Real Time Gender and Age Detection with OpenCV

JUNWEN WANG

NOC 30 2023

Problem Description

- The problem we aim to address is the estimation of gender and age from a single image or video of a person's face using Deep Learning. This has significant implications in various domains, such as marketing, content personalization, and security.

Approaches

Computer Vision

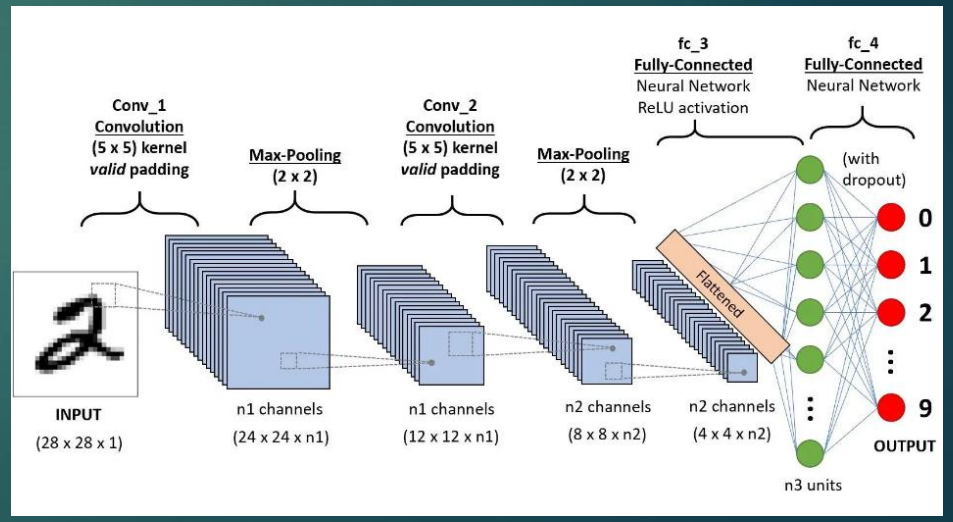
- **Computer Vision** is the field of study that enables computers to see and identify digital images and videos as a human would. The challenges it faces largely follow from the limited understanding of biological vision. Computer Vision involves acquiring, processing, analyzing, and understanding digital images to extract high-dimensional data from the real world in order to generate symbolic or numerical information which can then be used to make decisions.

OpenCV

- **OpenCV** is short for Open Source Computer Vision. It is an open-source Computer Vision and Machine Learning library. This library is capable of processing real-time image and video while also boasting analytical capabilities.

CNN

- A **Convolutional Neural Network** is a deep neural network (DNN) widely used for the purposes of image recognition. Also known as a ConvNet, a CNN has input and output layers, and multiple hidden layers, many of which are convolutional.

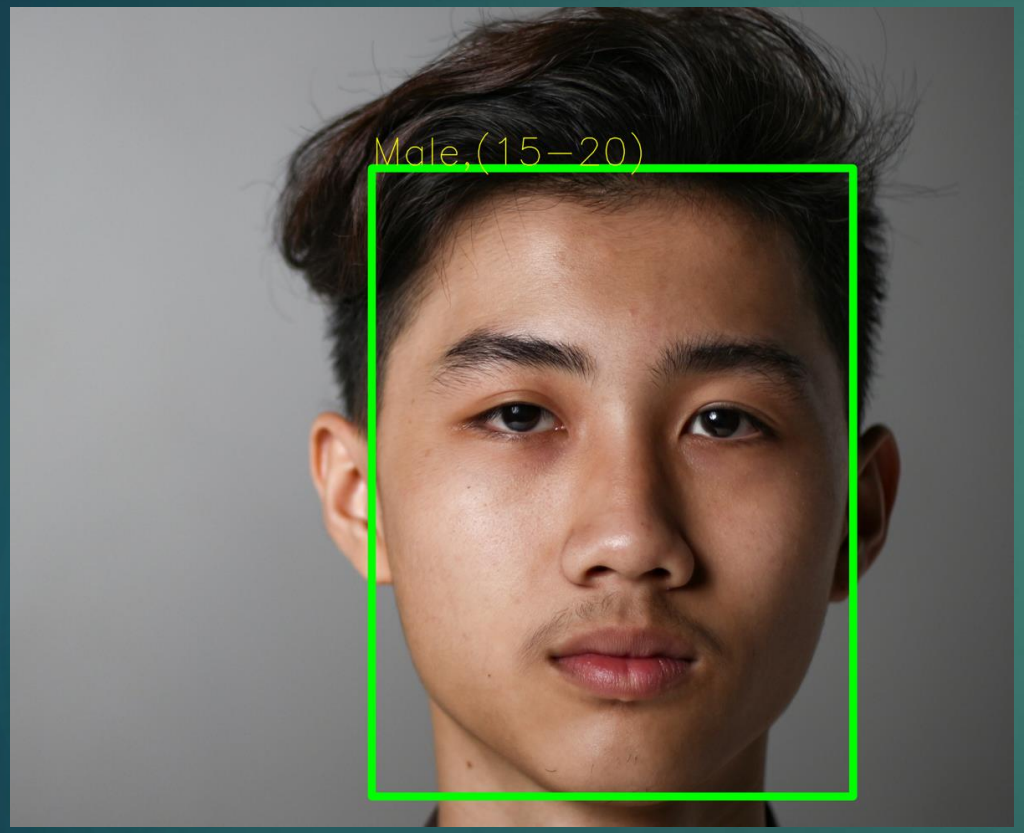


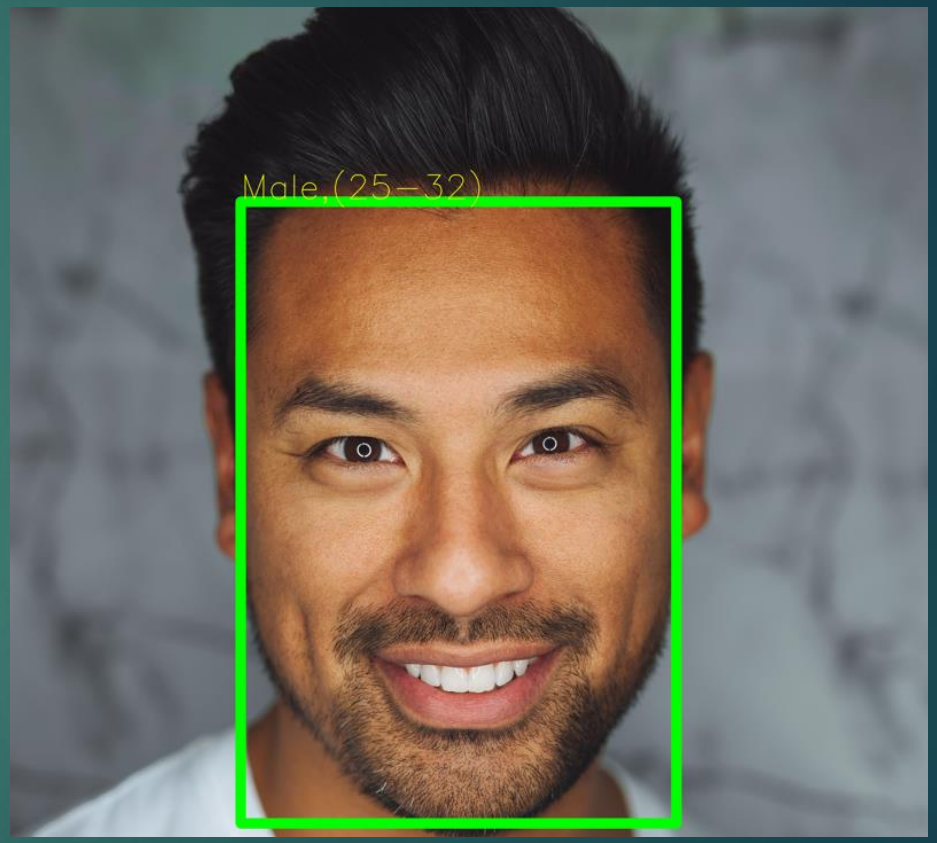
Procedure

- In the first part of this project, I will use the models by Tal Hassner and Gil Levi [1]to identify the gender and age of a person from a single image of a face at first. The predicted gender may be one of ‘Male’ and ‘Female’, and the predicted age may be one of the following ranges- (0 – 2), (4 – 6), (8 – 12), (15 – 20), (25 – 32), (38 – 43), (48 – 53), (over 60) (8 nodes in the final softmax layer). (26580 photos in eight age ranges)

- Then I will combine it with capturing real time video, thus it will make real time analysis of age and gender based on our webcam

Results





Results

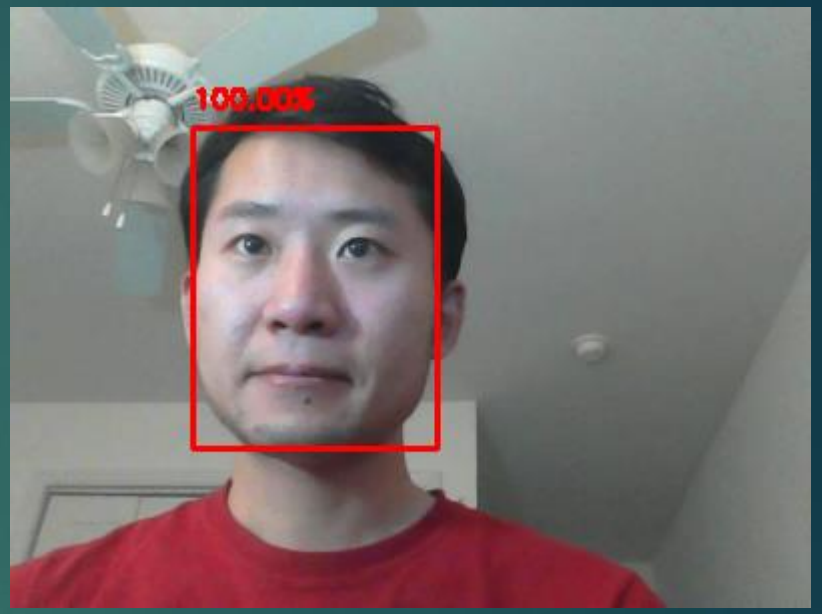
```
def take_photo(filename='photo.jpg', quality=0.8):
  js = Javascript('''
    async function takePhoto(quality) {
      const div = document.createElement('div');
      const capture = document.createElement('button');
      capture.textContent = 'Capture';
      div.appendChild(capture);

      const video = document.createElement('video');
      video.style.display = 'block';
      const stream = await navigator.mediaDevices.getUserMedia({video: true});

      document.body.appendChild(div);
      div.appendChild(video);
      video.srcObject = stream;
      await video.play();

      // Resize the output to fit the video element.
      google.colab.output.setIframeHeight(document.documentElement.scrollHeight, true);

      // Wait for Capture to be clicked.
      await new Promise((resolve) => capture.onclick = resolve);

      const canvas = document.createElement('canvas');
      canvas.width = video.videoWidth;
      canvas.height = video.videoHeight;
      canvas.getContext('2d').drawImage(video, 0, 0);
      stream.getVideoTracks()[0].stop();
      div.remove();
      return canvas.toDataURL('image/jpeg', quality);
    }
    ''')
  display(js)
  data = eval_js('takePhoto({})'.format(quality))
  binary = b64decode(data.split(',')[1])
  with open(filename, 'wb') as f:
    f.write(binary)
  return filename
```

```
image_file = take_photo()

image = cv2.imread(image_file)

# resize it to have a maximum width of 400 pixels
image = imutils.resize(image, width=400)
(h, w) = image.shape[:2]
print(w,h)
cv2_imshow(image)
```

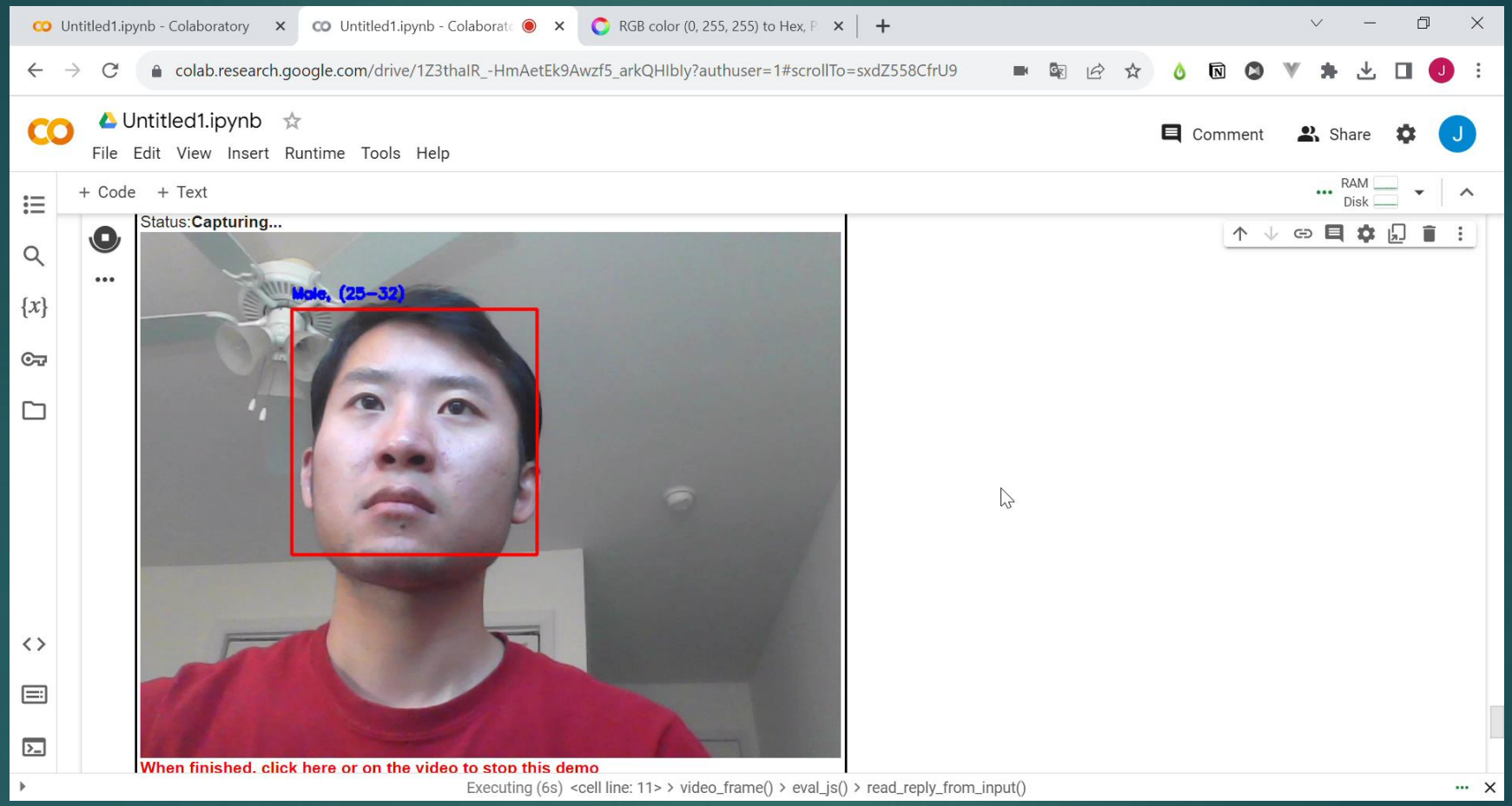
```
400 300
```

Results

```python
# function to convert the JavaScript object into an OpenCV image
def js_to_image(js_reply):
  """
  Params:
          js_reply: JavaScript object containing image from webcam
  Returns:
          img: OpenCV BGR image
  """
  # decode base64 image
  image_bytes = b64decode(js_reply.split(',')[1])
  # convert bytes to numpy array
  jpg_as_np = np.frombuffer(image_bytes, dtype=np.uint8)
  # decode numpy array into OpenCV BGR image
  img = cv2.imdecode(jpg_as_np, flags=1)

  return img

# function to convert OpenCV Rectangle bounding box image into base64 byte string to be overlayed on video stream
def bbox_to_bytes(bbox_array):
  """
  Params:
          bbox_array: Numpy array (pixels) containing rectangle to overlay on video stream.
  Returns:
        bytes: Base64 image byte string
  """
  # convert array into PIL image
  bbox_PIL = PIL.Image.fromarray(bbox_array, 'RGBA')
  iobuf = io.BytesIO()
  # format bbox into png for return
  bbox_PIL.save(iobuf, format='png')
  # format return string
  bbox_bytes = '{}'.format((str(b64encode(iobuf.getvalue()), 'utf-8')))

  return bbox_bytes
```



Results


Untitled1.ipynb - Colaboratory
Untitled1.ipynb - Colaborat
RGB color (0, 255, 255) to Hex,
colab.research.google.com/drive/1Z3thaIR_-HmAetEk9Awzf5_arkQHIbIy?authuser=1#scrollTo=sxdZ558CfrU9
Untitled1.ipynb
File Edit View Insert Runtime Tools Help
Comment
Share
+ Code + Text
RAM
Disk
Status:Capturing...
Male, (25-32)
When finished, click here or on the video to stop this demo
Executing (6s) <cell line: 11> > video_frame() > eval_js() > read_reply_from_input()

Lesson Learned

- In this project I'm trying to recognize gender and age with real time webcam. My results shows promising for this problems in some extent.
- However, we still can't recognize the exact age of people, the age range is still too large, due to many reasons.

Reference:

- Tal Hassner, Shai Harel*, Eran Paz* and Roee Enbar, *Effective Face Frontalization in Unconstrained Images,* IEEE Conf. on Computer Vision and Pattern Recognition (CVPR), Boston, June 2015
- Gil Levi and Tal Hassner, *Age and Gender Classification Using Convolutional Neural Networks,* IEEE Workshop on Analysis and Modeling of Faces and Gestures (AMFG), at the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR), Boston, June 2015
- DATASET: https://www.kaggle.com/datasets/ttungl/adience-benchmark-gender-and-age-classification
- https://data.vision.ee.ethz.ch/cvl/rrothe/imdb-wiki/

Thank You