

# ENTRENANDO UNA RED NEURONAL PARA CAPTURAR UNA FOTO CON LA CÁMARA

## MANUAL DE DESARROLLO

Versión 1.0.0

Fecha: 23 / 05 / 2019

Diseño y Gestión de Software

Alejandro Guerrero T.





## ÍNDICE

1. INTRODUCCIÓN.....	3
2. DESCRIPCIÓN.....	3
2.1. Alcance.....	3
2.2. Funcionalidad.....	3
3. INTRODUCCIÓN AL SISTEMA.....	3
4. SOFTWARE UTILIZADO.....	3
5. PROCESO DE DESARROLLO.....	4
6. INGRESO AL SISTEMA.....	8
7. COMPONENTES DE INTERFAZ.....	9
7.1. Pantalla de Interacción.....	9
7.2. Sección de Cámara.....	9
8. USO DEL SISTEMA.....	10
9. HERRAMIENTAS ADICIONALES PARA DOCUMENTACIÓN Y EMPAQUETADO DEL PRODUCTO SOFTWARE.....	12
9.1. Bloc de Notas.....	12
9.2. Microsoft Word.....	12
9.3. WinRAR.....	12
10. ANEXOS.....	12
10.1. Guía rápida de botones.....	12
10. BIBLIOGRAFÍA.....	13



## 1. INTRODUCCIÓN

El presente documento tiene como objetivo detallar las herramientas utilizadas y el proceso para la construcción del producto software, las cuales son de código abierto, libres y gratuitas (FLOSS – *Free/Libre Open Source Software*).

## 2. DESCRIPCIÓN

### 2.1. Alcance

Con la creación de este documento se pretende guiar al usuario a la correcta utilización del producto software, incluyendo el manejo de las librerías implementadas; con la finalidad de que este pueda hacer uso de ellas al modificar su comportamiento acorde a sus necesidades.

### 2.2. Funcionalidad

Dentro de este documento encontrará las instrucciones necesarias para el desarrollo del producto, y de igual forma, que el usuario pueda probar el funcionamiento del proyecto en su máquina local para fines de desarrollo y prueba.

## 3. INTRODUCCIÓN AL SISTEMA

El presente proyecto consiste en implementar un software que permita entrenar una red neuronal con el objetivo de que pueda identificar entre dos objetos A y B.

Se mostrará el primer objeto frente a la cámara y se le enseñará al sistema a tomar una fotografía cuando este se encuentre presente. Posteriormente se hará lo mismo con el otro objeto, con la diferencia de que si este se encuentra frente a la cámara, no capturará una fotografía alguna.

## 4. SOFTWARE UTILIZADO

La base para la elaboración de este proyecto es el lenguaje de marcado HTML en su versión 5, el cual emplea etiquetas `<script></script>` y `<link>` las cuales serán usadas para insertar las tres librerías de código abierto que se detallan a continuación:

- [ml5.js](#)

Es una biblioteca de JavaScript, lanzada en el año 2018. ml5.js es desarrollada para hacer que el aprendizaje de máquinas sea más accesible para artistas, programadores creativos y estudiantes. Está construida en base a [TensorFlow.js](#), una biblioteca de JavaScript de código abierto.

Con ml5.js entrenaremos el modelo para que pueda distinguir entre los dos objetos que se enfocan frente a la cámara.[2]

- [p5.js](#)

Es una biblioteca de JavaScript que permite crear páginas web con elementos audiovisuales interactivos, controlados por los algoritmos realizados con ml5.js. [3]

Con esta herramienta crearemos un objeto *Canvas* (lienzo) para mostrar una captura de video y este sea controlado por los algoritmos hechos con ml5.js.



- **Bootstrap**

Es un Framework CSS que permite dar forma a un sitio web mediante librerías CSS que incluyen tipografías, botones, cuadros, menús y otros elementos que pueden ser utilizados en cualquier sitio web. [4]

Esta herramienta la utilizaremos para dar estilos a nuestro entorno visual, y que a su vez sea *responsive* (responsivo) para que pueda ser visualizado en distintas pantallas.

Otras herramientas a utilizar es:

- **Visual Studio Code**

Es un editor de código fuente desarrollado por Microsoft que incluye soporte para la depuración, control integrado de Git, resaltado de sintaxis, finalización inteligente de código (*intelligence*), fragmentos y refactorización de código. Con este poderoso editor crearemos y modificaremos todos nuestros archivos para el desarrollo del producto software. [5]

- **Web Server for Chrome**

Es una APP para el navegador Google Chrome que permite configurar un servidor local para páginas web alojadas en una carpeta local y que corra de manera *offline*. Con ella podremos probar el funcionamiento del software.

## 5. PROCESO DE DESARROLLO

5.1. En primer lugar, se importarán las tres librerías mencionadas al documento HTML dentro las etiquetas `<head>...</head>` como se muestra en la Figura 1.

```
<head>
  <meta charset="UTF-8">
  <title>Capturar una Foto</title>
  <link rel="shortcut icon" type="image/x-icon" href="./img/camera.png">

  <!-- Librerías p5.js para el diseño CANVAS del video -->
  <script src="./libraries/p5.min.js"></script>
  <script src="./libraries/p5.dom.min.js"></script>
  <script src="./libraries/p5.sound.min.js"></script>

  <!-- Librería ml5.js para realizar el entranamiento neuronal -->
  <script src="./libraries/ml5.min.js" type="text/javascript"></script>

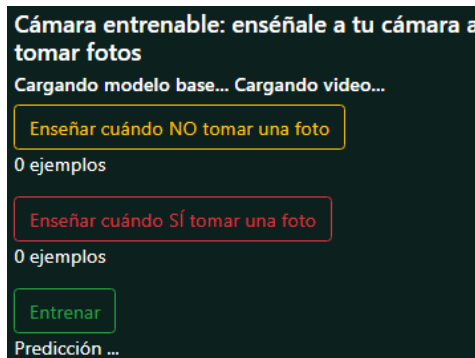
  <!-- Librería de estilos CSS de Bootstrap para los diseños -->
  <link rel="stylesheet" href="css/bootstrap.min.css">
</head>
```

Fig. 1. Importando las librerías.

5.2. Este software requiere de dos botones que permitan tomar muestras del objeto enfocado frente a la cámara. Un botón para decirle al modelo cuando tomar una foto, y otro botón para decirle



cuando. Para ello diseñaremos estos botones utilizando las etiquetas correspondientes, y aquí es donde emplearemos la librería de estilos CSS de Bootstrap que nos permite tener un diseño más elegante como se muestra en la Figura 2.



**Fig. 2.** Diseño de los botones para el entrenamiento.

Adicionalmente se requiere de un tercer botón para comenzar con el entrenamiento y obtener los resultados de la predicción.

Para obtener estos diseños basta con implementar cada una de las clases que están programadas en Bootstrap en cada una de las etiquetas correspondientes. Por ejemplo el código de la Figura 3 emplea la clase “*btn btn-outline-danger*”, la cual permite obtener el diseño mostrado en la Figura 2. De la misma forma se emplea las clases necesarias en cada uno de los objetos.

```
<button id="botonB" type="button" class="btn btn-outline-danger">Enseñar cuándo SÍ tomar una foto</button>
```

**Fig. 3.** Ejemplo de uso de clases Bootstrap para diseños.

Si desea saber que clases implementar y a qué objetos; como formularios, cabeceras, tarjetas, botones, carruseles, párrafos, etc. Se recomienda revisar la documentación en la página de [Bootstrap](https://getbootstrap.com/), ahí encontrará ejemplos de diseños para mejorar su sitio web.

**5.3.** En el código HTML agregamos una etiqueta `<div></div>` con el id “*videoContenedor*”, el cual contendrá un recuadro para mostrar la imagen que enfoca la cámara. Estos son los elementos necesarios para el entrenamiento neuronal, después se pueden personalizar la interfaz a fin de tener algo parecido como se ve en la Figura 4.



**Fig. 4.** Interfaz de usuario.



5.4. El archivo que controla toda la funcionalidad del programa, y que permite mostrar el video, entrenar el modelo usando las librerías p5.js y ml5.js se denomina *sketch.js*, por lo que debemos importarlo al documento HTML.

```
<script src="./js/sketch.js"></script>
```

Fig. 5. Importando el archivo *sketch.js*.

5.5. Este archivo empleará código JavaScript, para ello definiremos la función *Setup* la cual se ejecutará al momento de arrancar el programa y definirá ciertos parámetros de configuración iniciales. En esta función definiremos el lienzo para mostrar la imagen que enfoca la cámara el cual tendrá un tamaño y si lo agregará al elemento HTML que contiene el id “*videoContenedor*”, y de igual manera ejecutará una función llamada *configurarBotones()* cuyo contenido será mostrado posteriormente. El contenido de la función *Setup* se muestra en la Figura 6.

```
function setup() {  
  // crear lienzo  
  let miCanvas = createCanvas(500, 337);  
  miCanvas.parent('videoContenedor');  
  // densidad de pixeles de la pantalla  
  pixelDensity(1);  
  
  // crear una captura de video  
  video = createCapture(VIDEO);  
  video.size(500, 337);  
  video.hide();  
  
  // Agregar al elemento del DOM contenedorVideo  
  
  //video.parent("videoContenedor");  
  // Extract the already learned features from MobileNet  
  extractorCaracteristicas = ml5.featureExtractor('MobileNet', modeloListo);  
  // crear un nuevo clasificador usando esas caracteristicas y el video  
  clasificador = extractorCaracteristicas.classification(video, videoListo);  
  
  // crear imagen fuera del lienzo  
  imagenFueraLienzo = createGraphics(width, height);  
  
  // configurar botones interfaz de usuario  
  configurarBotones();  
}
```

Fig. 6. Contenido de la función *Setup()*.

5.6. Modificaremos el DOM en el HTML para que se muestren mensajes a penas se cargue el modelo, para ello implementamos las funciones de la Figura 7.

```
// función ejecutada cuando el modelo se carga  
function modeloListo() {  
  select('#estadoModelo').html('Modelo base (MobileNet) cargado!');  
}  
  
// función ejecutada cuando el video se carga  
function videoListo() {  
  select('#estadoVideo').html('Video listo!');  
}
```

Fig. 7. Funciones para modificar el DOM.



5.7. Luego definimos la función *configurarBotones()* la cual permite interactuar con el evento Click en cada uno de los botones. En esta función modificamos el DOM para que cada vez que se presione el botón aumente el número de muestras obtenidas, y de igual forma proceder al entrenamiento obteniendo el resultado final.

```
// funcion utilitaria para crear una interfaz de usuario
function configurarBotones() {
  // cuando se presiona el boton A, agregar el cuadro actual
  // del video con la etiqueta A al clasificador
  botonA = select('#botonA');
  botonA.mousePressed(function() {
    clasificador.addImage('A');
    select('#cantidadImagenesA').html(imagenesA++);
  });

  // cuando se presiona el boton B, agregar el cuadro actual
  // del video con la etiqueta B al clasificador
  botonB = select('#botonB');
  botonB.mousePressed(function() {
    clasificador.addImage('B');
    select('#cantidadImagenesB').html(imagenesB++);
  });

  // boton entrenar
  entrenar = select('#entrenar');
  entrenar.mousePressed(function() {
    clasificador.train(function(valorPerdida) {
      if (valorPerdida) {
        perdida = valorPerdida;
        select('#perdida').html('Pérdida: ' + perdida);
      } else {
        select('#perdida').html('Entrenamiento listo! Pérdida final: ' + perdida);
        clasificar();
      }
    });
  });
}
```

Fig. 8. Función *configurarBotones()*.

5.8. Finalmente para mostrar los resultados, implementamos la función *mostrarResultados()* la cual modificará el DOM como se muestra en la Figura 9.

```
// mostrar los resultados
function mostrarResultados(err, resultado) {
  // mostrar error si lo hay
  if (err) {
    console.error(err);
  }

  select('#resultado').html(resultado);

  resultadoClasificacion = resultado;

  clasificar();
}
```

Fig. 9. Definición de la función *mostrarResultados()*.



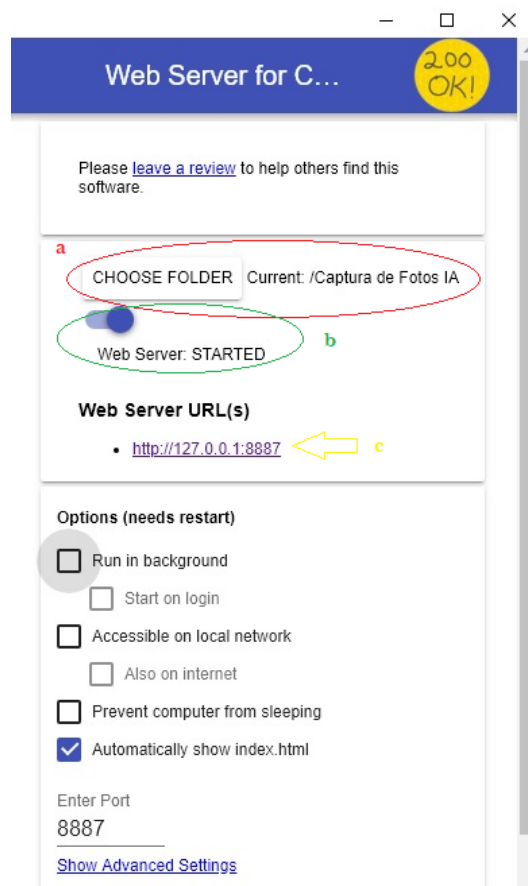
## 6. INGRESO AL SISTEMA

Para poder hacer uso del producto software, el usuario deberá haber copiado todos los archivos en un servidor web y ejecutar el archivo “*index.html*”

Si desea probar el funcionamiento del proyecto en su máquina local para fines de desarrollo y prueba, se recomienda utilizar un servidor local. Para este caso, haremos uso de Google Chrome y de la aplicación Web Server for Chrome.

En este apartado se explicará como ingresar en un servidor local. Para ello realizaremos los siguientes pasos:

- 1) Ejecutar la aplicación *Web Server for Chrome* el cual previamente deberá ser instalado en el navegador Google Chrome.
- 2) Una vez ejecutada la aplicación, se mostrará una imagen donde se deberá seleccionar la carpeta que contiene a todos los archivos del proyecto, encender el servidor y posteriormente ir a la dirección URL del Servidor Web como se puede apreciar en la Figura 10.



**Fig. 10.** Inicio del servidor local.

- 3) Una vez iniciado el servidor, este automáticamente iniciará ejecutando el archivo *index.html* donde se podrá visualizar la interfaz gráfica con la cual se procederá a interactuar. Figura 11.





## DISEÑO Y GESTIÓN DE SOFTWARE

### Entrenando una Red Neuronal para Capturar una Foto con la Cámara.

El presente proyecto tiene la finalidad de entrenar un modelo que tome una fotografía al mostrarle un objeto determinado, caso contrario no capturará nada.

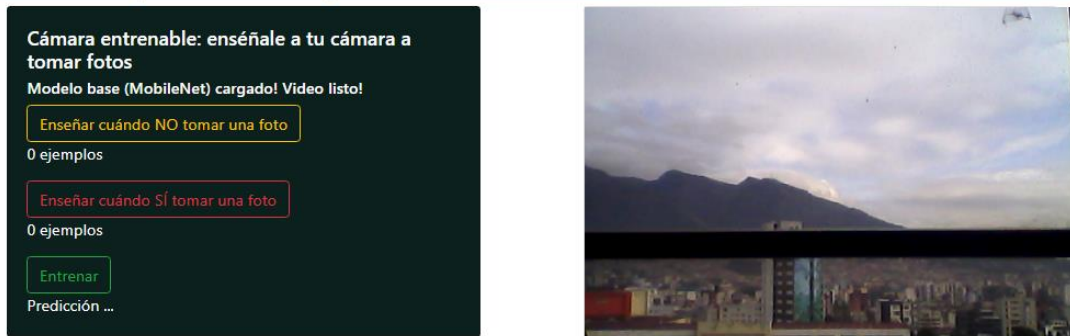


Fig. 11. Interfaz de inicio.

## 7. COMPONENTES DE INTERFAZ

### 7.1. Pantalla de Interacción

En la Figura 12, se puede observar cada uno de los botones que se usarán para entrenar a la red neuronal, donde el usuario puede interactuar con ellos de forma directa y visualizar cada uno de los resultados obtenidos al momento de realizar la predicción.



Fig. 12. Área de interacción.

### 7.2. Sección de Cámara

En la Figura 13 se tiene la sección correspondiente a la cámara, donde el usuario podrá enfocar cada uno de los objetos que se utilizarán para entrenar a la red neuronal.



Fig. 13. Sección de cámara.

## 8. USO DEL SISTEMA

Para poder explicar el uso del sistema, se mostrará brevemente un ejemplo que permita realizar el entrenamiento de la red neuronal. Para ello el usuario deberá realizar lo siguiente dentro de la interfaz:

- 1) Una vez visualizada la interfaz de inicio, el usuario deberá enfocar frente a la cámara uno de los dos objetos que desee tomar una fotografía cuando este se encuentre frente a la misma, y presionar varias el botón “*Enseñar cuando SI tomar una foto*”. El sistema tomará una muestra por cada pulsación de dicho botón; a mayor cantidad de muestras, mejor será la predicción. Esto se puede observar en la Figura 14 donde le enseñamos al modelo a capturar una fotografía cuando el teléfono celular esté frente a la cámara.

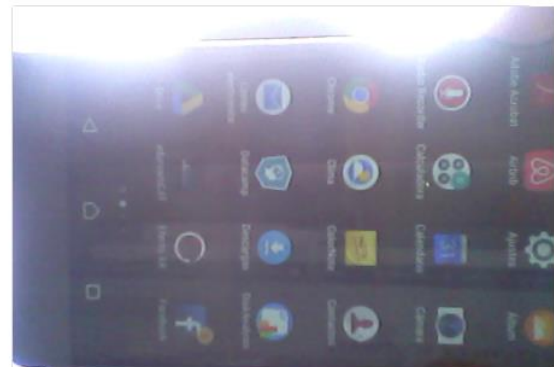


Fig. 14. Entrenando cuando capturar una fotografía.

- 2) Enfocar el segundo objeto, y realizar el mismo proceso, pero esta vez presionar el botón “*Enseñar cuando NO tomar una fotografía*”. En la Figura 15 le enseñamos al modelo a no capturar una fotografía cuando los lentes se encuentren frente a la cámara.



Fig. 15. Entrenando cuando no capturar una fotografía.



- 3) Una vez que haya ingresado un número de muestras que el usuario crea conveniente para objeto, deberá proceder a presionar el botón “*Entrenar*”. Inmediatamente el programa comenzará a realizar el aprendizaje mostrándole un valor de *Predicción* y su respectivo *Error* cuando finalice el proceso como se puede apreciar en la Figura 16.

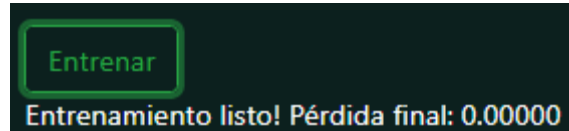


Fig. 16. Proceso completado de entrenamiento.

- 4) Una vez entrenado el modelo, este podrá distinguir entre uno de los dos objetos, que al momento de enfocarlo frente a la cámara procederá a tomar una fotografía de acuerdo al objeto que hay sido entrenado para dicho propósito, caso contrario no capturará nada. En la Figura 17 se puede apreciar que el modelo reconoce a los lentes y lo nombra como predicción “A”. Mientras que en la Figura 18, el modelo reconoce al teléfono celular como Predicción “B” y posteriormente capturará una fotografía de dicho objeto.



Fig. 17. Predicción “A”.

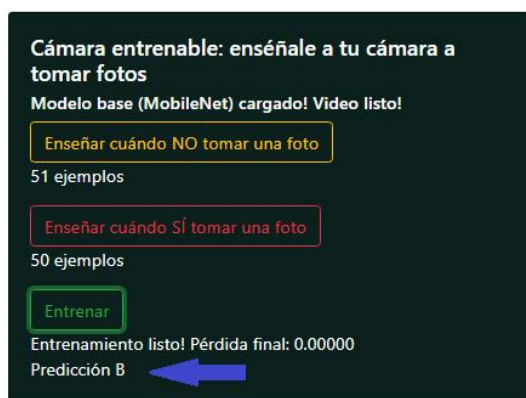


Fig. 18. Predicción “B”.

Como se puede observar en las figuras anteriores, la pérdida final es de 0.00000, debido a que se incluyeron la suficiente cantidad de muestras para el aprendizaje, facilitando así su mejor desempeño al momento de distinguir entre los dos objetos.



## 9. HERRAMIENTAS UTILIZADAS PARA LA DOCUMENTACIÓN

### 9.1. Bloc de Notas

Es un editor de texto incluido en los sistemas operativos de Microsoft cuya funcionalidad es muy simple y su extensión predeterminada es \*.txt.

Este editor fue utilizado para crear los archivos que contienen datos relevantes referentes al producto software, los cuales son:

- AUTHORS
- INSTALL
- LICENCE
- NEWS
- README

### 9.2. Microsoft Word 2016

Es un programa orientado al procesamiento de textos que ofrece la creación de documentos sencillos o profesionales con características potentes como herramientas de ortografía, gráficos, modelado de texto, plantillas, macros, etc.

Esta herramienta fue utilizada para crear este documento guía de desarrollo y uso del producto software.

### 9.3. WinRAR

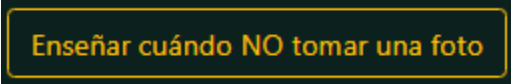
Es un software de compresión de datos. Posee un formato propio de compresión \*.rar, pero también es compatible con otros formatos como: ZIP, TAR, JAR, GZ, LZH, etc.

Con este programa comprimimos el producto entregable en el formato propio de WinRAR para hacer la entrega al cliente.

## 10. ANEXOS

### 10.1. Guía Rápida de Botones

#### a) “Enseñar cuando *NO* tomar una foto”



Enseñar cuándo NO tomar una foto

Este botón permite tomar muestras del objeto mostrado frente a la cámara, para que el modelo aprenda a no capturar una fotografía cuando este se encuentre presente.

#### b) “Enseñar cuando *SÍ* tomar una foto”

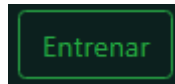


Enseñar cuándo SÍ tomar una foto

Este botón permite tomar muestras del objeto mostrado frente a la cámara, para que el modelo aprenda a capturar una fotografía cuando este se encuentre presente.



c) “Entrenar”



Este botón comienza a realizar el entramiento de la red neuronal en base a las muestras tomadas para cada uno de los objetos mostrados.

## 11. BIBLIOGRAFÍA

- [1] McCarthy, L. Reas, C. & Fry, B. (2016). “*Introducción a p5.js*”. Processing Foundation, Inc. United States of America.
- [2] “*ml5.js – Friendly Machine Learning for the Web*”. NYU ITP [Recuperado de:] <https://ml5js.org/>
- [3] “*p5.js - Referencia*”. Processing Foundation, Inc. [Recuperado de:] <https://p5js.org/es/>
- [4] “*Bootstrap v4.3.1 - Documentation*”. MIT. [Recuperado de:] <https://getbootstrap.com/>
- [5] Microsoft. “*Documentación for Visual Studio Code*”. [Recuperado de:] <https://code.visualstudio.com/docs> (2019)