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, 18.02.2024

Статья 3 «Обнаружение эмоций на лице в реальном времени с помощью веб-камеры в браузере с использованием библиотеки TensorFlow.js»

Статья **3 Обнаружение эмоций на лице в реальном времени с помощью веб-камеры в браузере с использованием библиотеки TensorFlow.js(Real-Time Facial Emotion Detection with Webcam in the Browser UsingTensorFlow.js); https://www.codeproject.com/Articles/5293493/Real-Time-Facial-Emotion-Detection-with-Webcam-in)** является статьей из серии статей **Фильтры искусственного интеллекта лица в браузере(AI Face Filters in the Browser)**.

4 февраля 2021

В этой статье мы будем использовать **живое видео из веб-камеры(live webcam video)** нашего лица и увидим, может ли в режиме реального времени модель реагировать на наши выражения лица.

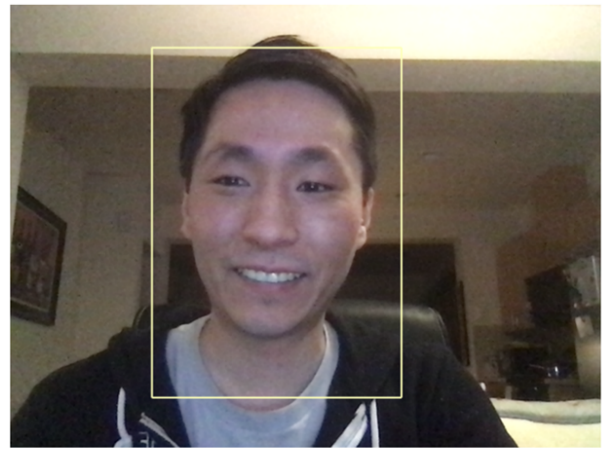
Ранее мы научились использовать **искусственный интеллект(ИИ; AI)** в веб-браузере для отслеживания лиц в режиме реального времени и применять глубокое обучение для обнаружения и классификации эмоций на лице. Таким образом, здесь мы соединяем эти две функциональности и увидим, можем ли мы выполнить обнаруживать эмоции из веб-камеры в режиме реального времени.

- **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 Browser Using TensorFlow.js»; https://www.codeproject.com/Articles/5272760/Getting-Started-With-Deep-Learning-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 TensorFlow.js»;**

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.).

Ранее мы научились использовать искусственный интеллект(ИИ; AI) в веб-браузере для отслеживания лиц в режиме реального времени и применять глубокое обучение для обнаружения и классификации эмоций на лице. Следующий логический шаг должен был бы соединить эти две функциональности и увидеть, можем ли мы выполнить обнаруживать эмоции из веб-камеры в режиме реального времени. Давайте сделаем это!

Добавление обнаружения эмоций на лице

В этом проекте мы нашу обученную модель обнаружения эмоций на лице проверим на видео реального времени от веб-камеры. Мы начнем со стартовоо шаблона на базе финального кода из проекта отслеживания лица и изменим его в части кода обнаружения эмоции на лице.

Давайте загрузим и используем нашу предварительно обученную модель выражения на лице. Сначала мы определим некоторые глобальные переменные для обнаружения эмоций, точно так же, как мы делали ранее:

JavaScript

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

Затем, мы можем загрузить модель обнаружения эмоции внутри асинхронного блока:

JavaScript

```
(async () => {
    ...
```

```
    // 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' );

    ...
})();
```

И мы можем добавить служебную функцию выполнения модели на основе введенных **ключевых лицевых точек(key facial points)** и возвращающую прогноз , со следующим кодом:

JavaScript

```
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 ];
}
```

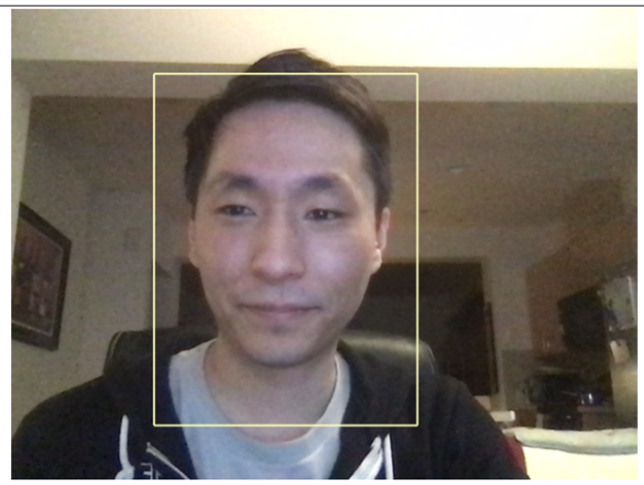
Наконец, мы должны захватить **ключевые лицевые точки(key facial points)** из обнаружения внутри функции отслеживания лица **trackFace** и передать их модулю прогнозирования эмоции.

JavaScript

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

    let points = null;
    faces.forEach( face => {
        ...

        // Добавьте только нос, щеки, глаза, брови & рот
```

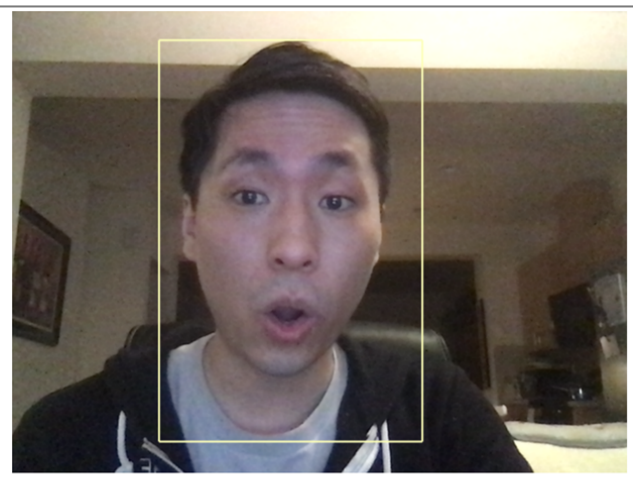
```
        // Add just the nose, cheeks, eyes, eyebrows & mouth
        const features = [
            "noseTip",
            "leftCheek",
            "rightCheek",
            "leftEyeLower1", "leftEyeUpper1",
            "rightEyeLower1", "rightEyeUpper1",
            "leftEyebrowLower", //"leftEyebrowUpper",
            "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 );
            });
        });
    });

    if( points ) {
        let emotion = await predictEmotion( points );
        // setText( `${setIndex + 1}.
        // Ожидаемо: ${ferData[ setIndex ].emotion} vs. ${emotion}` );
        //  setText( `Обнаружено: ${emotion}` );
            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}` );
                    }
             else {
                setText( "Не лицо" );
        }
        requestAnimationFrame( trackFace );
    }
```

Это - всё, что требуется для получения выполнения. Теперь, когда вы открываете веб-страницу, она должна обнаружить ваше лицо и распознать различные эмоции. Экспериментируйте и и развлекайтесь!

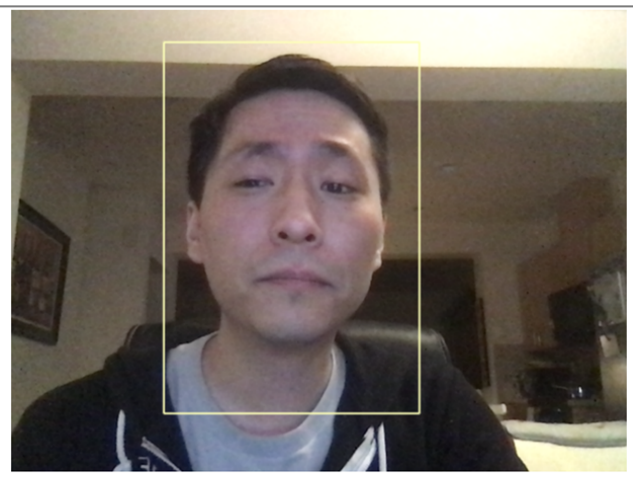


(надпись под снимком экрана: **`Обнаружено: нейтральный(Detected: neutral)`**)

(надпись под снимком экрана: **`Обнаружено: отвращение(Detected: disgust)`**)

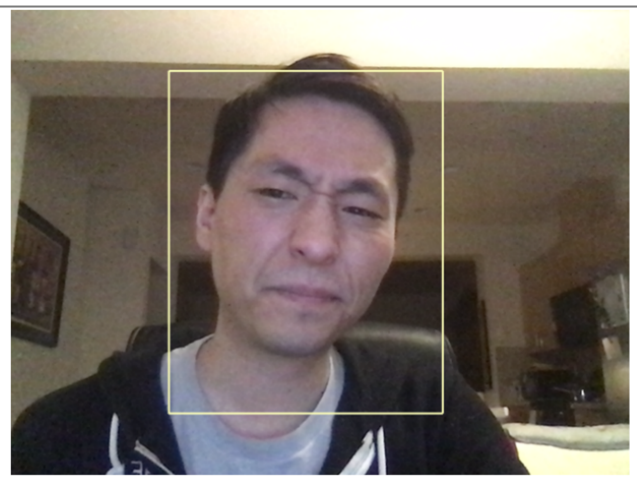


(надпись под снимком экрана: **`Обнаружено: удивление(Detected: surprise)`**)



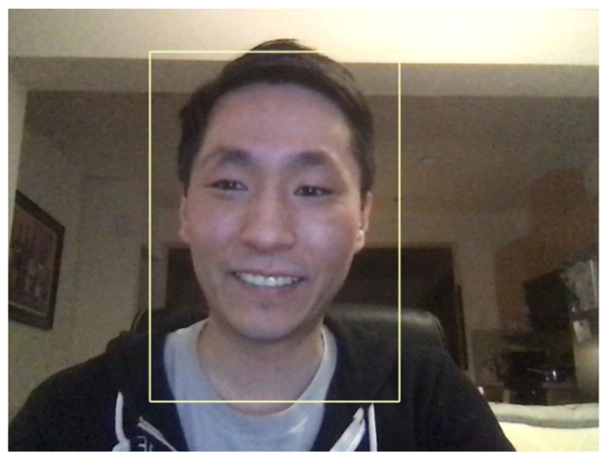
Detected: fear

(надпись под снимком экрана: `Обнаружено: страх(Detected:fear)`)



Detected: angry

(надпись под снимком экрана: `Обнаружено: сердитый(Detected:angry)`)



Detected: happy

(надпись под снимком экрана: `Обнаружено: счастливый(Detected: happy)`)

Финишная черта

Вот полный код, нужный для завершения этого проекта:

HTML

```
<html>
    <head>
        <meta charset="UTF-8">
        <title>Обнаружение эмоций на лице в реальном времени с помощью
веб-камеры в браузере с использованием библиотеки TensorFlow.js</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-detect
ion@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>
        <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;

        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 points = null;
            faces.forEach( face => {
            // Рисуем ограничивающий прямоугольник вокруг лица
```

```
// Draw the bounding box
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;
drawLine( output, x1, y1, x2, y1 );
drawLine( output, x2, y1, x2, y2 );
drawLine( output, x1, y2, x2, y2 );
drawLine( output, x1, y1, x1, y2 );

// Добавьте только нос, щеки, глаза, брови & 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 );
            });
        });
    });

     if( points ) {
let emotion = await predictEmotion( points );
// setText( `${setIndex + 1}.
// Ожидаемо: ${ferData[ setIndex ].emotion} vs. ${emotion}` );
//  setText( `Обнаружено: ${emotion}` );
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}` );
                 }
        else {
             setText( "Не лицо" );
        }
            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? Когда мы сможем носить виртуальные очки?

Код, из первых двух статей этой серии, позволил нам создать детектор эмоций на лице в реальном времени, только с небольшим количеством **JavaScript**-кода. Вообразите, что еще можно сделать с помощью библиотеки **TensorFlow.js**!

В следующей статье мы возвратимся к нашей цели создания в браузере фильтра лица в стиле **Snapchat**, используем полученные знания об отслеживании лиц и добавим **3D**-рендеринг с помощью библиотеки **Three.js**. Оставайтесь с нами!

Эта статья является статьей из серии статей **Фильтры искусственного интеллекта лица в браузере(AI Face Filters in the Browser)**.