• CREATE AN HTM L FILE

- 11 Comments

Handwritten recognition enable us to convert the handwriting documents into digital form. This technology is now being use in numerous ways : reading postal addresses, bank check amounts, digitizing historical literature.

Thanks to tensorflow.js, it brings this powerful technology into the browser. In this article, we are going to build a web application that can predict the digit you draw on the canvas.

Contents hide

Handwritten digit recognition demo

Draw on the black canvas below with your mouse on desktop or your finger on your mobile, click “Predict” to get result of the hand written digit prediction, click “Clean” to start drawing again

Predict
Clear

Step 3 : Set up canvas

For the user to draw a digit using mouse on desktop or finger on mobile devices, we need to create a HTML5 element called canvas. Inside the canvas, the user will draw the digit. We will feed the user drawn digit into the deep neural network that we have created to make predictions.

HTML – index.html

Add a placeholder to contain the canvas that you can draw digit on

```
<div id="canvas_box" class="canvas-box"></div>
```

Add “Predict” button to get result of the hand written digit prediction, “Clean” button to wipe the canvas and start drawing again

```
<button id="clear-button" class="btn btn-dark">Clear</button>
<button id="predict-button" class="btn btn-dark">Predict</button>
```

At the end of the <body>, include the main javascript file digit-recognition.js

```
            <script src="js/digit-recognition.js"></script>
        </body>
</html>
```

Javascript – digit-recognition.js

Initialize the variables

```

let model;
var canvasWidth             = 150;
var canvasHeight            = 150;
var canvasStrokeStyle       = "white";
var canvasLineJoin          = "round";
var canvasLineWidth         = 10;
var canvasBackgroundColor   = "black";
var canvasId                = "canvas";
var clickX = new Array();
var clickY = new Array();
var clickD = new Array();
var drawing;

```

Create the canvas and append it to the placeholder to display

it to predict the digit that we drawn on the cavas.

Load Model

function loadModel() to call the tensorflow.js API `tf.loadLayersModel`

```
//-------------------------------------
// loader for cnn model
//-------------------------------------
```

```
async function loadModel() {
  // clear the model variable
  model = undefined;
  // load the model using a HTTPS request (where you have stored your model files)
  model = await tf.loadLayersModel("models/model.json");
}

loadModel();

```

Pre-process canvas

function preprocessCanvas to pre-process the canvas drawn by the user before feed it to the CNN model

```

//---------------------------------------------
// preprocess the canvas
//---------------------------------------------
function preprocessCanvas(image) {
    // resize the input image to target size of (1, 28, 28)
    let tensor = tf.browser.fromPixels(image)
        .resizeNearestNeighbor([28, 28])
        .mean(2)
        .expandDims(2)
        .expandDims()
        .toFloat();
    return tensor.div(255.0);
}

```

Prediction

When the “Predict” button is click, we get the image data from the canvas, pre-process it as a tensor, then feed it into the API `model.predict` to get the result of the prediction.

```
//---------------------------------------------
// predict function
//---------------------------------------------
$("#predict-button").click(async function () {
    // get image data from canvas
    var imageData = canvas.toDataURL();

    // preprocess canvas
    let tensor = preprocessCanvas(canvas);

    // make predictions on the preprocessed image tensor
    let predictions = await model.predict(tensor).data();

    // get the model's prediction results
    let results = Array.from(predictions);

    // display the predictions in chart
    $("#result_box").removeClass('d-none');
    displayChart(results);
```

```
        displayLabel(results);
    });
```

Display result

function loadChart to utilize the Chart.js library to display prediction result as a visual bar chart.

```
    //-----------------------------
    // Chart to display predictions
    //-----------------------------
    var chart = "";
    var firstTime = 0;
    function loadChart(label, data, modelSelected) {
        var ctx = document.getElementById('chart_box').getContext('2d');
        chart = new Chart(ctx, {
            // The type of chart we want to create
            type: 'bar',

            // The data for our dataset
            data: {
                labels: label,
                datasets: [{
                    label: modelSelected + " prediction",
                    backgroundColor: '#f50057',
                    borderColor: 'rgb(255, 99, 132)',
                    data: data,
                }]
            },

            // Configuration options go here
            options: {}
        });
    }

    //----------------------------
    // display chart with updated
    // drawing from canvas
    //----------------------------
    function displayChart(data) {
        var select_option = "CNN";

        label = ["0", "1", "2", "3", "4", "5", "6", "7", "8", "9"];
        if (firstTime == 0) {
            loadChart(label, data, select_option);
            firstTime = 1;
        } else {
            chart.destroy();
            loadChart(label, data, select_option);
        }
        document.getElementById('chart_box').style.display = "block";
    }

    function displayLabel(data) {
        var max = data[0];
```

```
        var maxIndex = 0;

        for (var i = 1; i < data.length; i++) {
            if (data[i] > max) {
                maxIndex = i;
                max = data[i];
            }
        }
        $(".prediction-text").html("Predicting you draw <b>"+maxIndex+"</b> with <b>"+
    }
```

Finally, testing

Simply include the scripts for `tfjs` in the <head> section of the html file. I also include the jquery library and the chart library as well.

```
<html>
<head>
    <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/
    <link rel="stylesheet" type="text/css" href="style/digit.css">
    <script src="https://code.jquery.com/jquery-3.3.1.min.js"></script>
    <script src="js/chart.min.js"></script>
    <script src="https://cdn.jsdelivr.net/npm/@tensorflow/tfjs@latest"></script>
</head>
```

Step 3 : Set up canvas

For the user to draw a digit using mouse on desktop or finger on mobile devices, we need to create a HTML5 element called canvas. Inside the canvas, the user will draw the digit. We will feed the user drawn digit into the deep neural network that we have created to make predictions.

HTML – index.html

Add a placeholder to contain the canvas that you can draw digit on

```
<div id="canvas_box" class="canvas-box"></div>
```

Add “Predict” button to get result of the hand written digit prediction, “Clean” button to wipe the canvas and start drawing again

```
<button id="clear-button" class="btn btn-dark">Clear</button>
<button id="predict-button" class="btn btn-dark">Predict</button>
```

At the end of the <body>, include the main javascript file digit-recognition.js

```
            <script src="js/digit-recognition.js"></script>
        </body>
</html>
```
