AI & Machine Learning Products (https://cloud.google.com/products/machine-learning/)
Cloud Vision API (https://cloud.google.com/vision/)
Documentation (https://cloud.google.com/vision/docs/) Guides

Crop Hints Tutorial

Audience

The goal of this tutorial is to help you develop applications using the Cloud Vision API Crop Hints feature. It assumes you are familiar with basic programming constructs and techniques. However, even if you are a beginning programmer, you should be able to follow along and run this tutorial without difficulty, then use the Cloud Vision API reference documentation (https://cloud.google.com/vision/reference/rest/) to create basic applications.

This tutorial steps through a Vision API application, showing you how to make a call to the Vision API to use its Crop Hints feature.

Prerequisites

- Set up a Cloud Vision API project (https://cloud.google.com/vision/docs/getting-started#set_up_a_google_cloud_vision_api_project) in the Google Cloud Console.
- Set up your environment for using Application Default Credentials (https://cloud.google.com/vision/docs/setup#auth).

PYTHON

- Install Python (https://www.python.org/).
- Install pip (https://pip.pypa.io/en/latest/installing/).
- Install the Google Cloud Client Library (https://cloud.google.com/vision/docs/reference/libraries#installing_the_client_library).
- Install the Python Imaging Library (http://python-pillow.github.io/)

Overview

This tutorial walks you through a basic Vision API application that uses a **`Crop Hints`** *request*. You can provide the image to be processed either through a Cloud Storage URI (Cloud Storage bucket location) or embedded in the request. A successful **`Crop Hints`** *response* returns the coordinates for a bounding box cropped around the dominant object or face in the image.

Code listing

As you read the code, we recommend that you follow along by referring to the Cloud Vision API Python reference (https://googleapis.github.io/google-cloud-python/latest/vision/index.html).

vision/cloud-client/crop_hints/crop_hints.py (https://github.com/GoogleCloudPlatform/python-docs-samples/blob/master/vision/cloud-client/crop_hints/crop_hints.py)

'FORM/PYTHON-DOCS-SAMPLES/BLOB/MASTER/VISION/CLOUD-CLIENT/CROP_HINTS/CROP_HINTS.PY)

```
import argparse
import io

from google.cloud import vision
from google.cloud.vision import types
from PIL import Image, ImageDraw


def get_crop_hint(path):
    """Detect crop hints on a single image and return the first result."""
    client = vision.ImageAnnotatorClient()

    with io.open(path, 'rb') as image_file:
        content = image_file.read()

    image = types.Image(content=content)

    crop_hints_params = types.CropHintsParams(aspect_ratios=[1.77])
    image_context = types.ImageContext(crop_hints_params=crop_hints_params)

    response = client.crop_hints(image=image, image_context=image_context)
    hints = response.crop_hints_annotation.crop_hints
```

```
    # Get bounds for the first crop hint using an aspect ratio of 1.77.
    vertices = hints[0].bounding_poly.vertices

    return vertices


def draw_hint(image_file):
    """Draw a border around the image using the hints in the vector list."""
    vects = get_crop_hint(image_file)

    im = Image.open(image_file)
    draw = ImageDraw.Draw(im)
    draw.polygon([
        vects[0].x, vects[0].y,
        vects[1].x, vects[1].y,
        vects[2].x, vects[2].y,
        vects[3].x, vects[3].y], None, 'red')
    im.save('output-hint.jpg', 'JPEG')
    print('Saved new image to output-hint.jpg')


def crop_to_hint(image_file):
    """Crop the image using the hints in the vector list."""
    vects = get_crop_hint(image_file)

    im = Image.open(image_file)
    im2 = im.crop([vects[0].x, vects[0].y,
                   vects[2].x - 1, vects[2].y - 1])
    im2.save('output-crop.jpg', 'JPEG')
    print('Saved new image to output-crop.jpg')


if __name__ == '__main__':
    parser = argparse.ArgumentParser()
    parser.add_argument('image_file', help='The image you\'d like to crop.')
    parser.add_argument('mode', help='Set to "crop" or "draw".')
    args = parser.parse_args()

    if args.mode == 'crop':
        crop_to_hint(args.image_file)
    elif args.mode == 'draw':
        draw_hint(args.image_file)
```

A closer look

Importing libraries

vision/cloud-client/crop_hints/crop_hints.py (https://github.com/GoogleCloudPlatform/python-docs-samples/blob/master/vision/cloud-client/crop_hints/crop_hints.py)

FORM/PYTHON-DOCS-SAMPLES/BLOB/MASTER/VISION/CLOUD-CLIENT/CROP_HINTS/CROP_HINTS.PY)

```
import argparse
import io

from google.cloud import vision
from google.cloud.vision import types
from PIL import Image, ImageDraw
```

We import standard libraries:

- `argparse` to allow the application to accept input filenames as arguments
- `io` for file I/O

Other imports:

- The `ImageAnnotatorClient` class within the `google.cloud.vision` library for accessing the Vision API.
- The `types` module within the `google.cloud.vision` library for constructing requests
- The `Image` and `ImageDraw` modules from the `Python Imaging Library` (PIL). to draw a boundary box on the input image.

Running the application

vision/cloud-client/crop_hints/crop_hints.py (https://github.com/GoogleCloudPlatform/python-docs-samples/blob/master/vision/cloud-client/crop_hints/crop_hints.py)

FORM/PYTHON-DOCS-SAMPLES/BLOB/MASTER/VISION/CLOUD-CLIENT/CROP_HINTS/CROP_HINTS.PY)

```
parser = argparse.ArgumentParser()
parser.add_argument('image_file', help='The image you\'d like to crop.')
```

```
parser.add_argument('mode', help='Set to "crop" or "draw".')
args = parser.parse_args()

if args.mode == 'crop':
    crop_to_hint(args.image_file)
elif args.mode == 'draw':
    draw_hint(args.image_file)
```

Here, we simply parse the passed-in argument that specifies the local image filename, and pass it to a function to crop the image or draw the hint.

Authenticating to the API

Before communicating with the Vision API service, you must authenticate your service using previously acquired credentials. Within an application, the simplest way to obtain credentials is to use Application Default Credentials (https://cloud.google.com/vision/docs/setup#auth-env) (ADC). By default, the client library will attempt to obtain credentials from the `GOOGLE_APPLICATION_CREDENTIALS` environment variable, which should be set to point to your service account's JSON key file (see Set Up a Service Account (https://cloud.google.com/vision/docs/setup#keys) for more information.)

Getting crop hint annotations for the image

Now that the Vision client library is authenticated, we can access the service by calling the `crop_hints` method of the `ImageAnnotatorClient` instance. The aspect ratio for the *output* is specified in an `ImageContext` object; if multiple aspect ratios are passed in then multiple crop hints will be returned, one for each aspect ratio.

vision/cloud-client/crop_hints/crop_hints.py (https://github.com/GoogleCloudPlatform/python-docs-samples/blob/master/vision/cloud-client/crop_hints/crop_hints.py)

FORM/PYTHON-DOCS-SAMPLES/BLOB/MASTER/VISION/CLOUD-CLIENT/CROP_HINTS/CROP_HINTS.PY)

```
"""Detect crop hints on a single image and return the first result."""
client = vision.ImageAnnotatorClient()

with io.open(path, 'rb') as image_file:
    content = image_file.read()

image = types.Image(content=content)
```

```
crop_hints_params = types.CropHintsParams(aspect_ratios=[1.77])
image_context = types.ImageContext(crop_hints_params=crop_hints_params)

response = client.crop_hints(image=image, image_context=image_context)
hints = response.crop_hints_annotation.crop_hints

# Get bounds for the first crop hint using an aspect ratio of 1.77.
vertices = hints[0].bounding_poly.vertices
```

The client library encapsulates the details for requests and responses to the API. See the Vision API Reference (https://cloud.google.com/vision/reference/rest) for complete information on the structure of a request.

Using the response to crop or draw the hint's bounding box

Once the operation has been completed successfully, the API response will contain the bounding box coordinates of one or more `cropHints`. The `draw_hint` method draws lines around the CropHints bounding box, then writes the image to `output-hint.jpg`.

vision/cloud-client/crop_hints/crop_hints.py (https://github.com/GoogleCloudPlatform/python-docs-samples/blob/master/vision/cloud-client/crop_hints/crop_hints.py)

'FORM/PYTHON-DOCS-SAMPLES/BLOB/MASTER/VISION/CLOUD-CLIENT/CROP_HINTS/CROP_HINTS.PY)

```
vects = get_crop_hint(image_file)

im = Image.open(image_file)
draw = ImageDraw.Draw(im)
draw.polygon([
    vects[0].x, vects[0].y,
    vects[1].x, vects[1].y,
    vects[2].x, vects[2].y,
    vects[3].x, vects[3].y], None, 'red')
im.save('output-hint.jpg', 'JPEG')
print('Saved new image to output-hint.jpg')
```

The `crop_to_hint` method crops the image using the suggested crop hint.

vision/cloud-client/crop_hints/crop_hints.py (https://github.com/GoogleCloudPlatform/python-docs-samples/blob/master/vision/cloud-client/crop_hints/crop_hints.py)

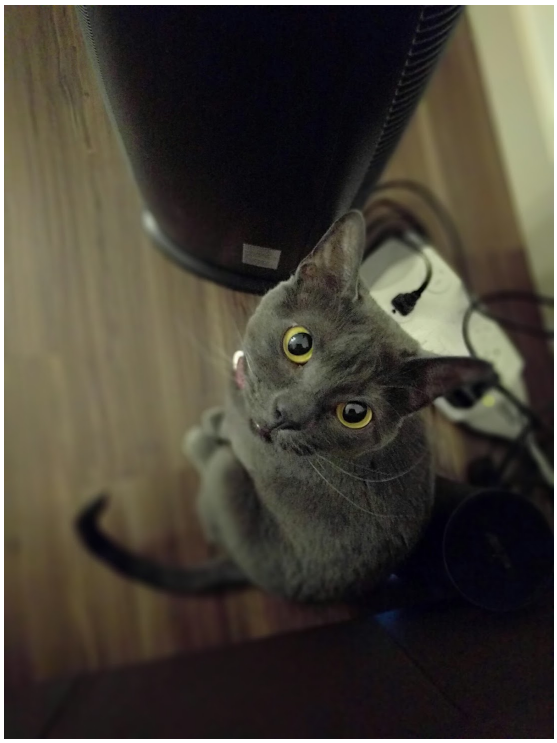
FORM/PYTHON-DOCS-SAMPLES/BLOB/MASTER/VISION/CLOUD-CLIENT/CROP_HINTS/CROP_HINTS.PY)

```
vects = get_crop_hint(image_file)

im = Image.open(image_file)
im2 = im.crop([vects[0].x, vects[0].y,
               vects[2].x - 1, vects[2].y - 1])
im2.save('output-crop.jpg', 'JPEG')
print('Saved new image to output-crop.jpg')
```

Running the application

To run the application, you can download this `cat.jpg` file (https://cloud.google.com/vision/docs/images/cat.jpg) (you may need to right-click the link), then pass the location where you downloaded the file on your local machine to the tutorial application (`crop_hints.py`).



Here is the Python command, followed by console output, which displays the JSON `cropHintsAnnotation` response. This response includes the coordinates of the `cropHints`

bounding box. We requested a crop area with a 1.77 width-to-height aspect ratio, and the returned top-left, bottom-right x,y coordinates of the crop rectangle are `0,336`, `1100,967`.

```
$ python crop_hints.py cat.jpeg crop
```

Note: Zero coordinate values omitted. When the API detects a coordinate ("x" or "y") value of 0, ***that coordinate is omitted in the JSON response***. Thus, a response with a bounding poly around the entire image would be
[{},{"x": 100},{"x": 100,"y": 100},{"y": 100}] for an image that is 100px by 100px. For more information, see the API Reference documentation (https://cloud.google.com/vision/docs/reference/rest/v1/images/annotate#boundingpoly).


```
{
 "responses": [
  {
   "cropHintsAnnotation": {
    "cropHints": [
     {
      "boundingPoly": {
       "vertices": [
        {
         "y": 336
        },
        {
         "x": 1100,
         "y": 336
        },
        {
         "x": 1100,
         "y": 967
        },
        {
         "y": 967
        }
       ]
      },
      "confidence": 0.79999995,
      "importanceFraction": 0.69
     }
    ]
   }
  }
```

```
  ]
}
```

And here is the cropped image.

Congratulations! You've run the Cloud Vision Crop Hints API to return the optimized bounding box coordinates around the dominant object detected in the image!

Last updated December 5, 2019.