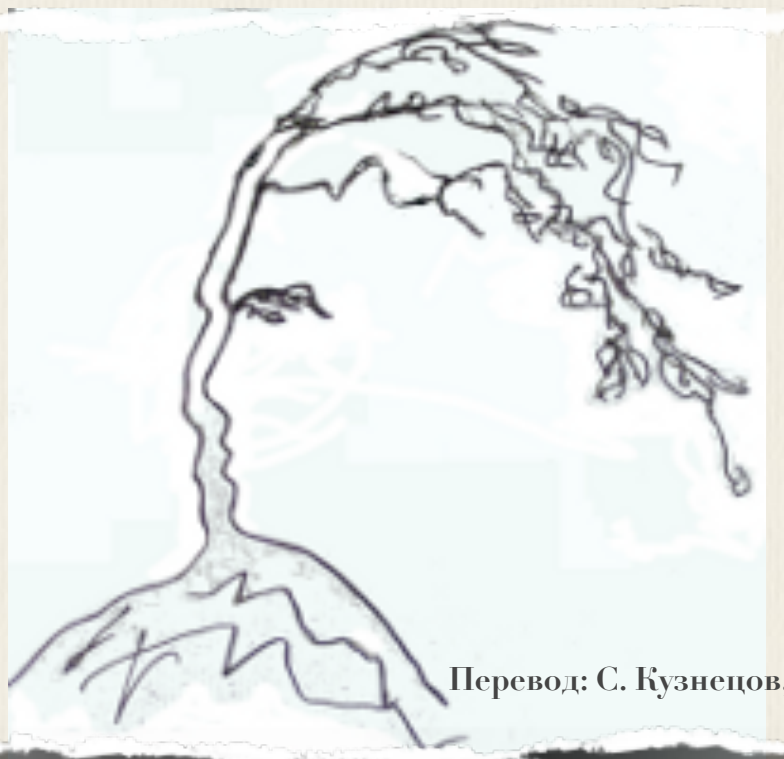
РАФАЭЛЬ МУН(RAPHAEL MUN)

Статьи по машинному обучению в браузере с использованием фреймворка TensorFlow.js

УЧЕБНЫЕ РУКОВОДСТВА



Перевод: С. Кузнецов, 2023 г.

Articles on machine training in the browser with use of a framework TensorFlow.js

Raphael Mun

2020 · 2021

https://www.codeproject.com/Articles/instafluff#Article

Статьи по машинному обучению в браузере с использованием фреймворка TensorFlow.js

Рафаэль Мун

2020 · 2021

https://www.codeproject.com/Articles/instafluff#Article

Перевод: С. Кузнецов, 13.11.2023

Статья 5 «Обнаружение касания лица в фреймворке TensorFlow.js, часть 2: Использование библиотеки BodyPix»

Статья **5 Обнаружение касания лица в фреймворке TensorFlow.js, часть 2: Использование библиотеки BodyPix(Face Touch Detection with TensorFlow.js Part 2: Using BodyPix);** https://www.codeproject.com/Articles/5272775/Face-Touch-Detection-with-TensorFlow-js-Part-2-Usi) является статьей из серии статей **Обнаружение касания лица с помощью Tensorflow.js(Face Touch Detection with Tensorflow.js)**.

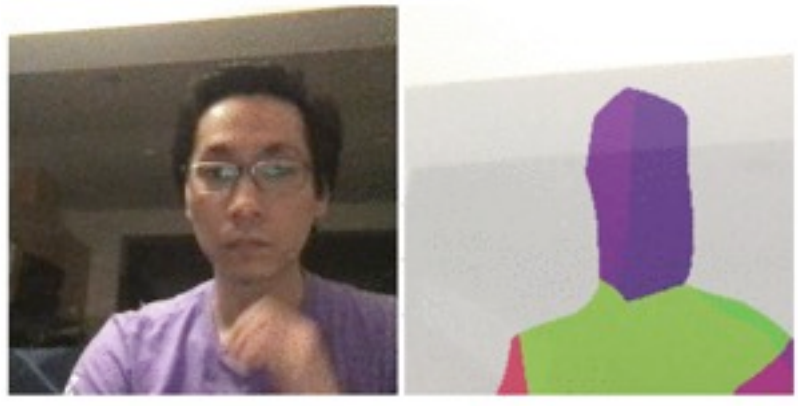
14 июля 2020

В этой статье мы собираемся использовать библиотеку **BodyPix**, **библиотеку обнаружения и сегментации частей тела(body part detection and segmentation library)**, чтобы попытаться удалить шаг обучения(тренировки) **обнаружения касания лица(face touch detection)**.

Здесь мы рассмотрим темы: установка библиотеки **BodyPix**, обнаружение касания лица, как я написал свою функцию **предсказания изображения predictImage()** из шаблона начальной точки, используя **формулу расстояния(distance formula)**, чтобы проверить на **перекрытие области лица(face region overlap)**, и как мы можем использовать библиотеку **BodyPix**, чтобы **оценить позы тела человека(estimate a person's body poses)**.

TensorFlow + JavaScript. Самый популярный, ультрасовременный **AI**-фреймворк(инфраструктура) теперь поддерживает наиболее широко используемый язык программирования на планете, поэтому давайте заставим волшебство произойти посредством **глубокого изучения(deep learning)** прямо в нашем веб-браузере, ускоренном **графическим процессорным устройством(ГПУ; GPU)** через графическую библиотеку **WebGL**, используя фреймворк машинного обучения **TensorFlow.js**!

В предыдущей газете мы обучали AI-систему с помощью TensorFlow.js для моделирования приложения, названного не касайся лица(donottouchyourface.com), которое помогает людям снижать риск заболевания, уча их прекращению касаться их лиц. В этой статье мы собираемся использовать библиотеку BodyPix, библиотеку обнаружения и сегментации частей тела(body part detection and segmentation library), чтобы попытаться удалить шаг обучения(тренировки) обнаружения касания лица(face touch detection).



Not Touch

Начальная точка

Для этого проекта вы должны сделать следующее:

- Импортируйте TensorFlow.js и библиотеку BodyPix
- Добавьте видео элемент для веб-камеры
- Добавьте элемент холст canvas для отладки
- Добавьте элемент текст состояния Касание(Touch) по сравнению с состоянием Нет касания(Not-Touch)
- Добавьте функциональность установки веб-камеры
- Выполните прогноз на модели каждые 200 мс, вместо выбора изображения, но после того, как модель будет обучена впервые

Вот наша начальная точка:

JavaScript

```
<html>
    <head>
        <title>Face Touch Detection with TensorFlow.js Part 2: Using
BodyPix</title>
        <script
src="https://cdn.jsdelivr.net/npm/@tensorflow/tfjs@2.0.0/dist/tf.min.js"><
/script>
        <script
src="https://cdn.jsdelivr.net/npm/@tensorflow-models/body-pix@2.0"></scrip
t>
        <style>
            img, video {
                object-fit: cover;
            }
        </style>
    </head>
    <body>
        <video autoplay playsinline muted id="webcam"
                    width="224" height="224"></video>
        <canvas id="canvas" width="224" height="224"></canvas>
        <h1 id="status">Загрузка.../Loading...</h1>
        <script>
        async function setupWebcam() {
            return new Promise( ( resolve, reject ) => {
                const webcamElement = document.getElementById( "webcam" );
                const navigatorAny = navigator;
                navigator.getUserMedia = navigator.getUserMedia ||
                navigatorAny.webkitGetUserMedia ||
                 navigatorAny.mozGetUserMedia ||
                navigatorAny.msGetUserMedia;
                if( navigator.getUserMedia ) {
                    navigator.getUserMedia( { video: true },
                        stream => {
                            webcamElement.srcObject = stream;
                            webcamElement.addEventListener( 'loadeddata',
                                  resolve, false );
                        },
                    error => reject());
                }
                else {
                    reject();
                }
        });
}

(async () => {
    await setupWebcam();

    setInterval( predictImage, 200 );
})();

async function predictImage() {
```

```
            // Сюда поместите код прогноза
            // Prediction Code Goes Here
        }
        </script>
    </body>
</html>
```

Установка библиотеки BodyPix

Библиотека `BodyPix` , при загрузке, принимает несколько параметров, вы могли бы распознать некоторые из них. Она поддерживает две различных предварительно обученные модели архитектур: MobileNetV1 и ResNet50. Необходимые параметры могут измениться в зависимости от выбранной вами модели. Мы будем использовать модель архитектуры `MobileNet` и библиотеку `BodyPix` инициализируем следующим кодом:

JavaScript

```
(async () => {
    model = await bodyPix.load({
        architecture: 'MobileNetV1',
        outputStride: 16,
        multiplier: 0.50,
        quantBytes: 2
    });
    await setupWebcam();
    setInterval( predictImage, 200 );
})();
```

Обнаружение касаний лица

С помощью `сегментации частей тела(body part segmentation)` мы получаем две части данных от библиотеки `BodyPix`:

- `Характерные элементы(ключевые точки; key points)` частей тела, такие как `нос(nose)`, `уши(ears)`, `запястье(wrist)`, `колено(elbow)` и т.д., представленные в координатах пикселей `2D`-экрана
- `Пиксельные данные 2D-сегментации(2-D segmentation pixel data)`, сохраненные в `формате 1D-массива(1-D array format)`

После краткого тестирования я нашел, что координаты **характерных элементов(ключевых точек; key points)**, извлекаемые для **носа(nose)**, **уши(ears)**, были довольно надежны, в то время как координаты **характерных элементов(ключевых точек; key points)** для **запястий(wrists)** человека были не достаточно точны, чтобы определить, **касается ли рука лица(hand is touching the face)**. Поэтому, мы будем использовать **пиксели сегментации(segmentation pixels)**, чтобы определить касание лица.

Поскольку характерные элементы нос и уши кажутся надежными, то мы можем использовать их, чтобы вычислить **круговую область лица человека. (circle region for the person's face)**. Используя эту круговую область, мы можем определить, перекрывают ли эту область какие-либо пиксели сегментации левой руки или правой руки - и **пометить состояние как касание лица(mark the status as a face touch)**.

Вот то, как я записал свой функцию предсказания изображения **predictImage()** из шаблона начальной точки, используя **формулу расстояния(distance formula)**, для **проверки на перекрытие области лица(check for face region overlap)**:

Python

```
async function predictImage() {
    const img = document.getElementById( "webcam" );
    const segmentation = await model.segmentPersonParts( img );
    if( segmentation.allPoses.length > 0 ) {
        const keypoints = segmentation.allPoses[ 0 ].keypoints;
        const nose = keypoints[ 0 ].position;
        const earL = keypoints[ 3 ].position;
        const earR = keypoints[ 4 ].position;
        const earLtoNose = Math.sqrt( Math.pow( nose.x - earL.x, 2 ) +
                    Math.pow( nose.y - earL.y, 2 ) );
        const earRtoNose = Math.sqrt( Math.pow( nose.x - earR.x, 2 ) +
                    Math.pow( nose.y - earR.y, 2 ) );
        const faceRadius = Math.max( earLtoNose, earRtoNose );

         // Проверка того, что любой из пикселей
         // левой руки left_hand(10) или правой руки right_hand (11)
         // находится внутри круга faceRadius у носа
         // Check if any of the left_hand(10) or right_hand(11)
         // pixels are within the nose to faceRadius
        let isTouchingFace = false;
        for( let y = 0; y < 224; y++ ) {
```

```
        for( let x = 0; x < 224; x++ ) {
            if( segmentation.data[ y * 224 + x ] === 10 ||
                segmentation.data[ y * 224 + x ] === 11 ) {
                const distToNose = Math.sqrt( Math.pow(
                      nose.x - x, 2 ) + Math.pow( nose.y - y, 2 ) );
                // console.log( distToNose );
                if( distToNose < faceRadius ) {
                    isTouchingFace = true;
                    break;
                }
            }
        }
        if( isTouchingFace ) {
            break;
        }
    }
    if( isTouchingFace ) {
        document.getElementById(
                  "status" ).innerText = "Касание(Touch)";
    }
    else {
        document.getElementById(
                  "status" ).innerText = "Нет касания(Not-Touch)";
    }

     // --- «Раскомментируйте»(удалите //) следующий код,
     // чтобы увидеть BodyPix-маску ---
     // --- Uncomment the following to view the BodyPix mask ---
        // const canvas = document.getElementById( "canvas" );
        // bodyPix.drawMask(
        //     canvas, img,
        //     bodyPix.toColoredPartMask( segmentation ),
        //     0.7,
        //     0,
        //     false
        // );
    }
}
```

Если вы захотите видеть пиксели, предсказанные библиотекой **BodyPix**, то вы можете «раскомментировать»(удалить //) нижний отрывок кода функции.

Мой подход к функции предсказания изображения **predictImage()** является очень грубым вычислением, которое использует **близость пикселя руки(hand pixel's proximity)**. Вам может быть брошен забавный вызов, состоящий в нахождении более точного способа обнаружения события, когда рука человека коснулась лица!

Технические примечания

- Одно из преимуществ использования библиотеки **BodyPix** для **обнаружения касания лица(Face Touch Detection)** состоит в том, что пользователь не должен обучать(тренировать) **AI**-систему с помощью примеров нежелательного поведения
- Другое преимущество библиотеки **BodyPix** состоит в том, что она может **сегментировать лицо спереди(segment the face in front)**, когда рука человека скрыта позади него.
- Этот подход и прогноз более специфичный для распознавания **действия касания лица(Face Touch action)**, чем подход, который мы использовали в **предыдущей статье(previous article)**; однако, первый подход может привести к более точным прогнозам, при обучении(тренировке) на достаточном объеме демо-данных
- Ожидайте проблемы с производительностью, поскольку библиотека **BodyPix** является затратной в вычислительном отношении

Финишная черта

Вот полный код:

JavaScript

```
<html>
    <head>
        <title>Face Touch Detection with TensorFlow.js Part 2: Using
BodyPix</title>
        <script
src="https://cdn.jsdelivr.net/npm/@tensorflow/tfjs@2.0.0/dist/tf.min.js"><
/script>
        <script
src="https://cdn.jsdelivr.net/npm/@tensorflow-models/body-pix@2.0"></scrip
t>
        <style>
            img, video {
                object-fit: cover;
            }
        </style>
    </head>
```

```
<body>
    <video autoplay playsinline muted id="webcam"
                        width="224" height="224"></video>
    <canvas id="canvas" width="224" height="224"></canvas>
    <h1 id="status">Загрузка.../Loading...</h1>
    <script>
    async function setupWebcam() {
        return new Promise( ( resolve, reject ) => {
            const webcamElement = document.getElementById( "webcam" );
            const navigatorAny = navigator;
            navigator.getUserMedia = navigator.getUserMedia ||
            navigatorAny.webkitGetUserMedia ||
             navigatorAny.mozGetUserMedia ||
            navigatorAny.msGetUserMedia;
            if( navigator.getUserMedia ) {
                navigator.getUserMedia( { video: true },
                    stream => {
                        webcamElement.srcObject = stream;
                        webcamElement.addEventListener( 'loadeddata',
                          resolve, false );
                    },
                error => reject());
            }
            else {
                reject();
            }
        });
}

let model = null;

(async () => {
    model = await bodyPix.load({
        architecture: 'MobileNetV1',
        outputStride: 16,
        multiplier: 0.50,
        quantBytes: 2
    });
    await setupWebcam();
    setInterval( predictImage, 200 );
})();

async function predictImage() {
    const img = document.getElementById( "webcam" );
    const segmentation = await model.segmentPersonParts( img );
    if( segmentation.allPoses.length > 0 ) {
        const keypoints = segmentation.allPoses[ 0 ].keypoints;
        const nose = keypoints[ 0 ].position;
        const earL = keypoints[ 3 ].position;
        const earR = keypoints[ 4 ].position;
        const earLtoNose = Math.sqrt(
             Math.pow( nose.x - earL.x, 2 ) +
             Math.pow( nose.y - earL.y, 2 ) );
        const earRtoNose = Math.sqrt(
```

```
                    Math.pow( nose.x - earR.x, 2 ) +
                    Math.pow( nose.y - earR.y, 2 ) );
                const faceRadius = Math.max( earLtoNose, earRtoNose );

                // Проверка того, что любой из пикселей
                // левой руки left_hand(10) или правой руки right_hand
(11)
                // находится внутри круга faceRadius у носа
                // Check if any of the left_hand(10) or right_hand(11)
                // pixels are within the nose to faceRadius
                let isTouchingFace = false;
                for( let y = 0; y < 224; y++ ) {
                    for( let x = 0; x < 224; x++ ) {
                        if( segmentation.data[ y * 224 + x ] === 10 ||
                            segmentation.data[ y * 224 + x ] === 11 ) {
                            const distToNose = Math.sqrt(
                                      Math.pow( nose.x - x, 2 ) +
                                      Math.pow( nose.y - y, 2 ) );
                            // вывод расстояния в журнал консоли
                            // console.log( distToNose );
                            if( distToNose < faceRadius ) {
                                isTouchingFace = true;
                                break;
                            }
                        }
                    }
                    if( isTouchingFace ) {
                        break;
                    }
                }
                if( isTouchingFace ) {
                     document.getElementById(
                         "status" ).innerText = "Касание(Touch)";
                 }
                 else {
                     document.getElementById(
                         "status" ).innerText = "Нет касания(Not-Touch)";
                 }

                 // --- «Раскомментируйте»(удалите //) следующий код,
                 // чтобы увидеть BodyPix-маску ---
                 // --- Uncomment the following to view
                 // the BodyPix mask ---
                // const canvas = document.getElementById( "canvas" );
                // bodyPix.drawMask(
                //     canvas, img,
                //     bodyPix.toColoredPartMask( segmentation ),
                //     0.7,
                //     0,
                //     false
                // );
    }
}
</script>
```

```
    </body>
</html>
```

Что далее? Можем ли мы сделать еще больше с помощью фреймворка машинного обучения TensorFlow.js?

В этом проекте мы видели, как легко мы можем использовать библиотеку **BodyPix**, чтобы **оценить позы тела человека(estimate a person's body poses)**. В следующем проекте давайте повторно рассмотрим изучение передачи изображений из веб-камеры и немного позабавиться с ними.

В серии статей, смотрите следующую статью **интерпретация жестов рук и языка жестов в веб-камере с AI-системой, используя TensorFlow.js(**Interpreting Hand Gestures and Sign Language in the Webcam with AI using TensorFlow.js**)**, где мы рассмотрим, можем ли мы более глубоко обучить **AI**-систему некоторым жестам рук и языку жестов.

Эта статья - статья из серии статей **Обнаружение касания лица с помощью Tensorflow.js(Face Touch Detection with Tensorflow.js)**.