Classifying Good vs Bad NBA defense using COCO-SSD

Gene Zaleski
Computer Vision
Spring 2023

Description

- For my project, I used the pre-trained TFJS COCO-SSD model.
- I used the model to detect the Person class, then gave the model footage of a basketball game.
- Multiple persons (players) are detected.
- The min distance between any bounding box is measured.
- The distance is then measured and a relevant color bounding box is created

CV Concepts

- This project used a Pre-Trained TFJS model.
- No transfer learning was used, I used the currently existing classes in the model!
- COCO (Common Objects in Context) is a training dataset used to train this model.
- SSD (Single Shot MultiBox Detection) is the method of object detection used by this model.

Video Implementation

- Convert downloaded mp4 files to webcam-like stream.
- Pass stream to COCO-SSD for prediction

HTML

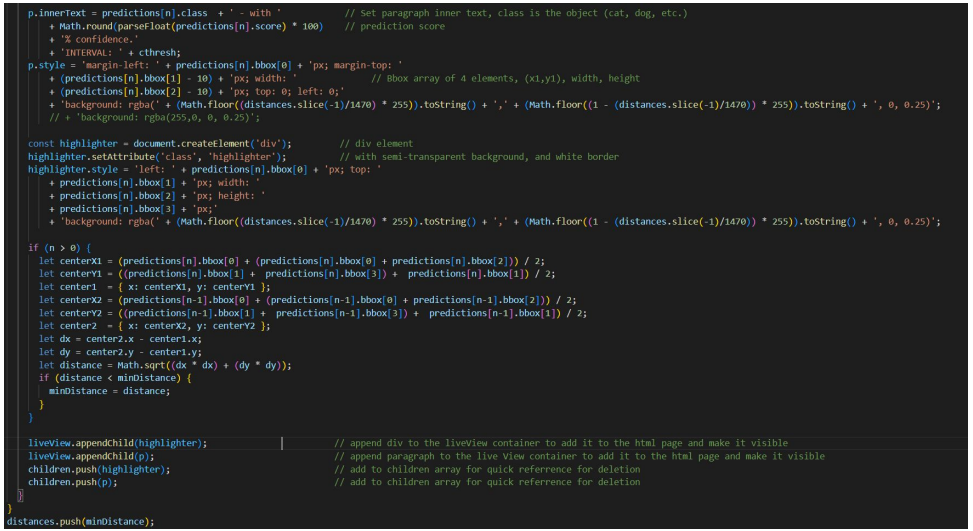
```
<section id="demos" class="invisible">
  <div id="liveView" class="camView">
    <button id="webcamButton">Classify Video</button>
    <video id="webcam" autoplay muted width="1280" height="720"></video>
    <video class="invisible" id="vid-content" muted width="1280" height="720">
      <!-- <source class="invisible" src="./draymondGreen.mp4"> -->
      <source class="invisible" src="./sixersG1.mp4">
    </video>
  </div>
</section>
```

JS

```
// Enable the live webcam view and start classification.
function enableCam(event) {
  // Only continue if the COCO-SSD has finished loading.
  if (!model) {
    return;          // return if model is not loaded
  }

  // Hide the button once clicked.
  event.target.classList.add('removed');

  // getUsermedia parameters to force video but not audio.
  const constraints = {
    video: true                  // only want video stream
  };

  // Activate the webcam stream with asynchronous call, thus the use of then. Then we use an an

  // console.log(video);

  const stream = vcont.captureStream();
  vcont.play()
  // navigator.mediaDevices.getUserMedia(constraints).then(function(stream) {
  video.srcObject = stream;
  video.addEventListener('loadeddata', predictWebcam);   // Register method predictWebcam
  // });
  // video.addEventListener('loadeddata', predictWebcam);
}
```

Classification Algorithm

- Bounding box coordinates are taken from predictions.
- If more than one box, compare the distance between the center of each box.
- Color the box on a scale generated from the distance.



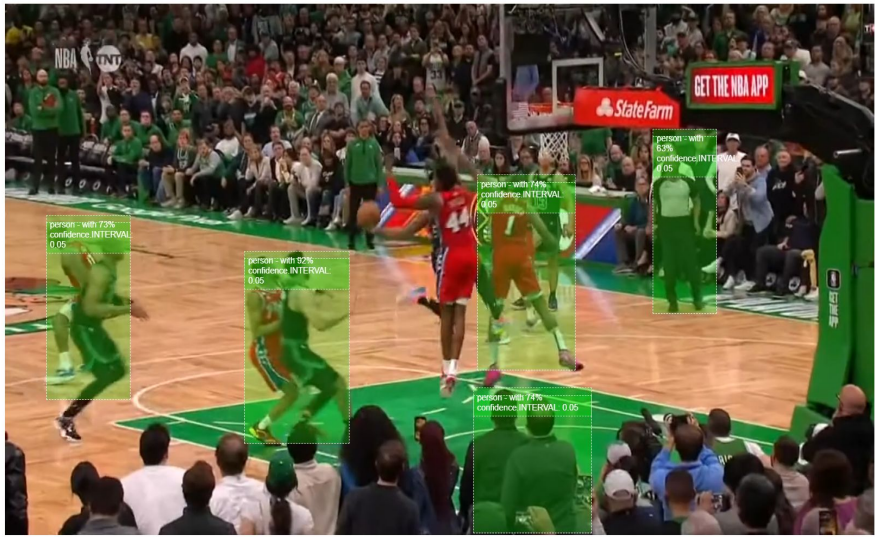
```
    p.innerText = predictions[n].class  + ' - with '             // Set paragraph inner text, class is the object (cat, dog, etc.)
        + Math.round(parseFloat(predictions[n].score) * 100)     // prediction score
        + '% confidence.'
        + 'INTERVAL: ' + cthresh;
    p.style = 'margin-left: ' + predictions[n].bbox[0] + 'px; margin-top: '
        + (predictions[n].bbox[1] - 10) + 'px; width: '              // Bbox array of 4 elements, (x1,y1), width, height
        + (predictions[n].bbox[2] - 10) + 'px; top: 0; left: 0;'
        + 'background: rgba(' + (Math.floor((distances.slice(-1)/1470) * 255)).toString() + ',' + (Math.floor((1 - (distances.slice(-1)/1470)) * 255)).toString() + ', 0, 0.25)';
        // + 'background: rgba(255,0, 0, 0.25)';

    const highlighter = document.createElement('div');          // div element
    highlighter.setAttribute('class', 'highlighter');           // with semi-transparent background, and white border
    highlighter.style = 'left: ' + predictions[n].bbox[0] + 'px; top: '
        + predictions[n].bbox[1] + 'px; width: '
        + predictions[n].bbox[2] + 'px; height: '
        + predictions[n].bbox[3] + 'px;'
        + 'background: rgba(' + (Math.floor((distances.slice(-1)/1470) * 255)).toString() + ',' + (Math.floor((1 - (distances.slice(-1)/1470)) * 255)).toString() + ', 0, 0.25)';

    if (n > 0) {
      let centerX1 = (predictions[n].bbox[0] + (predictions[n].bbox[0] + predictions[n].bbox[2])) / 2;
      let centerY1 = ((predictions[n].bbox[1] +  predictions[n].bbox[3]) +  predictions[n].bbox[1]) / 2;
      let center1  = { x: centerX1, y: centerY1 };
      let centerX2 = (predictions[n-1].bbox[0] + (predictions[n-1].bbox[0] + predictions[n-1].bbox[2])) / 2;
      let centerY2 = ((predictions[n-1].bbox[1] +  predictions[n-1].bbox[3]) +  predictions[n-1].bbox[1]) / 2;
      let center2  = { x: centerX2, y: centerY2 };
      let dx = center2.x - center1.x;
      let dy = center2.y - center1.y;
      let distance = Math.sqrt((dx * dx) + (dy * dy));
      if (distance < minDistance) {
        minDistance = distance;
      }
    }

    liveView.appendChild(highlighter);          // append div to the liveView container to add it to the html page and make it visible
    liveView.appendChild(p);                    // append paragraph to the live View container to add it to the html page and make it visible
    children.push(highlighter);                 // add to children array for quick referrence for deletion
    children.push(p);                           // add to children array for quick referrence for deletion
  }
}
distances.push(minDistance);
```

Demo

NBA Defense Classification

Wait for the COCO-SSD model to load, and then hit the "Classify Video" Button.

Good defense == Green, Bad Defense == Red!



NBA Defense Classification

Wait for the COCO-SSD model to load, and then hit the "Classify Video" Button.

Good defense == Green, Bad Defense == Red!