human Body Segmentation using TensorFlow.js

Abstact   
Body segmentation is a deep learning task that segments and highlights the boundaries between different sections of the body, such as the torso, lower arm, upper arm, thigh, and lower leg. Study of human motion analysis is related to several research areas of computer vision such as the motion capture, detection, tracking and segmentation of people. In this paper, In this paper, we’ll explore human body image segmentation.

we will also t cover how to implement human body segmentation using TensorFlow.js. TensorFlow.js is very useful and powerful library and BodyPix is a body segmentation model built on TensorFlow. The model uses a pre-trained neural network to segment a human body from a frame.

# Introduction to human body segmentation

In digital image processing and computer vision, image segmentation is the process of partitioning a digital image into multiple image segments, also known as image regions or image objects (sets of pixels). Human body segmentation in human images is a very important step in many

computer vision tasks, such as image processing, video tracking, pose estimation,

content-based image retrieval, pedestrian detection, action understanding, etc.

The goal of segmentation is to simplify and/or change the representation of an image into something that is more meaningful and easier to analyze. Image segmentation is typically used to locate objects and boundaries (lines, curves, etc.) in images. More precisely, image segmentation is the process of assigning a label to every pixel in an image such that pixels with the same label share certain characteristics.



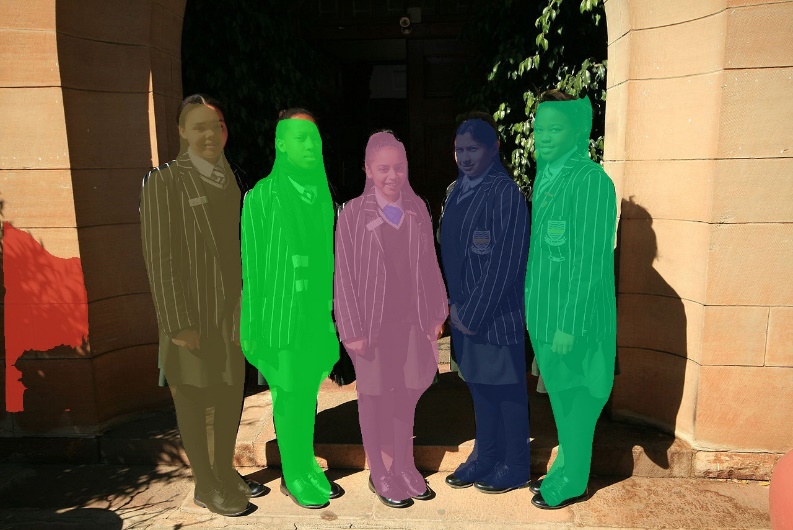
Two major types of body Segmentation:

## Semantic Segmentation:

Objects classified with the same pixel values are segmented with the same colormaps.

## Instance Segmentation:

It differs from semantic segmentation because different instances of the same object are segmented with different color maps.



# Image Segmentation with PixelLib:

It is a library built for an easy implementation of Image Segmentation in real life problems. PixelLib is a flexible library that can be integrated into software solutions that require the application of Image Segmentation.

import pixellib

from pixellib.semantic import semantic\_segmentation

segment\_image = semantic\_segmentation()

segment\_image.load\_pascalvoc\_model("deeplabv3\_xception\_tf\_dim\_ordering\_tf\_kernels.h5")

segment\_image.segmentAsPascalvoc("path\_to\_image", output\_image\_name = "path\_to\_output\_image")

We shall observe each line of code:

import pixellib

from pixellib.semantic import semantic\_segmentation

segment\_image = semantic\_segmentation()

The class for performing semantic segmentation is imported from pixelLib and we created an instance of the class.

Then we loaded the function to perform segmentation on an image. The function takes two parameters:

* path\_to\_image: this is the path to the image to be segmented.
* output\_image\_name: this is the path to save the segmented image. It will be saved in your current working directory.



# Popular deep learning models:

Most segmentation models are measured against by their mean Intersection over Union (mIoU). Intersection over union is an evaluation metric that measures the accuracy of an object detector on a dataset by comparing the location of the predicted bounding box (the one the detector came up with) with the one of the ground-truth bounding box (the hand-labeled one). It’s scored from 0 to 1; the closer to 1, the more accurate the model.

Some of the most popular ones are Mask R-CNN (simple to train and easy to use), CDCL (Cross-Domain Complementary Learning), SCHO (good for single, multiple, and video segmentation), and WSHP (which transfers human body parts to raw images by exploiting their anatomical similarity).

# Body Segmentation with TensorFlow JS:

TensorFlow uses BODYPIX model. The BodyPix model is trained to do this for a person and twenty-four body parts (parts such as the left hand, front right lower leg, or back torso). In other words, BodyPix can classify the pixels of an image into two categories:

1) pixels that represent a person and

2) pixels that represent background.

It can further classify pixels representing a person into any one of twenty-four body parts.

# Why would we want to do this in the browser?

Similar to the case of PoseNet, real-time person segmentation was only possible to do before with specialized hardware or hard-to-install software with steep system requirements. Instead both BodyPix and PoseNet can be used without installation and just a few lines of code. We don’t need any specialized lenses to use these models — they work with any basic webcam or mobile camera. And finally users can access these applications by just opening a url. Since all computing is done on device, the data stays private.

# Understanding BodyPix of TensorflowJS:

## Importing the TensorFlow.js and BodyPix Libraries:

The library can be installed with: npm install @tensorflow-models/body-pix and then imported using es6 modules:

import \* as bodyPix from '@tensorflow-models/body-pix';

async function loadAndUseBodyPix() {

const net = await bodyPix.load();

// BodyPix model loaded

}

Or via a bundle in the page, with nothing to install:

<html>

<body>

<!-- Load TensorFlow.js -->

<script src="https://cdn.jsdelivr.net/npm/@tensorflow/tfjs@0.13.3"></script>

<!-- Load BodyPix -->

<script src="https://cdn.jsdelivr.net/npm/@tensorflow-models/body-pix"></script>

bodypix.load().then(function(net) {

// BodyPix model loaded

});

</script>

</body>

</html>

## Person Segmentation:

At a basic level, person segmentation segments an image into pixels that are part of a person and those that are not. Under the hood, after an image is fed through the model, it gets converted into a two-dimensional image with float values between 0 and 1 at each pixel indicating the probability that the person exists in that pixel. A value called the “segmentation threshold” represents the minimum value a pixel’s score must have to be considered part of a person. Using the segmentation threshold, those 0–1 float values become binary 0s or 1s (ie. if the threshold is 0.5, any values over 0.5 are converted to 1s and any values below 0.5 are converted to 0s).

We call the API method estimatePersonSegmentation to perform person segmentation on an image or video; this short code block shows how to use it:

const imageElement = document.getElementById('image');

// load the BodyPix model from a checkpoint

const net = await bodyPix.load();

// arguments for estimating person segmentation.

const outputStride = 16;

const segmentationThreshold = 0.5;

const personSegmentation = await net.estimatePersonSegmentation(imageElement, outputStride, segmentationThreshold);

## Drawing the Person Segmentation Output:

Another benefit of BodyPix being in the browser is that we have access to web APIs such as Canvas Compositing. We use this to mask or replace parts of images using the outputs from BodyPix. We’ve provided utility functions that wrap this functionality to get you started:

toMaskImageData takes the output from estimating person segmentation and generates a transparent image that, depending on the argument maskBackground, will be opaque either where the person or background is. This can then be drawn as a mask on top of the original image using the method drawMask:

const imageElement = document.getElementById('image');

const net = await bodyPix.load();

const segmentation = await net.estimatePersonSegmentation(imageElement);

const maskBackground = true;

// Convert the personSegmentation into a mask to darken the background.

const backgroundDarkeningMask = bodyPix.toMaskImageData(personSegmentation, maskBackground);

const opacity = 0.7;`

const canvas = document.getElementById('canvas');

// draw the mask onto the image on a canvas. With opacity set to 0.7 this will darken the background.

bodyPix.drawMask(

canvas, imageElement, backgroundDarkeningMask, opacity);

# Project CODE:

Let’s go through the main Code: (we have explained with the comments inside)

*// 1. Install dependencies*

*// 2. Import dependencies*

*// 3. Setup webcam and canvas*

*// 4. Define references to those*

*// 5. Load handpose*

*// 6. Detect function*

*// 7. Draw using drawMask*

import React, { useRef } from "react";

*// import logo from './logo.svg';*

import \* as tf from "@tensorflow/tfjs";

import \* as bodyPix from "@tensorflow-models/body-pix";

import Webcam from "react-webcam";

import "./App.css";

**function** App() {

**const** webcamRef = useRef(null);

**const** canvasRef = useRef(null);

**const** runBodysegment = **async** () **=>** {

**const** net = await bodyPix.load();

    console.log("BodyPix model loaded.");

*//  Loop and detect hands*

    setInterval(() **=>** {

      detect(net);

    }, 100);

  };

**const** detect = **async** (net) **=>** {

*// Check data is available*

    if (

      typeof webcamRef.current !== "undefined" &&

      webcamRef.current !== null &&

      webcamRef.current.video.readyState === 4

    ) {

*// Get Video Properties*

**const** video = webcamRef.current.video;

**const** videoWidth = webcamRef.current.video.videoWidth;

**const** videoHeight = webcamRef.current.video.videoHeight;

*// Set video width*

      webcamRef.current.video.width = videoWidth;

      webcamRef.current.video.height = videoHeight;

*// Set canvas height and width*

      canvasRef.current.width = videoWidth;

      canvasRef.current.height = videoHeight;

*// Make Detections*

*// \* One of (see documentation below):*

*// \*   - net.segmentPerson*

*// \*   - net.segmentPersonParts*

*// \*   - net.segmentMultiPerson*

*// \*   - net.segmentMultiPersonParts*

*// const person = await net.segmentPerson(video);*

**const** person = await net.segmentPersonParts(video);

      console.log(person);

*// const coloredPartImage = bodyPix.toMask(person);*

**const** coloredPartImage = bodyPix.toColoredPartMask(person);

**const** opacity = 0.7;

**const** flipHorizontal = false;

**const** maskBlurAmount = 0;

**const** canvas = canvasRef.current;

      bodyPix.drawMask(

        canvas,

        video,

        coloredPartImage,

        opacity,

        maskBlurAmount,

        flipHorizontal

      );

    }

  };

  runBodysegment();

  return (

    <div className="App">

      <header className="App-header">

        <Webcam

          ref={webcamRef}

          style={{

            position: "absolute",

            marginLeft: "auto",

            marginRight: "auto",

            left: 0,

            right: 0,

            textAlign: "center",

            zindex: 9,

            width: 640,

            height: 480,

          }}

        />

        <canvas

          ref={canvasRef}

          style={{

            position: "absolute",

            marginLeft: "auto",

            marginRight: "auto",

            left: 0,

            right: 0,

            textAlign: "center",

            zindex: 9,

            width: 640,

            height: 480,

          }}

        />

      </header>

    </div>

  );

}

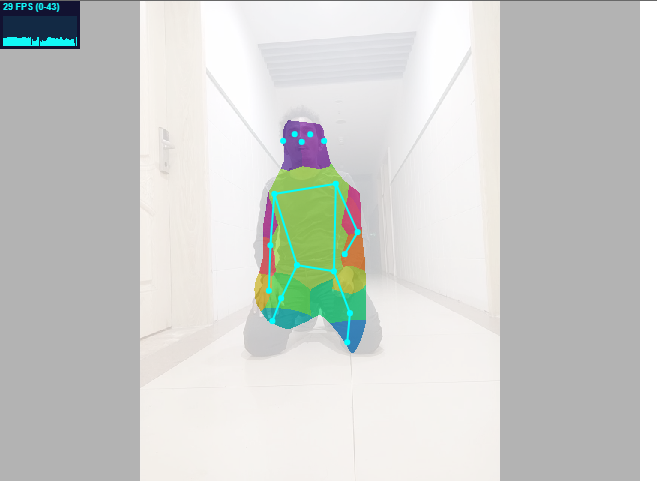
export default App;

# Result Output:

## Single body selection:



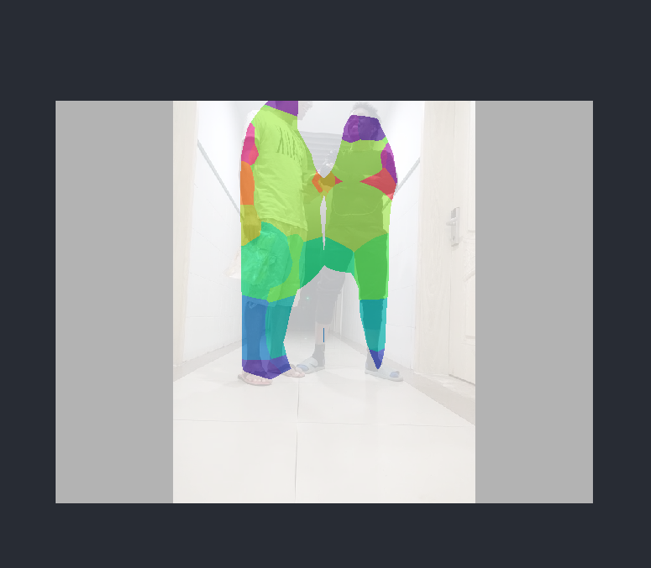
Real image



Segmented Image

## Multi Body Selection:





Conclusion

In this paper, we discussed human body segmentation using TensorFlow js. We installed the required dependencies, loaded the model, input the images and give the expected output.

Finally, we performed a body segmentation on the image.