Identiframe API

Release 0.0.1

Virtusa

Nov 02, 2020

Table of Contents

This is an API that performs various Image Analysis Tasks such as Object Detection, Face Detection and Text Analysis using Python3, OpenCV and Flask.

Overview on How to Run this API

1. Either install a Python IDE or create a Python virtual environment to install the packages required
2. Install packages required
3. Run the app as mentioned below.

Chapter 2

Setup procedure

1. **Configure project environment (Either A. Install Python IDE OR B. Create a Virtual Environment)**

    1. **Install Python IDE**

        - Open the Identiframe API Directory
        - Manually install packages to project OR type the command below on the activated virtual environment.

        ```
        pip install -r requirements.txt
        ```

    2. **Create a Python Virtual Environment**

        - Install virtualenv:

        ```
        sudo pip install virtualenv
        ```

        - Create virtialenv:

        ```
        virtualenv -p python3 <name of virtualenv>
        ```

        - Install requirements:

        ```
        pip install -r requirements.txt
        ```

2. Run app.py:

```
python app.py
```

Endpoints of the Identiframe API

1. Upload an Image
2. Select what image analysis technique you wanted to apply
3. Get the resultant image
4. Get the supporting output files such as json or text files
5. Download the supporting output files for your further usage or Analysis.

IdentiframeAPI main

Identiframe API .. moduleauthor:: Virtusa

app.**index**( )
Function that integrates html template with logic functions.

IdentiframeAPI controller

class imageanalysis.image_analysis_controller.**ImageAnalysisController** (*app_config*)

This class is the controller class that initializes all the application properties and controls the workflow between image_analysis_logic and app.

predict ( )

Controls and correlate all the functions required for predicting the objects/text/face from the image.

Returns Returns the output image in form of bytes.

Return type String

IdentiframeAPI ObjectDetectionLogic

class imageanalysis.image_analysis_logic.**ImageAnalysisLogic** (*IMG_UPLOAD_FOLDER*, *JSON_UPLOAD_FOLDER*, *YOLOV3_CONFIG*, *EXECUTION_PATH*)

This class is used to perform image analysis and detects object from the image.

darknet_model (*Width*, *Height*, *config*, *weights*, *image*, *classes*)

Function used to detect the objects from the given image using pretrained darknet model.

Parameters

- **Width** (*Int*) – Width of the input image.
- **Height** (*Int*) – Height of the input image.
- **config** (*File object*) – Configuration file that has model parameters.
- **weights** (*File object*) – pretrained weights.
- **image** (*Long*) – Input image.
- **classes** (*List*) – List of objects from COCO dataset.

Returns Width of the bounding box. Height (Int): Height of the bounding box. x (Int): Top-left x-coordinate of the bounding box. y (Int): Top-left y-coordinate of the bounding box. class_id (Int): Index of detected object from the list of classes. confidence (Float): Confidence score. Boxes (List): Contains width,height,Top-left x-coordinate,Top-left y-coordinate of each bounding box.

Return type Width (Int)

draw_prediction (*img*, *classes*, *class_id*, *confidence*, *x*, *y*, *x_plus_w*, *y_plus_h*, *COLORS*)

Function to draw bounding box on the detected object with class name.

Parameters

- **img** (*Long*) – Input image
- **classes** (*List*) – List of class of objects from COCO dataset.
- **class_id** (*Int*) – Index of detected object from the list of classes.
- **confidence** (*Float*) – Confidence score.
- **x** (*Int*) – Top-left x-coordinate of the bounding box.
- **y** (*Int*) – Top-left y-coordinate of the bounding box.
- **x_plus_w** (*int*) – Width of bounding box.
- **y_plus_h** (*int*) – Height of bounding box.
- **COLORS** (*Numpy Array*) – Randomly generated color for each bounding box.

get_output_layers (*net*)

Function to get the output layer names in the model architecture.

Parameters **net** (*Class*) – Pretrained darknet model.

Returns Returns the names of the output layers giving out predictions.

Return type List

method_download ( )

This function returns the response as a json file when user clicks the download button in the UI.

Returns Retuns json file containing detected objects and its respective coordinates and confidence score.

Return type Json Response

object_detection (*image, YOLOV3_CONFIG*)

Function that applies NMS and gives final output image and saves json object which has output values.

Parameters

- **image** (*Long*) – Input image.
- **YOLOV3_CONFIG** (*String*) – Path where config file and weights for pretrained object are stored.

Returns List of detected objects. base64img (Bytes): Output image. object_json (Json object): Json object that has all detected objects and its respective coordinates and confidence score.

Return type object_list (List)

upload_file ( )

Function which accepts the input image uploaded by user and passes it to object_detection function and

returns the output image returned by that function.

Returns Returns the output image.

Return type Image (Long)

Chapter 7

IdentiframeAPI FaceDetectionLogic

class imageanalysis.face_detection_logic.**DetectPerson**

This class is used to perform image analysis and detects faces from the image.

draw_prediction (*classes*, *class_id*, *confidence*, *x*, *y*, *x_plus_w*, *y_plus_h*, *COLORS*)

Function to draw bounding box on the detected object with class name.

Parameters

- **classes** (*List*) – List of class of objects from COCO dataset.
- **class_id** (*Int*) – Index of detected object from the list of classes.
- **confidence** (*Float*) – Confidence score.
- **x** (*Int*) – Top-left x-coordinate of the bounding box.
- **y** (*Int*) – Top-left y-coordinate of the bounding box.
- **x_plus_w** (*int*) – Width of bounding box.
- **y_plus_h** (*int*) – Height of bounding box.
- **COLORS** (*Numpy Array*) – Randomly generated color for each bounding box.

get_coordinates ( )

Once we have received the bounding boxes , confidence scores and classes, we will create a json list from the same and also draw the bounding boxes on the image and return the result.

Returns Returns the output image in form of bytes. Json Object : Json object that has all detected objects and its respective coordinates and confidence score.

Return type Long

get_image_details (*inp_image*)

This function reads the input image and creates blob for further processing. :param inp_image: Input image :type inp_image: Long

get_predictions ( )

This function gets the predicted bounding boxes, confidences and classes. This function uses the yolov3 weights and configuation.

initialize_yolov3 (*inp_image*)

This function is used to intialize the yolov3 weights and configurations.

Parameters **inp_image** (*Long*) – Input image

predict_faces (*inp_image*)

This function will predict if there are any faces in the image and return the bounding boxes , confidence scores and classes. This function uses the haarcascaden technique for predicting the faces from the image.

Parameters **inp_image** (*Long*) – Input image

train_model ( )

This function runs a forward pass to predict the objects present in the image.

Chapter 8

IdentiframeAPI TextDetectionLogic

class imageanalysis.text_detection_logic.**text_extract**

This class is used to perform image analysis and detects text from the image.

build_lines (*table_cells*)

This function builds table lines based on find table function

Parameters **table_cells** (*list*) – coordinates of text to form table

Returns coordinates of horizontal and vertical lines

Return type list

draw_boxes (*image*, *details*, *threshold_point*)

This function draws boxes around the predicted texts

Parameters

- **image** (*png*) – input out.png file
- **details** (*dict*) – coordinates of predicted texts
- **threshold_point** (*int*) – value for drawing box around the text

final_output (*in_file*, *execution_path*)

This function returns final output image and writes detected text to the text file.

Parameters **in_file** (*Long*) – Input Image

Returns output image.

Return type Long

find_table_in_boxes (*boxes*, *cell_threshold=10*, *min_columns=2*)

This function Identifies if any table structure is present

Parameters

- **boxes** (*list*) – Coordinates of text boxes
- **cell_threshold** (*int*) – threshold value of a cell. Defaults to 10.
- **min_columns** (*int*) – minimum column required for creating a table. Defaults to 2.

Returns table strcture coordinates

Return type list

find_text_boxes (*pre*, *min_text_height_limit=2*, *max_text_height_limit=70*)

This function is used for idenntifying the texts

Parameters
- **pre** (*image*) – Pre processed image
- **min_text_height_limit** (*int*) – Minimum text height. Defaults to 2.
- **max_text_height_limit** (*int*) – Maximum text height. Defaults to 70.

Returns coordinates of bounding boxes for texts

Return type list

format_text (*details*)

This function formats the text captured

Parameters **details** (*dict*) – dictonary of predicted texts from the image

Returns list of formatted texts

Return type list

parse_text (*threshold_img*)

This function predicts the text using Tesseract

Parameters **threshold_img** (*png*) – out.png file is taken as input

Returns returns dictonary of english keywords

Return type dict

pre_process_image (*img*, *save_in_file*, *morph_size=8, 8*)

This function is used for pre processing the image

Parameters
- **img** (*jpg/png*) – Input image file
- **save_in_file** (*png*) – Pre processed image is saved as pre.png
- **morph_size** (*tuple, optional*) – Morphological transformation parameters. Defaults to (8, 8).

Returns pre processed image file

Return type image

Chapter 9

Indices and tables

- *Index*
- *Module Index*
- *Search Page*

a

c

f

i

o

t

O

P

T

U