.

https://api-ninjas.com/api/objectdetection

The (required) — must be an input image file. Must be either JPEG or PNG format and smaller than 2000 by 2000. Also the (required) — API Key associated with your account.

```
Procedure PyCodeObjectDetect(imgpath, aAPIKey: string); begin with TPythonEngir
```

```
Procedure PyCodeObjectDetect(imgpath, aAPIKey: string);
begin
  with TPythonEngine.Create(Nil) do begin
  //pythonhome:= 'C:\Users\User\AppData\Local\Programs\Python\Python312\';
  try
    loadDLL;
    ExecString('import requests');
    ExecStr('url= "https://api.api-ninjas.com/v1/objectdetection"');
    ExecStr('image_file_descriptor = open("'+imgpath+'", "rb")');
    ExecStr('headers= {"X-Api-Key": "'+aAPIKey+'"}');
    ExecStr('files = {"image": image_file_descriptor}  ');
    ExecStr('r=requests.post(url, headers=headers, files=files)');
    println(EvalStr('r.json()'));
  except
    raiseError;
  finally
    free;
  end;
 end;
end;
```

Behind the is the complicated configuration of a multipartformdata mechanism. On the other hand, `multipart/form-data` is the **encoding used when an HTML form has a file upload field.** When you make a POST request, you have to encode the data that forms the body of the request in some way.

- `application/x-www-form-urlencoded` is more or less the same as a query string on the end of the URL.
- **`multipart/form-data`** is significantly more complicated but it allows entire files to be included in the data.
- `multipart/form-data` : adds a few bytes of boundary overhead to the message, and must spend some time calculating it, but sends each byte in one byte.

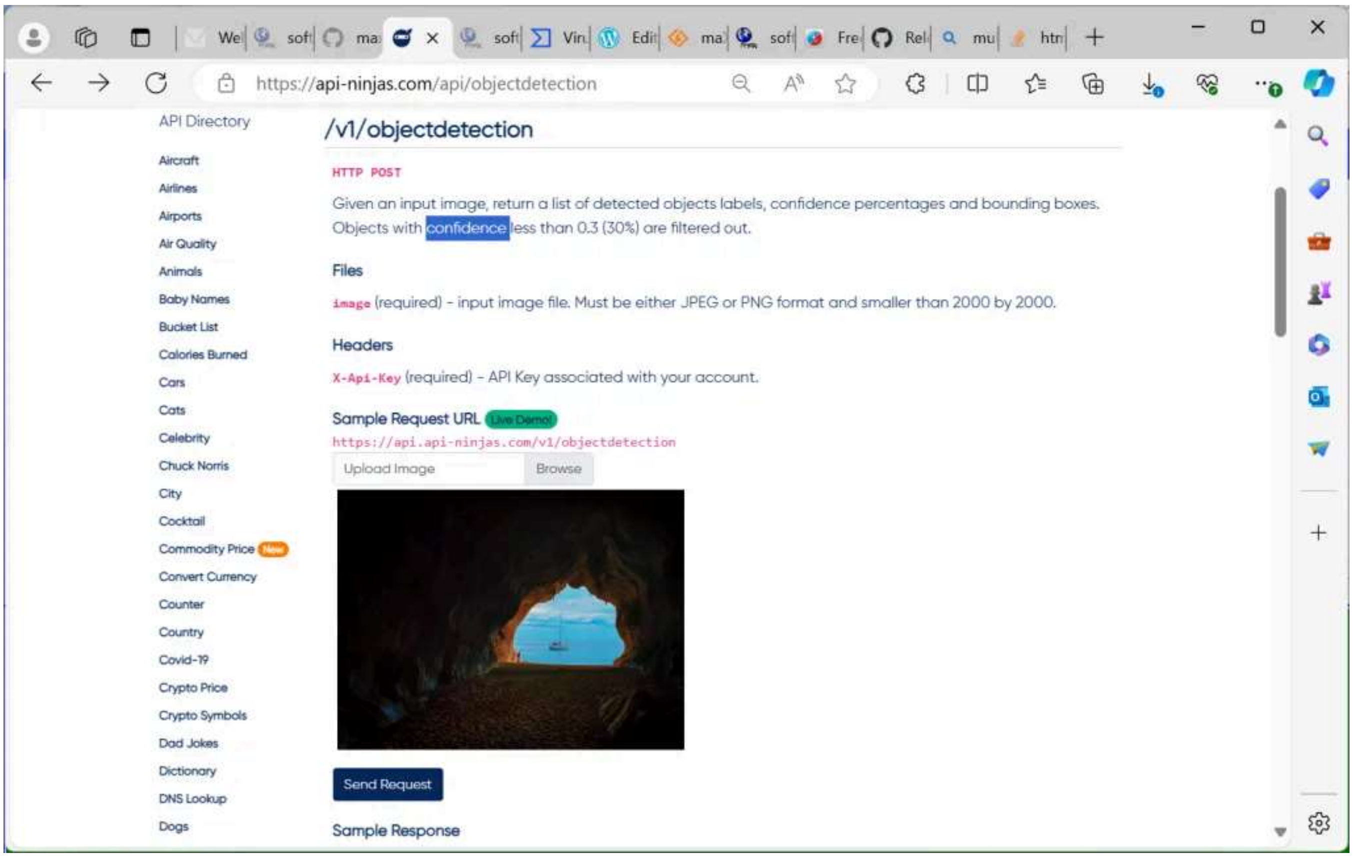
```
Procedure PostMultipartFormData(const aUrl:AnsiString; const aRequestFields: TA
```

```
Procedure PostMultipartFormData(const aUrl:AnsiString;
                                const aRequestFields: TALStrings;
                                const aRequestFiles: TALMultiPartFormDataConter
                                const aResponseContent: TStream;
                                const aResponseHeader: TALHTTPResponseHeader2;
                                const ARequestHeaderValues: TALNameValueArray =

https://code-maze.com/aspnetcore-multipart-form-data-in-httpclient/
```

Then we send the request and get the following JSON result of the detector (Sample Response):

> [{'label': ' **boat**', 'confidence': '0.52', 'bounding_box': {'x1': '308', 'y1': '179', 'x2': '527', 'y2': '328'}}, {'label': 'umbrella', 'confidence': '0.46', 'bounding_box': {'x1': '308', 'y1': '179', 'x2': '527', 'y2': '328'}}, {'label': 'boat', 'confidence': '0.34', 'bounding_box': {'x1': '385', 'y1': '277', 'x2': '425', 'y2': '295'}}, {'label': 'bed', 'confidence': '0.32', 'bounding_box': {'x1': '10', 'y1': '14', 'x2': '630', 'y2': '449'}}, {'label': 'boat', 'confidence': '0.31', 'bounding_box': {'x1': '384', 'y1': '285', 'x2': '426', 'y2': '298'}}, {'label': 'cat', 'confidence': '0.31', 'bounding_box': {'x1': '9', 'y1': '15', 'x2': '630', 'y2': '449'}}, {'label': ' **person** ', 'confidence': '0.3', 'bounding_box': {'x1': '8', 'y1': '11', 'x2': '633', 'y2': '444'}}]
>
> Yes we can see the boat and the small person, the umbrella maybe a false positive of the cave. A cat or a bed could be an imagination. We also have a false negative, the unseen sea or sky. Also a live demo from api_ninjas is available:


https://api-ninjas.com/api/objectdetection

API Directory

- Aircraft
- Airlines
- Airports
- Air Quality
- Animals
- Baby Names
- Bucket List
- Calories Burned
- Cars
- Cats
- Celebrity
- Chuck Norris
- City
- Cocktail
- Commodity Price New
- Convert Currency
- Counter
- Country
- Covid-19
- Crypto Price
- Crypto Symbols
- Dad Jokes
- Dictionary
- DNS Lookup
- Dogs

/v1/objectdetection

HTTP POST

Given an input image, return a list of detected objects labels, confidence percentages and bounding boxes. Objects with confidence less than 0.3 (30%) are filtered out.

Files

`image` (required) - input image file. Must be either JPEG or PNG format and smaller than 2000 by 2000.

Headers

`X-Api-Key` (required) - API Key associated with your account.

Sample Request URL Live Demo!

`https://api.api-ninjas.com/v1/objectdetection`

Sample Response

EKON 28

Max Kleiner 17/04/2024

Originally published at http://maxbox4.wordpress.com on April 17, 2024.

AI   Image Recognition   Object Detection

Edit profile

Written by Max Kleiner

27 Followers · Writer for Nerd For Tech

Max Kleiner's professional environment is in the areas of OOP, UML and coding - among other things as a trainer, developer and consultant.

More from Max Kleiner and Nerd For Tech

Max Kleiner in Nerd For Tech

OCR with a Neural Net

This API recognizes and reads a text embedded in pictures or photos. Image to Text API uses a neural net (LSTM) based OCR engine which is...