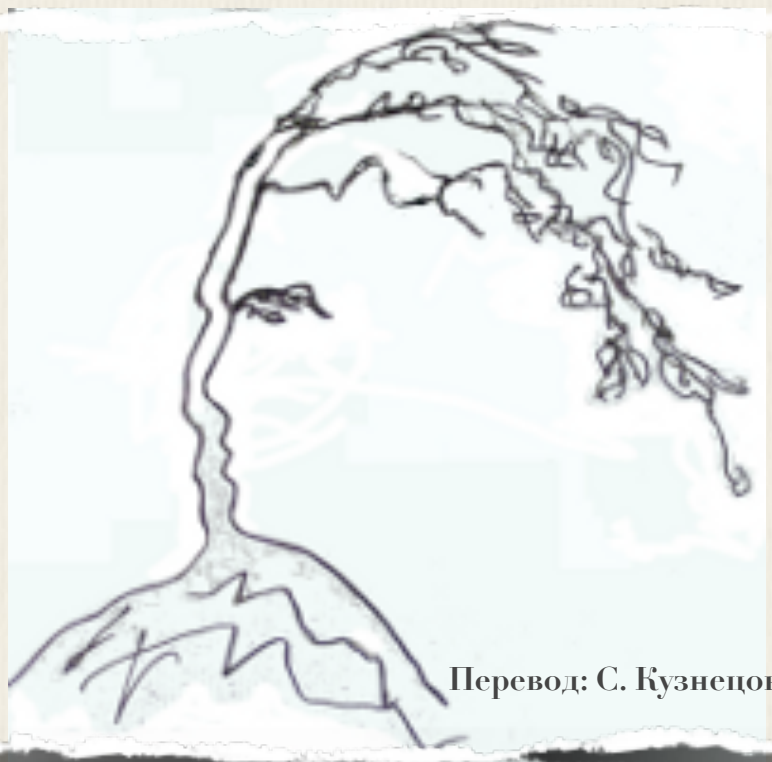
РАФАЭЛЬ МУН(RAPHAEL MUN)

Серия статей «Фильтры искусственного интеллекта лица в браузере»

УЧЕБНЫЕ РУКОВОДСТВА



Перевод: С. Кузнецов, 2024 г.

Articles: AI Face Filters in the Browser

Raphael Mun

2021

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

Серия статей «Фильтры искусственного интеллекта лица в браузере»

Рафаэль Мун

2021

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

Перевод: С. Кузнецов, 24.02.2024

Статья 5 «Создание волшебной шляпы обнаружения эмоций в браузере с использованием библиотеки TensorFlow.js»

Статья 5 Создание волшебной шляпы обнаружения эмоций в браузере с использованием библиотеки TensorFlow.js(Building a Magical Emotion Detection Hat in the Browser with TensorFlow.js); https://www.codeproject.com/Articles/5293495/Building-a-Magical-Emotion-Detection-Hat-in-the-Br) является статьей из серии статей Фильтры искусственного интеллекта лица в браузере(AI Face Filters in the Browser).

8 февраля 2021

В этой статье мы собираемся объединить все предыдущие знания об отслеживании лиц и построим забавный визуальный образ.

Здесь мы собираемся соединить все предыдущие части, чтобы создать волшебную шляпу обнаружения эмоций(magical emotion detection hat), которая распознает и отвечает на наши выражения лица(facial expressions), когда мы виртуально носим ее.

- Загрузка кода и файлов - 565.6 KB

Введение

Приложения, подобные приложению Snapchat, предлагают удивительное разнообразие фильтров лиц и линз, которые позволяют вам накладывать интересные эффекты на фотографии и видео. Если когда-либо вы «приделывали» себе виртуальные уши собаки или маскарадную шляпу, то знаете, это может быть забавно!

Задавались ли вы вопросом, как создать эти виды фильтров с нуля? Ну, теперь есть шанс научиться делать все в веб-браузере! В этой серии статей

мы собираемся показать, как в браузере создать фильтры в стиле **Snapchat**, обучить **AI**-модель понимать выражения лица, используя отслеживание лица и библиотеку **Tensorflow.js**.

(надпись под снимком экрана: **Обнаружено: счастливый!(Detected:happy)**)

Вы можете загрузить демонстрационный пример этого проекта. Возможно, для обеспечения производительности, вы будете должны в своем веб-браузере включить поддержку **Web**-графики **WebGL**.

Также можно загрузить код и файлы для этой серии статей.

Предполагается, что вы знакомы с языками **JavaScript** и **HTML** и имеете, по крайней мере, базовое понимание нейронных сетей. Если вы плохо знакомы с фреймворком **TensorFlow.js**, то рекомендуем прочитать статью «Начало работы с глубоким изучением в браузере с использованием фреймворка TensorFlow.js»**(«Getting Started With Deep Learning in Your B r o w s e r U s i n g T e n s o r F l o w . j s » ; https://www.codeproject.com/Articles/5272760/Getting-Started-With-Deep-Lea rning-in-Your-Browser)**, которая является статьей из серии статей **Обнаружение касания лица с помощью Tensorflow.js(Face Touch Detection with Tensorflow.js)**

Если хотели бы увидеть больше того, что возможно в веб-браузере с помощью фреймворка **TensorFlow.js**, прочтите статьи из серии по **искусственному интеллекту(AI)**: «Собаки и пицца: машинное зрение в браузере с использованием TensorFlow.js»**(«Dogs and Pizza: Computer Vision in the Browser With T e n s o r F l o w . j s » ;**

https://www.codeproject.com/Articles/5272771/Dogs-and-Pizza-Computer-Vision-in-the-Browser-With) и «Роботы чатов с помощью фреймворка TensorFlow.js»(Chatbots using TensorFlow.js.).

Ношение виртуальных аксессуаров является забавой, но это находится только в одном шаге от ношения их в реальной жизни. Мы могли легко создать приложение, позволяющее фактически примерять шляпу, - похожее на приложение, желаемое создать вами для веб-сайта электронной коммерции. Но если мы собираемся сделать это, то почему бы заодно немного не позабавиться? Программном обеспечение является волшебным, потому что мы можем наше воображение сделать реальным.

В этой статье мы собираемся соединить все предыдущие части, чтобы создать волшебную шляпу обнаружения эмоций(magical emotion detection hat), которая распознает и отвечает на наши выражения лица(facial expressions), когда мы виртуально носим ее.

Создание волшебной шляпы

Помните, когда мы разработали проект функциональности по обнаружению эмоций на лице в реальном времени с помощью веб-камеры в браузере с использованием библиотеки TensorFlow.js? Теперь давайте к тому проекту добавим некоторую графику - придадим ему «лицо»(«face»), так сказать.

Чтобы создать нашу «живущую виртуальную шляпу»(living virtual hat), мы к веб-странице добавим графические активы, в виде скрытых HTML-элементов класса изображение <img>:

HTML

```
<img id="hat-angry" src="web/hats/angry.png" style="visibility: hidden;"
/>
<img id="hat-disgust" src="web/hats/disgust.png" style="visibility:
hidden;" />
<img id="hat-fear" src="web/hats/fear.png" style="visibility: hidden;" />
<img id="hat-happy" src="web/hats/happy.png" style="visibility: hidden;"
/>
```

```
<img id="hat-neutral" src="web/hats/neutral.png" style="visibility:
hidden;" />
<img id="hat-sad" src="web/hats/sad.png" style="visibility: hidden;" />
<img id="hat-surprise" src="web/hats/surprise.png" style="visibility:
hidden;" />
```

Ключевое свойство этого проекта состоит в том, что мы все время показываем шляпу в надлежащей позиции и размере и поэтому мы сохраним `"состояния"("states")` шляпы в глобальной переменной:

JavaScript

```
let currentEmotion = "neutral";
let hat = { scale: { x: 0, y: 0 }, position: { x: 0, y: 0 } };
```

Чтобы рисовать шляпу с этим размером и в этой позиции, мы будем в каждом кадре использовать преобразования `2D`-холста.

JavaScript

```
async function trackFace() {
    ...

    output.drawImage(
        video,
        0, 0, video.width, video.height,
        0, 0, video.width, video.height
    );
    let hatImage = document.getElementById( `hat-${currentEmotion}` );
    output.save();
    output.translate( -hatImage.width / 2, -hatImage.height / 2 );
    output.translate( hat.position.x, hat.position.y );
    output.drawImage(
        hatImage,
        0, 0, hatImage.width, hatImage.height,
        0, 0, hatImage.width * hat.scale, hatImage.height * hat.scale
    );
    output.restore();

    ...
}
```

Используя **ключевые лицевые точки(key facial points)**, предоставленные **TensorFlow**-моделью, мы можем вычислить размер и позицию шляпы относительно лица, чтобы установить вышеупомянутые значения.

Размер головы мы можем оценить по расстоянию между глазами и аппроксимировать вектор направления **"вверх"("up")**, используя точку середины расстояния между глазами **midwayBetweenEyes** и точку нижней части носа **noseBottom**, и он может использоваться для перемещения шляпы на лоб(в отличие от виртуальных очков в предыдущей статье).

JavaScript

```
const eyeDist = Math.sqrt(
    ( face.annotations.leftEyeUpper1[ 3 ][ 0 ] -
face.annotations.rightEyeUpper1[ 3 ][ 0 ] ) ** 2 +
    ( face.annotations.leftEyeUpper1[ 3 ][ 1 ] -
face.annotations.rightEyeUpper1[ 3 ][ 1 ] ) ** 2 +
    ( face.annotations.leftEyeUpper1[ 3 ][ 2 ] -
face.annotations.rightEyeUpper1[ 3 ][ 2 ] ) ** 2
);
```

```
const faceScale = eyeDist / 80;
let upX = face.annotations.midwayBetweenEyes[ 0 ][ 0 ] -
face.annotations.noseBottom[ 0 ][ 0 ];
let upY = face.annotations.midwayBetweenEyes[ 0 ][ 1 ] -
face.annotations.noseBottom[ 0 ][ 1 ];
const length = Math.sqrt( upX ** 2 + upY ** 2 );
upX /= length;
upY /= length;

hat = {
    scale: faceScale,
    position: {
        x: face.annotations.midwayBetweenEyes[ 0 ][ 0 ] +
                upX * 100 * faceScale,
        y: face.annotations.midwayBetweenEyes[ 0 ][ 1 ] +
                upY * 100 * faceScale,
    }
};
```

Мы сохраняем предсказанную эмоцию в переменной **currentEmotion** и соответствующее изображение шляпы будет показано – и мы готовы примерить его!

JavaScript

```
if( points ) {
    let emotion = await predictEmotion( points );
    // Было: setText( `Detected: ${emotion}` );
    // Стало: начало кода руссификации
    let cur_emotion = emotion
    let rus_eng_cur_emotion = cur_emotion
    if ( cur_emotion == "angry" ) {
                    rus_eng_cur_emotion = "сердитый(angry)" }
    else if ( cur_emotion == "disgust" ) {
                    rus_eng_cur_emotion = "отвращение(disgust)" }
    else if ( cur_emotion == "fear" ) {
                    rus_eng_cur_emotion = "страх(fear)" }
    else if ( cur_emotion == "happy" ) {
                    rus_eng_cur_emotion = "счастливый(happy)" }
    else if ( cur_emotion == "neutral" ) {
                    rus_eng_cur_emotion = "нейтральный(neutral)" }
    else if ( cur_emotion == "sad" ) {
                    rus_eng_cur_emotion = "печальный(sad)" }
    else if ( cur_emotion == "surprise" ) {
                    rus_eng_cur_emotion = "удивление(surprise)" }
    setText( `Обнаружено: ${rus_eng_cur_emotion}` );
    // Стало: конец кода
    currentEmotion = emotion;
}
else {
    setText( "Не лицо/No Face" );
}
```

Финишная черта

Вот полный код для этого проекта:

HTML

```
<html>
    <head>
        <title>Создание волшебной шляпы обнаружения эмоций</title>
        <script src=
"https://cdn.jsdelivr.net/npm/@tensorflow/tfjs@2.4.0/dist/tf.min.js">
        </script>
        <script src=
"https://cdn.jsdelivr.net/npm/@tensorflow-models/face-landmarks-detection@
0.0.1/dist/face-landmarks-detection.js">
        </script>
```

```
</head>
<body>
    <canvas id="output"></canvas>
    <video id="webcam" playsinline style="
        visibility: hidden;
        width: auto;
        height: auto;
        ">
    </video>
    <h1 id="status">Загрузка.../Loading...</h1>
    <img id="hat-angry" src="web/hats/angry.png"
            style="visibility: hidden;" />
    <img id="hat-disgust" src="web/hats/disgust.png"
            style="visibility: hidden;" />
    <img id="hat-fear" src="web/hats/fear.png"
            style="visibility: hidden;" />
    <img id="hat-happy" src="web/hats/happy.png"
            style="visibility: hidden;" />
    <img id="hat-neutral" src="web/hats/neutral.png"
            style="visibility: hidden;" />
    <img id="hat-sad" src="web/hats/sad.png"
            style="visibility: hidden;" />
    <img id="hat-surprise" src="web/hats/surprise.png"
            style="visibility: hidden;" />
    <script>
    function setText( text ) {
        document.getElementById( "status" ).innerText = text;
    }

    function drawLine( ctx, x1, y1, x2, y2 ) {
        ctx.beginPath();
        ctx.moveTo( x1, y1 );
        ctx.lineTo( x2, y2 );
        ctx.stroke();
    }

    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();
        }
    });
}
         //  [ "сердитый",  "отвращение", "страх", "счастливый",
         //   "нейтральный", "печальный",  "удивление"];
const emotions = [ "angry", "disgust", "fear", "happy",
                   "neutral", "sad", "surprise" ];
let emotionModel = null;

let output = null;
let model = null;

let currentEmotion = "neutral";
let hat = { scale: { x: 0, y: 0 }, position: { x: 0, y: 0 } };

async function predictEmotion( points ) {
    let result = tf.tidy( () => {
        const xs = tf.stack( [ tf.tensor1d( points ) ] );
        return emotionModel.predict( xs );
    });
    let prediction = await result.data();
    result.dispose();
    // Получите индекс максимального значения
     // Get the index of the maximum value
    let id = prediction.indexOf( Math.max( ...prediction ) );
    return emotions[ id ];
}

async function trackFace() {
    const video = document.querySelector( "video" );
    const faces = await model.estimateFaces( {
        input: video,
        returnTensors: false,
        flipHorizontal: false,
    });
    output.drawImage(
        video,
        0, 0, video.width, video.height,
        0, 0, video.width, video.height
    );
    let hatImage = document.getElementById(
                                `hat-${currentEmotion}` );
    output.save();
    output.translate( -hatImage.width / 2, -hatImage.height / 2 );
    output.translate( hat.position.x, hat.position.y );
    output.drawImage(
        hatImage,
```

```
        0, 0, hatImage.width, hatImage.height,
        0, 0, hatImage.width * hat.scale,
              hatImage.height * hat.scale
    );
    output.restore();

    let points = null;
    faces.forEach( face => {
        const x1 = face.boundingBox.topLeft[ 0 ];
        const y1 = face.boundingBox.topLeft[ 1 ];
        const x2 = face.boundingBox.bottomRight[ 0 ];
        const y2 = face.boundingBox.bottomRight[ 1 ];
        const bWidth = x2 - x1;
        const bHeight = y2 - y1;

        // Добавьте только нос, щеки, глаза, брови и рот
        // Add just the nose, cheeks, eyes, eyebrows & mouth
        const features = [
            "noseTip",
            "leftCheek",
            "rightCheek",
            "leftEyeLower1", "leftEyeUpper1",
            "rightEyeLower1", "rightEyeUpper1",
            "leftEyebrowLower", //"leftEyebrowUpper",
            "rightEyebrowLower", //"rightEyebrowUpper",
            "lipsLowerInner", //"lipsLowerOuter",
            "lipsUpperInner", //"lipsUpperOuter",
        ];
        points = [];
        features.forEach( feature => {
            face.annotations[ feature ].forEach( x => {
                points.push( ( x[ 0 ] - x1 ) / bWidth );
                points.push( ( x[ 1 ] - y1 ) / bHeight );
            });
        });

        const eyeDist = Math.sqrt(
            ( face.annotations.leftEyeUpper1[ 3 ][ 0 ] -
              face.annotations.rightEyeUpper1[ 3 ][ 0 ] ) ** 2 +
            ( face.annotations.leftEyeUpper1[ 3 ][ 1 ] -
              face.annotations.rightEyeUpper1[ 3 ][ 1 ] ) ** 2 +
            ( face.annotations.leftEyeUpper1[ 3 ][ 2 ] -
              face.annotations.rightEyeUpper1[ 3 ][ 2 ] ) ** 2
        );
        const faceScale = eyeDist / 80;
        let upX = face.annotations.midwayBetweenEyes[ 0 ][ 0 ] -
                  face.annotations.noseBottom[ 0 ][ 0 ];
        let upY = face.annotations.midwayBetweenEyes[ 0 ][ 1 ] -
                  face.annotations.noseBottom[ 0 ][ 1 ];
        const length = Math.sqrt( upX ** 2 + upY ** 2 );
        upX /= length;
```

```
        upY /= length;

        hat = {
            scale: faceScale,
            position: {
                x: face.annotations.midwayBetweenEyes[ 0 ][ 0 ] +
                        upX * 100 * faceScale,
                y: face.annotations.midwayBetweenEyes[ 0 ][ 1 ] +
                        upY * 100 * faceScale,
            }
        };
    });

    if( points ) {
        let emotion = await predictEmotion( points );
        // Было: setText( `Detected: ${emotion}` );
        // Стало: начало кода руссификации
         let cur_emotion = emotion
         let rus_eng_cur_emotion = cur_emotion
         if ( cur_emotion == "angry" ) {
             rus_eng_cur_emotion = "сердитый(angry)" }
         else if ( cur_emotion == "disgust" ) {
             rus_eng_cur_emotion = "отвращение(disgust)" }
         else if ( cur_emotion == "fear" ) {
             rus_eng_cur_emotion = "страх(fear)" }
         else if ( cur_emotion == "happy" ) {
             rus_eng_cur_emotion = "счастливый(happy)" }
            else if ( cur_emotion == "neutral" ) {
                rus_eng_cur_emotion = "нейтральный(neutral)" }
            else if ( cur_emotion == "sad" ) {
                rus_eng_cur_emotion = "печальный(sad)" }
            else if ( cur_emotion == "surprise" ) {
                rus_eng_cur_emotion = "удивление(surprise)" }
            setText( `Обнаружено: ${rus_eng_cur_emotion}` );
            // Стало: конец кода
            currentEmotion = emotion;
        }
        else {
            setText( "Не лицо/No Face" );
        }

        requestAnimationFrame( trackFace );
    }

    (async () => {
        await setupWebcam();
        const video = document.getElementById( "webcam" );
        video.play();
        let videoWidth = video.videoWidth;
        let videoHeight = video.videoHeight;
        video.width = videoWidth;
        video.height = videoHeight;
```

```
let canvas = document.getElementById( "output" );
canvas.width = video.width;
canvas.height = video.height;

output = canvas.getContext( "2d" );
output.translate( canvas.width, 0 );
output.scale( -1, 1 ); // Зеркалируем

output.fillStyle = "#fdffb6";
output.strokeStyle = "#fdffb6";
output.lineWidth = 2;

// Загрузка модели обнаружения признаков лица
// Load Face Landmarks Detection
model = await faceLandmarksDetection.load(
    faceLandmarksDetection.SupportedPackages.mediapipeFacemesh
);
// Загрузка модели обнаружения эмоций на лице
// Load Emotion Detection
emotionModel = await tf.loadLayersModel(
                    'web/model/facemo.json' );

setText( "Загружено!/Loaded!" );

trackFace();
        })();
        </script>
    </body>
</html>
```

Что далее? Можем мы использовать наши глаза и рот в качестве «пульта управления»?

Этот проект объединил все предыдущие знания об отслеживании лиц и мы построили забавный визуальный образ. А что, если мы могли бы сделать использование нашего лица для интерактивного взаимодействия?

В следующей, заключительной статье этой серии статей мы реализуем **обнаружение мигания глаз и открытия рта(detect eye blinks and the mouth opens)**, чтобы сделать интерактивную сцену. Оставайтесь с нами!