maXbox

Post API Image Pipeline

Posted on **April 17, 2024April 17, 2024** by **maxbox4**

The Object Detection 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): stri
var httpq: THttpConnectionWinInet;
    rets: TMemoryStream;
    heads: TStrings; iht: IHttpConnection;
    Decoder: TIdDecoderMIME;
begin
  httpq:= THttpConnectionWinInet.Create(true);
  rets:= TMemoryStream.create;
  heads:= TStringlist.create;
  try
    heads.add('X-Api-Key='+aAPIkey);
    heads.add('Accept=image/jpg');
    iht:= httpq.setHeaders(heads);
    httpq.Get(Format(AURL,[urlencode(url_name)]), rets);
    if httpq.getresponsecode=200 then begin
       rets.Position:= 0;
       //ALMimeBase64decodeStream(rets, rets2)
       rets.savetofile((exepath+'randimage0.jpg'));
       openfile(exepath+'randimage0.jpg');
     end
      else result:='Failed:'+
            itoa(Httpq.getresponsecode)+Httpq.GetResponseHeader('message')
  except
    writeln('EWI_HTTP: '+ExceptiontoString(exceptiontype,exceptionparam));
  finally
    httpq:= Nil;
    heads.Free;
    rets.Free;
  end;
end;                  //}
```

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 **`Accept`** (required) – header indicating the content type to accept in the result. Must be set to the following: `image/jpg` .

The Randimage we get

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

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

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

```
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 (https://a
    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.

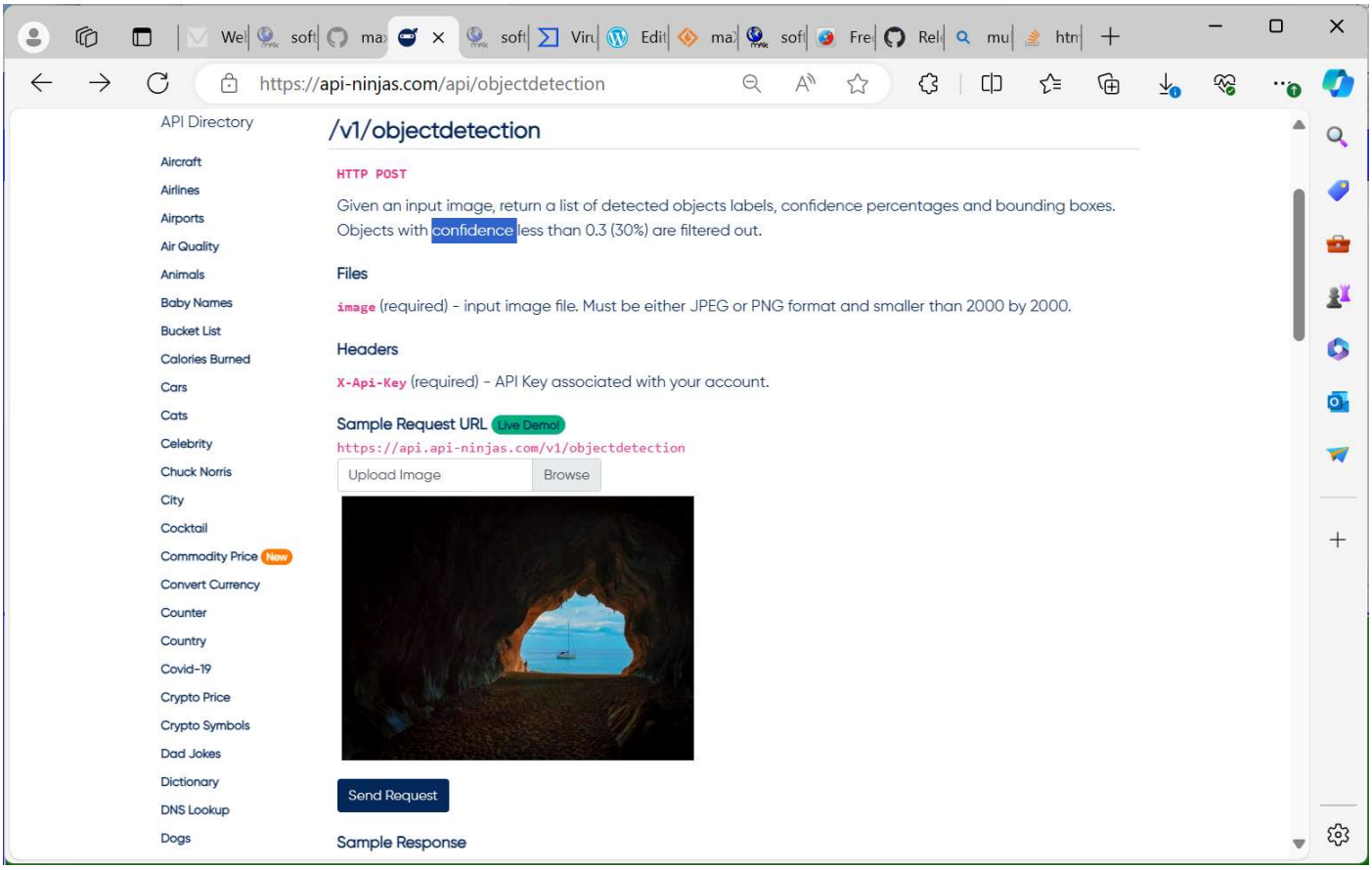
```
Procedure PostMultipartFormData(const aUrl:AnsiString;
                                const aRequestFields: TALStrings;
                                const aRequestFiles: TALMultiPartFormDataCon
                                const aResponseContent: TStream;
                                const aResponseHeader: TALHTTPResponseHeader
                                const ARequestHeaderValues: TALNameValueArra

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 is available:



Live API Demo

Posted in **Engineering**, **Machine Learning**, **maXbox**, **P4D**, **Python** Tagged **Coding**, **Detector**, **Machine Learning**, **Pascal**, **Python** **1 Comment**