2023

BANDEJA SENSORIAL AUTOMATIZADA

QUINTANILLA SAN MARTIN, CLAUDIO IGNACIO

UNIVERSIDAD TÉCNICA FEDERICO SANTA MARÍA

SEDE VIÑA DEL MAR - JOSÉ MIGUEL CARRERA

BANDEJA SENSORIAL AUTOMATIZADA

Trabajo de Titulación para optar al Título

de Técnico Universitario en ELECTRÓNICA

Estudiante:

Claudio Ignacio Quintanilla San Martín.

Profesor guía:

Ing. José Llantén Álvarez

Profesor correferente:

Ing. Sergio Riquelme Bravo.

2023

RESUMEN

KEYWORDS: EDUCACIÓN, BANDEJA SENSORIAL, VISIÓN POR COMPUTADORA.

La realización del proyecto está orientada a la integración de la tecnología en herramientas que son utilizadas en la educación. De esta forma, el enfoque del proyecto es mejorar un instrumento para el desarrollo de la lecto-escritura en niños, niñas, jóvenes y adultos con necesidades educativas especiales (tanto de carácter transitorias, como permanentes) o cualquier tipo de persona que requiera de dicha ayuda, el cual es conocido como Bandeja Sensorial. Cabe señalar, que la letra utilizada por el/la usuario/a deberá ser de tipo manuscrita, pues es el medio mayormente utilizado en las escuelas para comenzar con el proceso lecto-escritor.

El objetivo principal de este trabajo, es elaborar material didáctico que fomente la participación activa de los y las estudiantes en su proceso de aprendizaje lecto-escritor, para esto, se diseñará un sistema que complementará la Bandeja Sensorial, dotándola con Inteligencia Artificial, más específicamente, Visión por Computadora.

Asimismo, los objetivos específicos para la realización del proyecto consisten en la elaboración de la bandeja sensorial junto con el soporte dedicado para el celular, luego de esto se realizará una aplicación para dispositivos móviles con sistema operativo Android que permitirá activar la cámara del celular para la obtención de datos de lo que se estará digitando en la bandeja sensorial. Una vez realizada la aplicación, se implementará el modelo y los algoritmos de inteligencia artificial para la detección de objetos en tiempo real. Por último, se desarrollará la interfaz para mostrar la imagen de lo escrito o implementado en la bandeja sensorial.

Además, en el documento se describirán conceptos que guiarán a el/la lector/a para la completa comprensión del proyecto; entre los cuales se destacan Inteligencia Artificial (I.A), Visión por Computadora, con ayuda adicional de bibliografía complementaria que servirá de sustento teórico para el desarrollo del documento.

Finalmente, se destacan las subsecciones que contienen las temáticas más relevantes a desarrollar y/o comprender durante el transcurso del documento; las cuales son: ¿Qué es una bandeja sensorial?, Introducción a la Inteligencia Artificial, Opciones para la realización del proyecto, entre otras.

ÍNDICE

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

SIGLAS Y SIMBOLOGÍA

A. SIGLAS:

ML: Machine Learning (traducida al español, Aprendizaje de máquina).

APP: Aplicación.

AI: Artificial Inteligence (traducida al español, Inteligencia Artificial).

CV: Computer Vision (traducida al español, Visión de Computadora).

API: Application Programming Interfaces (traducida al español, Interfaces de Programación de Aplicaciones).

CPU: Central Processing Unit (traducida al español, Unidad Central de Procesamiento).

GPU: Graphic Processing Unit (traducida al español, Unidad de Procesamiento Gráfico).

B. SIMBOLOGÍA:

cm = Centímetros.

g = Gramos.

MB = MegaByte.

kB = KiloByte.

UF = Unidad de fomento.

INTRODUCCIÓN

En los últimos años se ha demostrado que es necesaria la integración de la tecnología en la educación, puesto que es una herramienta que permite crear plataformas para el desarrollo del aprendizaje a distancia, crear productos para el apoyo a el/la docente, elaborar material llamativo para los y las estudiantes, etc. Con esta idea en mente, se propone la incorporación de tecnología con inteligencia artificial para complementar el uso de una Bandeja Sensorial.

De este modo, el proyecto busca diseñar e implementar un sistema que sea capaz de entregar a el/la usuario/a un objetivo a realizar, por ejemplo, una imagen de la letra A y pueda visualizar, a través de una cámara, lo que se esté realizando en la Bandeja Sensorial; para así detectar si la letra, forma o figura se ha replicado correctamente. En caso que se haya detectado la letra y comparada con la inicial, se le debe indicar a el/la usuario/a, si lo ha realizado con o sin éxito. Cabe señalar, que la letra utilizada por el/la usuario/a deberá ser de tipo manuscrita, pues es el medio mayormente utilizado en las escuelas para comenzar con el proceso lecto-escritor. Una vez que se haya notificado el resultado de la comparación, el sistema debe ser capaz de mostrar otro objeto o figura, en su defecto, ofrecer la posibilidad de volver a intentar la misma figura.

Primero, para la completa comprensión del proyecto, ROUHIAINEN (2018) define como Inteligencia Artificial a la capacidad de las máquinas para usar algoritmos, aprender de los datos y utilizar lo aprendido en la toma de decisiones tal y como lo haría un ser humano. Lo anterior, se incluye dentro de una amplia gama de definiciones para dicho concepto, pues éste es de carácter polisémico.

Ahora bien, en el proyecto en cuestión, dicha Inteligencia Artificial (I.A) puede ser desarrollada para obtener los resultados requeridos. Para esto, es necesario indagar en el uso de un campo de desarrollo de la I.A. conocido como Aprendizaje Automático (Machine Learning), y más específicamente, el estudio de la Visión por Computadora (Computer Vision). Tal concepto, a grandes rasgos, es la capacidad que tiene una máquina de computación para obtener información

de imágenes y realizar la detección, clasificación, comparación, etc., de este ingreso de datos, con la base del Modelo contenido en la memoria.

Asimismo, el uso de la Inteligencia Artificial (I.A) incorporado en la bandeja sensorial, se evidencia en la necesidad que se tiene por lograr la detección de letras.

Por otra parte, la motivación de realizar este proyecto nace a partir de la urgencia por incorporar el uso de las tecnologías al área de la educación, con el fin de proporcionar un espacio educativo más cercano, innovador y amigable para todos/as los/as estudiantes con algún tipo de necesidad educativa especial o alguna persona que requiera dicha asistencia (jóvenes y/o adultos)

Es importante señalar, que con la ayuda de herramientas tecnológicas es posible utilizar materiales que sean llamativos para las y los estudiantes, generando un mayor interés por parte del estudiantado. De igual modo, considerando el impacto ambiental con respecto al uso del papel para la exposición del objetivo a digitar en la bandeja sensorial, el proyecto busca aportar en la disminución del uso excesivo de este recurso para complementar el manejo del instrumento.

CAPÍTULO 1: INTEGRACIÓN DE TECNOLOGÍA EN LA EDUCACIÓN

En esta sección del trabajo se detallarán los conceptos fundamentales para entender el propósito y funcionamiento del proyecto, abordando temáticas relacionadas con la educación y la tecnología de Inteligencia Artificial. Junto con esto, se expondrán dos opciones para la realización del proyecto, cada una con sus ventajas y desventajas, para luego seleccionar la que mejor se acomode a las necesidades tanto del desarrollador como de los/las posibles usuarios/as. Por último, se presentará un diagrama de bloques que permitirá guiar el proceso de elaboración del proyecto.

1.1 ¿QUÉ ES UNA BANDEJA SENSORIAL?

Para el correcto entendimiento del proyecto, primero se debe exponer la definición y el propósito de una Bandeja Sensorial.

Una bandeja sensorial es una herramienta muy sencilla, que está compuesta por un contenedor (caja de cartón o madera) en la que se vierte un material granulado (arena, azúcar, sémola, etc.) y se utiliza para realizar formas, figuras, letras, etc.

Esta herramienta, se utiliza para el trabajo de la grafo-motricidad y la facilitación del desarrollo de habilidades de lecto-escritura en niños, niñas, jóvenes y adultos con necesidades educativas especiales o cualquier persona que requiera de esta asistencia.

En la imagen 1-1 se presenta una imagen representativa del uso de una bandeja sensorial.

Fuente: https://www.creciendoconmontessori.com/2021/01/como-usar-la-caja-de-arena-montessori-imprimible-grafomotricidad-incluido.html

Figura 1-1: Bandeja Sensorial siendo utilizada

Junto con este instrumento de trabajo y la necesidad de crear material didáctico que fomente la participación activa de los y las estudiantes en su proceso de aprendizaje, es que surge la idea de integrar la tecnología en esta herramienta para la educación.

1.2 RAZÓN DEL PROYECTO

La necesidad de realizar este proyecto, nace a partir de la urgencia por incorporar el uso de las tecnologías al área de la educación; con el fin de proporcionar un espacio educativo más cercano, innovador y amigable para todos/as los/as estudiantes con algún tipo de necesidad educativa especial o cualquier persona que requiera de esta asistencia.

De igual modo, considerando el impacto ambiental con respecto al uso del papel para la exposición del objetivo a digitar en la bandeja sensorial, el proyecto busca aportar en la disminución del uso excesivo de este recurso para el complementar el manejo del instrumento.

Además, el proyecto anhela impulsar la innovación (en el entorno universitario) enfocando sus directrices en el área educativa, involucrando en su desarrollo los avances de la ciencia de la Inteligencia Artificial.

1.3 INTRODUCCIÓN A LA INTELIGENCIA ARTIFICIAL

La **Inteligencia Artificial** o **A.I**. (proveniente del inglés: Artificial Inteligence) es una nueva ciencia de la computación, que tiene como objetivo imitar las capacidades físicas y psicológicas de los/las seres humanos/as, a través de la simulación con algoritmos matemáticos. Estos sistemas son capaces de aprender y de tomar decisiones adecuadas dependiendo del tipo de tarea al cual están destinados.

En el estudio de la I.A. existe un campo de investigación conocido como **Aprendizaje de Máquina** o **Machine Learning**. Este se basa en el análisis de cómo las computadoras pueden obtener conocimientos o habilidades, a través de la reproducción del comportamiento de aprendizaje humano. Estos sistemas son diseñados para encontrar patrones dentro de un conjunto de datos, lo que permite realizar tareas tales como la comparación, la clasificación, la detección, entre otras.

Ahora bien, existen tres tipos principales de aprendizaje automático, estos son:

1. Aprendizaje Supervisado: Este tipo de algoritmo utiliza un conjunto de datos previamente etiquetado que le permite al sistema obtener las directrices para categorizar la nueva información. En estos modelos es indispensable la intervención humana para el etiquetado del conjunto de datos. Un ejemplo práctico sería (similar al caso de este trabajo) mostrar imágenes de perros para que el sistema pueda aprender y posteriormente detectarlos en imágenes futuras.
2. Aprendizaje no supervisado: En el aprendizaje no supervisado, los algoritmos son capaces de detectar patrones dentro del conjunto de datos sin necesidad de un etiquetado previo, eliminando así, la intervención humana dentro del aprendizaje de la máquina. Tomando

el ejemplo anterior, se le muestra al sistema el conjunto de imágenes de perros sin etiquetar y éste debería ser capaz de categorizar las fotografías según los patrones que se repitan de los canes.

3. Aprendizaje por refuerzo: En el aprendizaje por refuerzo, como indica su nombre, los algoritmos deben ser reforzados según el desempeño que están obteniendo, así, si la salida del sistema es acertada, se debe dar un refuerzo positivo.

   Este tipo de aprendizaje es comparable a situaciones de la vida cotidiana como, por ejemplo: entregar dulces por conducta positiva a un/a niño/a con la intensión de modificar o motivar una actitud.

Si se enfoca el Modelo de Aprendizaje de la I.A. al objetivo específico de la detección de objetos, éste debería contener un conjunto de datos con las imágenes del objeto que se desea detectar y las coordenadas puntuales de la ubicación del elemento (etiquetado). Contrario a esto, se le presentan imágenes en donde el objetivo no está, con esta información se le da a entender a la máquina qué es lo que debe encontrar al momento de recibir y procesar los datos.

Empresas como Google y Facebook, están optando por desarrollar entornos de A.I. TensorFlow y Pythorch respectivamente, que sean sencillas de utilizar y estén disponibles para todo aquel que desea integrar estas tecnologías en la resolución de problemas, o agregar funciones a sus productos.

1.4 OPCIONES PARA LA IMPLEMENTACIÓN DEL PROYECTO

Para el proceso de implementación y ejecución del proyecto, se consideraron dos alternativas principales, las cuales se detallan a continuación.

1.4.1 Implementación con el uso de una tarjeta microcontroladora

La primera opción, se basa principalmente en el trabajo con una placa Raspberry PI 3B que contendrá los algoritmos de inteligencia artificial (visión por computadora) y la base de datos del modelo a utilizar.

Luego, esta tarjeta microcontroladora estará conectada a una cámara para recopilar información del entorno, y posteriormente realizará la detección de objetos, adicionalmente, se debe contar con una pantalla para la visualización del objeto (letra) a replicar y en el caso que se quiera integrar una señal sonora para la indicación de la detección, se deben considerar elementos de emisión sonora como Buzzers y un conversor de niveles para su correcto funcionamiento.

Elementos necesarios para la implementación.

- **Placa Raspberry PI 3B:** Es una tarjeta de desarrollo que funciona como un pequeño computador, otorgando la posibilidad de trabajar en forma remota o directamente con ella, sin la necesidad de utilizar un computador de escritorio o notebook. Si solo se desea trabajar con la tarjeta microcontroladora, se debe tener presente que es necesario tener los periféricos esenciales para el trabajo en ordenadores, siendo estos un mouse, un teclado y una pantalla.

  Una gran ventaja del uso de los microcontroladores, es la posibilidad de agregar funcionalidades con módulos independientes, como lo sería una cámara, sensores, pantalla, etc. En la figura 1-3 se muestra la imagen de una Raspberry modelo PI 3B.

Fuente: https://www.mcielectronics.cl/page/homepage

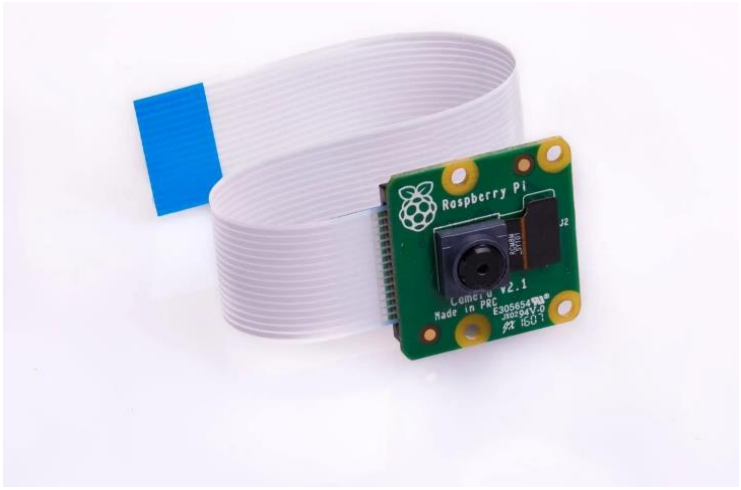
Figura 1-2: Tarjeta microcontroladora Raspberry PI 3B

- **Pantalla:** Es indispensable el uso de una pantalla para la visualización de las acciones u operaciones que se realiza con la tarjeta Rapsberry y de la proyección de la imagen objetivo que se quiere realizar. En la figura 1-4 se muestran imágenes de una pantalla para ser utilizada en una Raspberry.

Fuente: https://www.mcielectronics.cl/page/homepage

Figura 1-3: Pantalla dedicada para tarjeta microcontroladora Raspberry Pi 3.

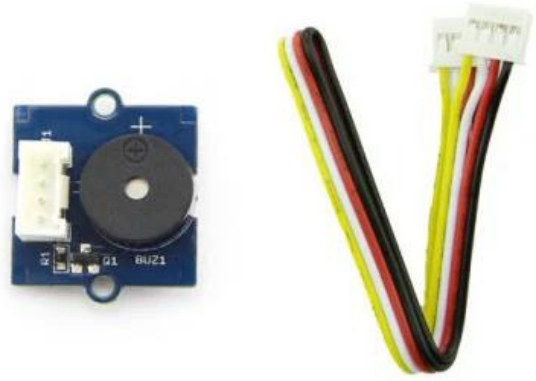
- **Cámara:** El completo desarrollo del proyecto se basa en el uso de una cámara para realizar la detección de objetos. Ésta será la que permita capturar la información del entorno y enviarla a la tarjeta Rapsberry para que sea procesada. En la figura 1-5 se muestra la imagen de una cámara especialmente diseñada para ser usada con dispositivos Raspsberry,



Fuente:https://www.mcielectronics.cl/page/homepage

Figura 1-4: Cámara dedicada para tarjeta microcontroladora Raspberry.

- **Buzzer:** Para complementar el funcionamiento del detector de objetos, se requiere de un dispositivo generador de sonido. En la figura 1-6 se muestra la imagen de un CI buzzer presente en el mercado.



Fuente: https://www.mcielectronics.cl/page/homepage

Figura 1-5: Circuito integrado de buzzer.

1.4.2 Análisis de la propuesta

A partir del análisis de esta opción, se logró encontrar las ventajas y desventajas que implicarían utilizar esta alternativa de desarrollo.

Entre las ventajas se encuentran:

- **La posibilidad de trabajar en un entorno orientado a la programación en Python:**
  Python es un lenguaje de programación interpretado, que es sencillo de entender y programar.
- **Interacción directa entre la tarjeta microcontroladora y la cámara:**
  Esta interacción directa, resulta muy útil para facilitar el control y manejo de datos que captura la cámara.

Por otro lado, en las desventajas se destacan:

- **Necesidad de considerar elementos externos para realizar funciones específicas**
  Como poder incorporar avisos sonoros para la identificación del resultado del intento.

- **Costo monetario.**
  Debido a la gran cantidad de herramientas que se deben usar, el costo de elaborar esta línea de ejecución es alto.
  Escasa portabilidad.
  El ejecutar este proyecto con tal cantidad de instrumentos, repercute negativamente en la capacidad de portabilidad. Considerando el caso en el que todos estos elementos se puedan integrar en un contenedor, la necesidad de suministrar energía a la tarjeta microcontroladora y los componentes externos, lo hace dependiente del uso cercano a sistemas de alimentación como tomas de corriente desde la red eléctrica.
- **Imposibilidad de ofrecerlo como material para el apoyo en la educación, por la naturaleza experimental del proyecto.**
  Al tratarse de un modelo de trabajo experimental o prototipo, es imposible ofrecer una versión posible de ser distribuida a su público objetivo. Pero luego de una evaluación, esto podría realizarse si los resultados son evaluados positivamente.

1.4.3 Implementación como una App de Android

La segunda opción, estructura el proyecto como un producto que sea de fácil distribución y alcance para sus posibles usuarios/as.

De esta forma, se elaborará una aplicación para dispositivos móviles con sistema operativo Android que, a través de la cámara frontal del dispositivo, será capaz de detectar letras, figuras, formas, etc. Esta aplicación, estará dotada con algoritmos de inteligencia artificial, específicamente, de visión por computadora; para así detectar las figuras que se repliquen en la Bandeja Sensorial.

1.4.3.1 Elementos necesarios para la elaboración:

La mayoría de herramientas a utilizar para la realización de esta propuesta son de carácter digital, más específico, programas o softwares de acceso gratuito. A continuación, se indica una lista mínima:

- **Computador:** Es necesario tener un equipo con adecuadas prestaciones técnicas para realizar la programación de la aplicación. En la figura 1-7 se muestra la imagen de un ordenador genérico a modo de ejemplo.

Fuente: https://www.kemik.gt/comprar/computadora-dell-optiplex-small-form-factor/

Figura 1-6: Computador de escritorio.

- **Android Studio:** Programa dedicado para la creación de aplicaciones en el ambiente de desarrollo de sistemas operativos Android. En la figura 1-8 se observa el logo de Android Studio

Fuente: https://www.freepng.es/png-hobovw/download.html

Figura 1-7: Logo del Software de programación para dispositivos Android.

- **TensorFlow y TensorflowLite:** Entorno de desarrollo para el aprendizaje automático desarrollado por Google y su versión para dispositivos móviles. En la figura 1-9 se muestran los logos de ambos entornos de desarrollo.

Fuente: https://www.pngwing.com/en/free-png-ddfyl

Figura1- 8: Logo del Software de aprendizaje automático TensorFlow

- **ML kit:** Librerías de Inteligencia Artificial optimizadas para el desarrollo de aplicaciones en dispositivos móviles. En la figura 1-10 se muestra el logo de ML kit.

Fuente: https://4.bp.blogspot.com/-ikzQvn-zORs/XvFITOxjstI/AAAAAAAAPHw/OFKAnXM_5gQOaDKUzbDgVroPTcCC9GBVQCLcBGAsYHQ/s1600/mlkit-logo-text.png

Figura 1-9: Logo de las librerías de M.L. optimizadas para dispositivos móviles.

- **Smartphone:** Dispositivo móvil para el que se desarrollará la aplicación. Este debe contar con una cámara frontal, para así captar lo que se desarrolla en la Bandeja Sensorial a la vez que se muestra el objetivo que se tiene que digitalizar. En la figura 1-11 se muestra una imagen de un Smartphone que cumple con las condiciones para ejecutar el programa.

Fuente: https://www.amazon.com/-/es/Smartphone-completamente-desbloqueado-T-Mobile-Tracfone/dp/B0844X3STN

Figura 1-10: Smartphone con sistema operativo Android.

1.4.4 Análisis de la propuesta

Analizando detalladamente esta opción, se lograron encontrar las ventajas y desventajas que implicarían utilizar este camino de desarrollo.

Entre las ventajas se encuentran:

- **El costo monetario.**

  El desarrollo de aplicaciones para sistemas operativos Android es prácticamente gratuito al igual que las librerías para la implementación de Inteligencia Artificial.

- **Elevada portabilidad.**

  El proyecto, al estar en su mayoría contenido en una aplicación para dispositivos móviles, es altamente portable, por lo que solo se debe considerar el traslado de la Bandeja Sensorial.

- **Elevado margen de mejora.**

  Capacidad de implementar nuevas funciones que permitan una mayor interacción con el/la usuario/a, una interfaz más amigable y de fácil uso, animaciones atractivas y/o llamativas para los/las usuarios/as.

- **Posibilidad de integrar una interfaz para la interacción entre el/la usuario/a y la App.**

  Esta interfaz permitirá una amena interacción de los/las usuarios/as con la aplicación y tendrá la capacidad de mostrar el objetivo a digitar en la pantalla del dispositivo.

- **De fácil acceso para sus posibles usuarios/as.**

  En caso que se distribuya al interior de la tienda de aplicaciones, sus posibles usuarios/as tendrán la facilidad de descargarla desde esa plataforma para los dispositivos móviles en forma gratuita. De igual modo, dicha aplicación se ofrecerá como material de apoyo para la educación, proveyendo a los/as docentes de una herramienta pedagógica para lograr el aprendizaje del estudiantado.

Por otro lado, las desventajas más significativas, se presentan a continuación.

- **Necesidad de adquirir conocimientos del desarrollo de aplicaciones en Android.**

  El entorno de desarrollo del sistema operativo Android se basa en una programación orientada a objetos escrita en lenguaje Java/Kotlin, por lo que es necesario entender claramente esta forma de programación para lograr el objetivo propuesto.

- **Necesidad de manejar los lenguajes de programación en el entorno de Android, específicamente Java y Kotlin.**

  Toda la programación de las aplicaciones en Android se escribe en lenguaje Java y con su sucesor Kotlin.

- **Aplicación disponible solo para dispositivos con sistema operativo Android.**

La aplicación para dispositivos móviles no ofrecerá cobertura al sistema operativo IOS, por lo que el alcance que tiene hacia la población podría ser limitado.

1.4.5 Selección de la alternativa para ejecutar el proyecto

Luego de proponer y analizar las opciones para la elaboración del proyecto, se logra concluir que la mejor alternativa es optar por realizar una aplicación para dispositivos móviles con sistema operativo Android.

Con esto definido, el objetivo general del proyecto será elaborar una aplicación para dispositivos móviles con sistema operativo Android, que a través de su cámara frontal, será capaz de detectar letras. Esta aplicación, estará dotada con algoritmos de inteligencia artificial, específicamente, de visión por computadora; para así detectar las figuras que se reproduzcan en la bandeja sensorial.

Como objetivos específicos se plantean:

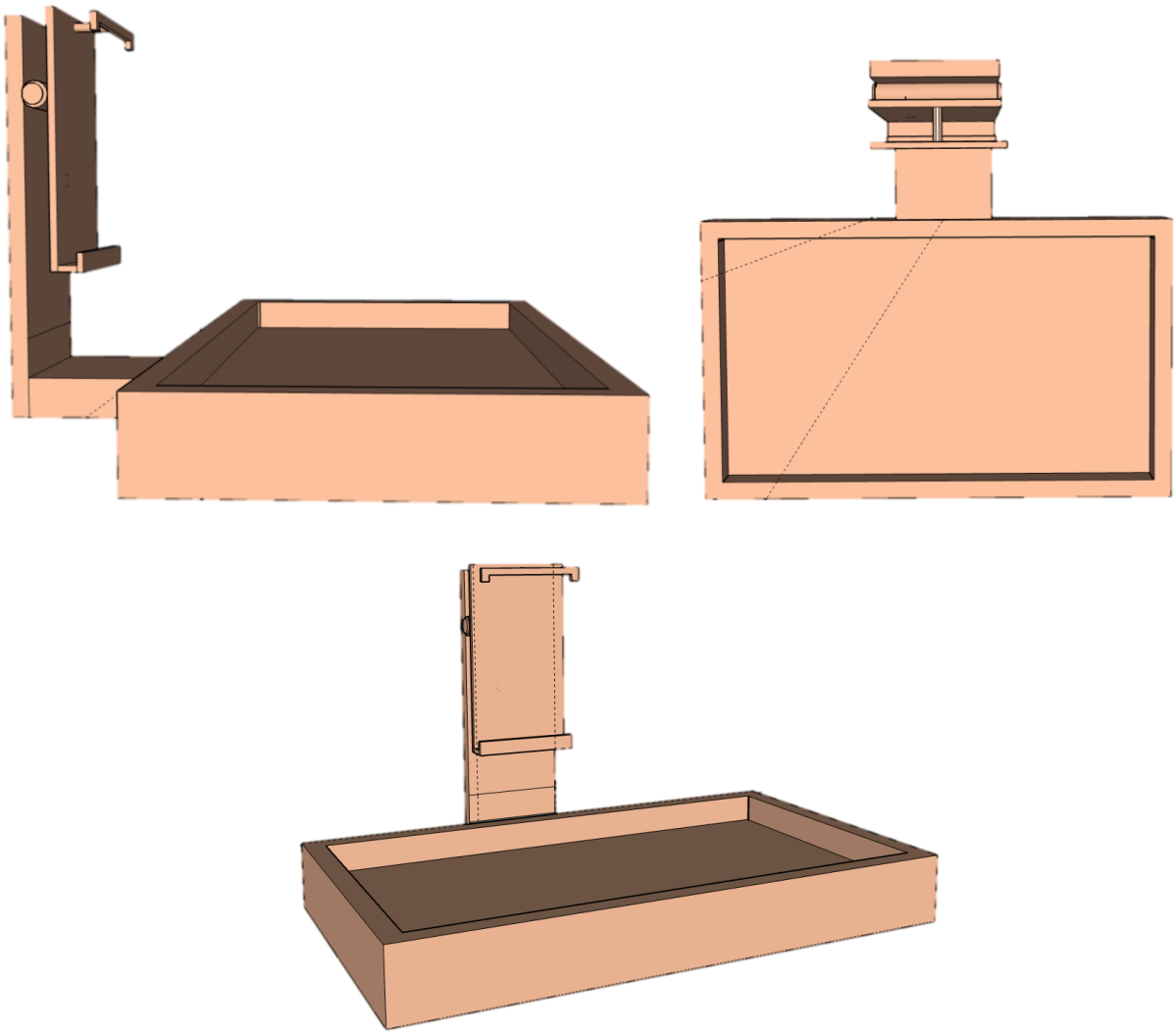
- Elaboración de la bandeja sensorial junto con el soporte dedicado para el celular.
- Desarrollar una aplicación en Android que permita activar la cámara del celular para la obtención de datos.
- Implementar las librerías y el modelo de aprendizaje automático en la aplicación, para la detección de objetos en tiempo real.
- Desarrollar una interfaz para mostrar la imagen del objetivo a realizar en la bandeja sensorial.

1.5 IMPLEMENTACIÓN DE LA BANDEJA SENSORIAL

En esta etapa del proyecto, se entregan las pautas de implementación de la Bandeja Sensorial.

Para elaborar el recipiente contenedor o bandeja, se utilizará madera como material principal, puesto que es resistente y de bajo costo. Sus medidas serán de aproximadamente 30cmx50cmx7cm y el soporte que sostendrá al celular será de 20cmx10cm.

En la figura 1-2, se presenta un prototipo inicial del diseño de la bandeja sensorial.

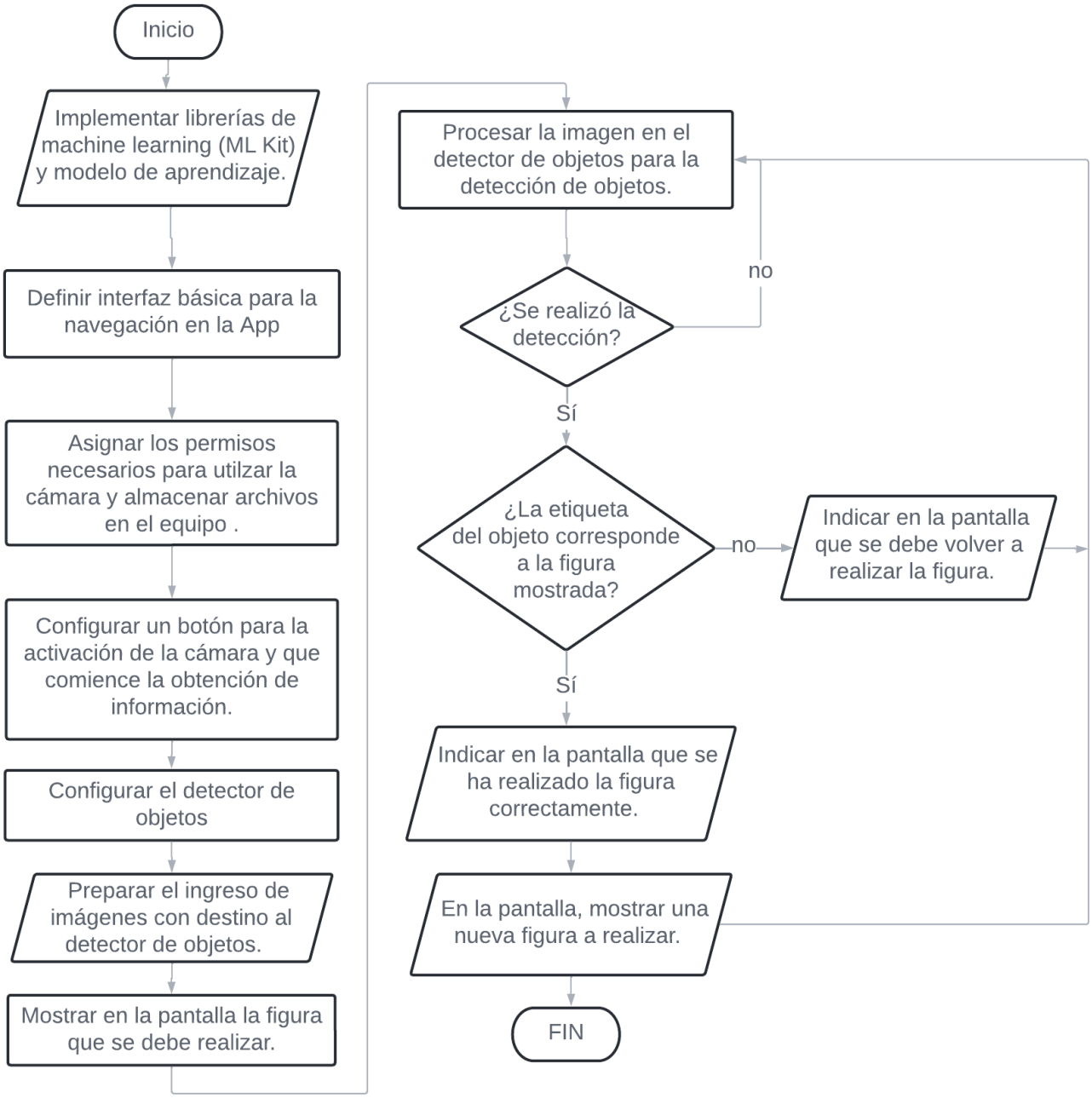


Fuente: Creación realizada por el estudiante en el software Sketchup.

Figura 1-11: Prototipo inicial de Bandeja Sensorial

1.6 DIAGRAMA DE FLUJO DE LA PROGRAMACIÓN DE LA APLICACIÓN

En la figura 1-12, se detalla un diagrama de flujo que permite exponer la programación que se realizará para la aplicación.

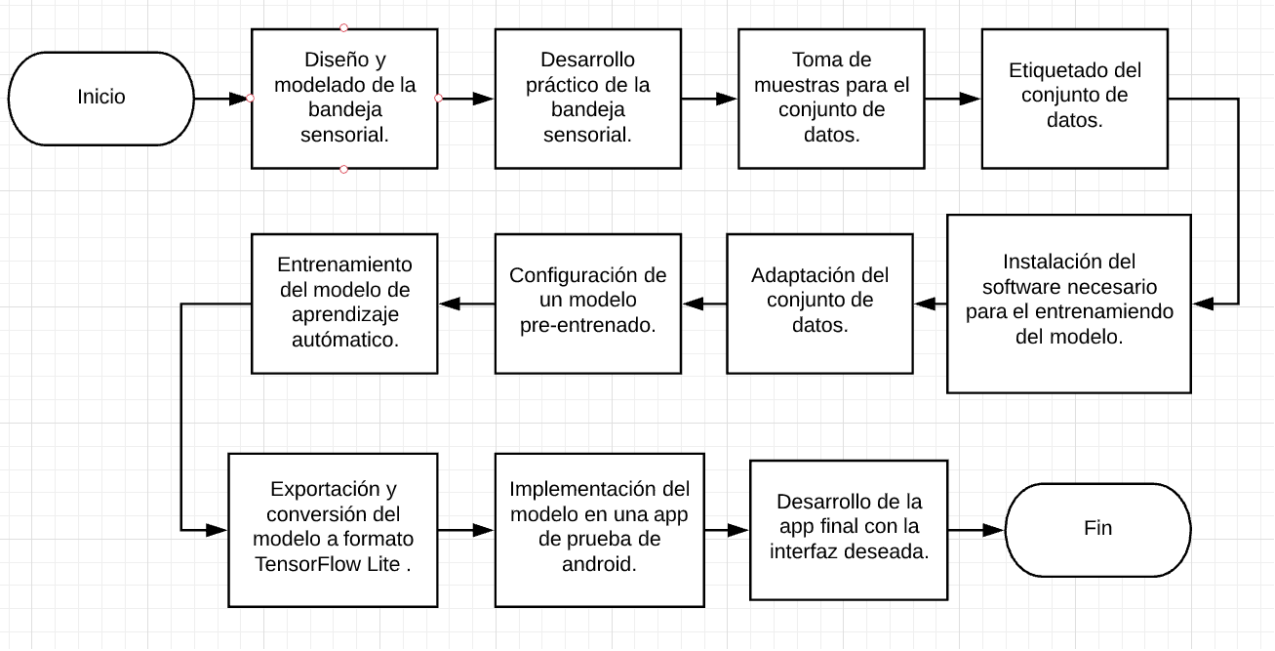


Fuente: Creación elaborada por el estudiante en Lucidchart.

Figura 1-12: Diagrama de bloques de la programación.

CAPÍTULO 2: ANÁLISIS Y DESARROLLO PRÁCTICO DE LA BANDEJA SENSORIAL AUTOMATIZADA

En el presente capítulo se detallará el análisis y desarrollo práctico de la bandeja sensorial automatizada, incluyendo la evaluación del prototipo de la bandeja sensorial antes descrito, su implementación y el desarrollo del modelo de aprendizaje automático que será necesario para realizar las predicciones en la aplicación para sistemas operativos Android, concluyendo con la implementación de este modelo en la aplicación. La figura 2-1, presenta un diagrama de bloques que grafica el proceso de desarrollo del proyecto.



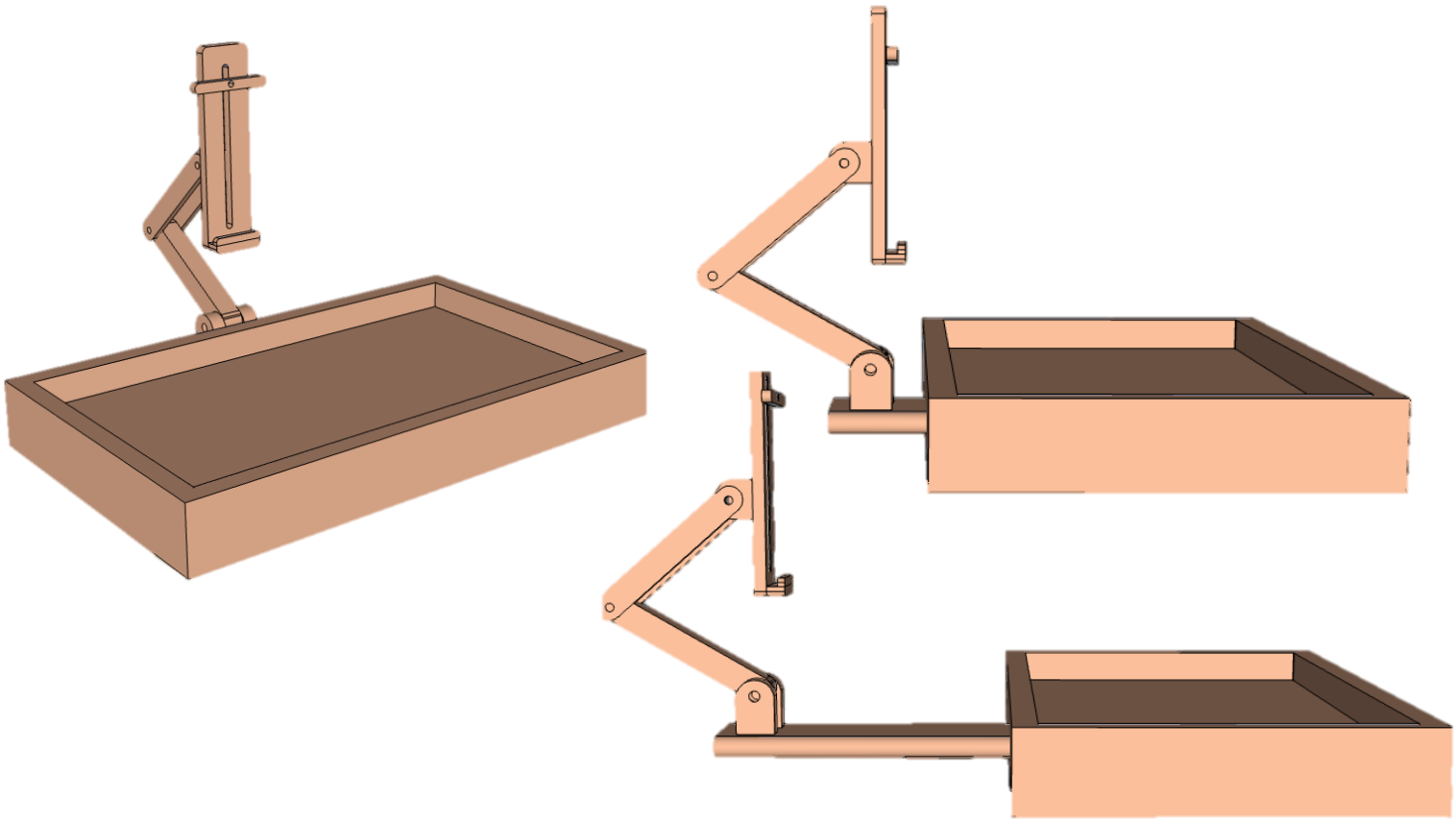
Fuente: Creación realizada por el estudiante en el software Lucidchart.

Figura 2-1: Diagrama de bloques del proyecto.

2.1 MEJORAMIENTO Y ELABORACIÓN DEL PROTOTIPO DE LA BANDEJA SENSORIAL

Al momento de analizar el prototipo propuesto y de cómo se comportaría en la práctica, se concluye que es necesario realizar algunos cambios en el soporte de celular para ubicar en forma correcta el dispositivo (refiriéndose con esto, a la posición en que la cámara frontal del celular pueda visualizar la totalidad de la bandeja y, a la vez, el/la usuario/a debe tener completa visibilidad de lo que se estará mostrando en la pantalla del dispositivo). Debido a esto, se diseñó un nuevo modelo para la bandeja sensorial, en el que el soporte para celular tiene un mayor ajuste de su posición, y así puede acomodarse mejor a la variedad de dispositivos existentes cumpliendo con el requisito anteriormente mencionado.

En la Figura 2-2 se presenta el modelo actualizado con las mismas dimensiones detalladas anteriormente.



Fuente: Creación realizada por el estudiante en el software Sketchup.

Figura 2- 2: Modelo actualizado de la bandeja sensorial.

Ahora bien, en el proceso de implementación, se utilizaron materiales reciclados, tales como: retazos de terciado de pino, madera de pino y piso laminado.

En la Figura 2-3, se observa el terciado de pino reciclado.

Fuente: Fotografía realizada por el estudiante.

Figura 2-3: Retazos de terciado de pino reciclados.

En la tabla 2-1, se encuentran las herramientas que fueron necesarias para implementar la bandeja sensorial.

Tabla 2-1: Herramientas necesarias para la construcción de la bandeja sensorial.

Nombre de la herramienta	**Función**
Sierra circular	La sierra circular se utilizó para dimensionar la estructura de la bandeja sensorial.
Sierra caladora	La sierra caladora se utilizó para realizar los cortes finos en las partes que componen el soporte para celular
Fresadora de palma	Esta herramienta se utilizó para realizar canalizaciones en la estructura de la bandeja sensorial y para definir las terminaciones.

Huincha de medir	Se utilizó para medir y dimensionar las partes que componen la bandeja sensorial.
Lápiz mina	Fue utilizado para marcar las dimensiones del proyecto.

Fuente: Tabla realizada por el estudiante.

Ya en el proceso de construcción, primero se dimensionó la estructura de la bandeja con las medidas estipuladas y luego se procedió a cortar los trozos de madera con las esquinas en 45° para realizar una unión limpia y sin la necesidad de utilizar tornillos. Luego, se realizó una canal en la parte inferior de cada trozo de madera (utilizando la fresadora) para que así se pueda ubicar el piso laminado, que será la base de la estructura.

A continuación, en Figura 2-4, se observan los trozos de madera ya dimensionados con las canales y el piso flotante que se utilizó para la base.

Fuente: Fotografía realizada por el estudiante.

Figura 2-4: Piezas que componen la bandeja sensorial.

En relación al armado estructural de la Bandeja Sensorial, se utilizó, como material de sujeción, cola fría para pegar las uniones.

En la siguiente imagen, Figura 2-4, se visualiza el armado y pegado de la bandeja sensorial, la cual fue sujetada con pasadores de mezclilla para que mantuviera una posición correcta mientras el pegamento se secaba.

Fuente: Fotografía realizada por el estudiante.

Figura 2-5: Bandeja sensorial en proceso de armado y secado.

Ya con el pegamento seco, se elaboró el sistema que permitirá el ajuste de cercanía del soporte con el/la usuario/a, el cual consiste en dos guías ubicadas en la parte inferior de la bandeja, con un orificio que permite el desplazamiento y fijación del soporte, lo que se muestra en la Figura 2-6.

Fuente: Fotografía realizada por el estudiante.

Figura 2-6: Creación del sistema de fijación para el soporte.

Del mismo modo, se procedió al dimensionamiento de las partes que conforman el soporte para dispositivos móviles, el cual fue elaborado a partir de retazos de terciado de pino reciclado y madera de pino, para la sujeción de las articulaciones.

En la figura 2-7, se encuentran todas las partes que componen el soporte para dispositivos móviles, compuesto por los pernos y tuercas que realizarán la fijación, y las partes articuladas que darán al soporte la flexibilidad para su correcto ajuste.

Fuente: Fotografía realizada por el estudiante.

Figura 2-7: Piezas que componen el soporte para dispositivos móviles.

En la Figura 2-8 a continuación, se presenta el soporte ya ensamblado y listo para su integración en la bandeja sensorial.

Fuente: Fotografía realizada por el estudiante.

Figura 2-8: Soporte para dispositivos móviles.

Ya con las partes fundamentales elaboradas y el soporte ensamblado, se debe integrar este último en la bandeja para completar el armado final. Una vez realizado, el siguiente paso es llenar el interior de la bandeja con el material granulado, que en este caso será sémola, para así poder dar uso al instrumento y comenzar con la toma de muestras para la creación del conjunto de datos que será utilizado para entrenar el modelo de aprendizaje automático.

En la Figura 2-9 se muestra la bandeja sensorial ya con el soporte instalado y lista para su uso.

Fuente: Fotografía realizada por el estudiante.

Figura 2-9: Bandeja sensorial con el soporte para celular integrado.

El listado completo de los materiales utilizados en la elaboración de la bandeja sensorial se detallan en la tabla 2-2.

Tabla 2-2: Materiales utilizados en la elaboración de la bandeja sensorial.

Material	**Cantidad**
Terciado de pino reciclado	1
Piso laminado reciclado	2
Perno 1,5 pulgadas	1
Perno 2 pulgadas	2

Perno 2,5 pulgadas	1
Tuerca mariposa	4
Cola de carpintero	-
Sémola	500g

Fuente: Tabla realizada por el estudiante.

2.2 ANÁLISIS Y CREACIÓN DEL MODELO DE APRENDIZAJE

Un modelo es una representación de lo que un sistema de Aprendizaje Automático *aprendió* de los datos de entrenamiento. Estos datos de entrenamiento están contenidos en lo que se conoce como un conjunto de datos (Data set), que permite suministrar información al sistema. Para modelos de aprendizaje supervisado en el campo de la visión por computadora, es necesario crear un conjunto de datos compuesto por imágenes con y sin etiquetas (representativas de lo que el sistema deberá analizar en su funcionamiento) en las que se definen las clases u objetos que se desean detectar. Estas imágenes etiquetadas, son las que el sistema utiliza para entrenar y así poder realizar la predicción sobre las imágenes que no están etiquetadas (imágenes de prueba). Una vez entrenado el modelo y dependiendo de la plataforma en la que se elabore, se entregarán los datos de la precisión para la detección de las clases, el tiempo de respuesta y la facilidad con la que se identifica cada etiqueta, etc.

En la actualidad, existen modelos de aprendizaje ya entrenados para su uso en visión por computadora, capaces de detectar y clasificar objetos presentes en la vida cotidiana. Estos modelos, están diseñados para su uso en aplicaciones de CV en los que se requiere un conjunto de datos generalizado, recomendados en el caso de una aplicación en la que se requiera la detección de elementos comunes tales como automóviles, personas, rostros u otros objetos dentro de las clases definidas en los modelos.

Para efectos del proyecto, no existe un modelo ya entrenado que se adapte correctamente a las necesidades de éste, por lo que se necesitará crear un modelo de aprendizaje personalizado a partir de un modelo pre-entrenado. Para esto, en primera instancia se consideró el uso de la plataforma en la nube Google Cloud, que permite un desarrollo de modelos he implementación

de algoritmos de visión por computadora con un acercamiento más sencillo con su interfaz gráfica de AutoML Vision. Esta opción de trabajo se descartó debido a que este es un servicio de pago, si bien en un principio dan la posibilidad de obtener una prueba gratuita, era necesario tener en posesión una tarjeta de crédito para poder crear una cuenta y posteriormente, luego del periodo de prueba, se aplicarían cargos por horas de entrenamiento del modelo. Otro factor que insidió en el cambio de la metodología de trabajo, fue la necesidad de adquirir los conocimientos para el entrenamiento de modelos. Esto debido a que la plataforma está diseñada para simplificar en gran medida del trabajo de entrenamiento de los modelos de aprendizaje automático, evitando una aproximación más centrada en el código. Por este motivo se trabajará en un entorno de desarrollo de aprendizaje automático conocido como TensorFlow a través del intérprete de Python. La construcción, entrenamiento y despliegue del modelo se complementó con la documentación en línea llamada TensorFlow 2 Object Detection API tutorial.

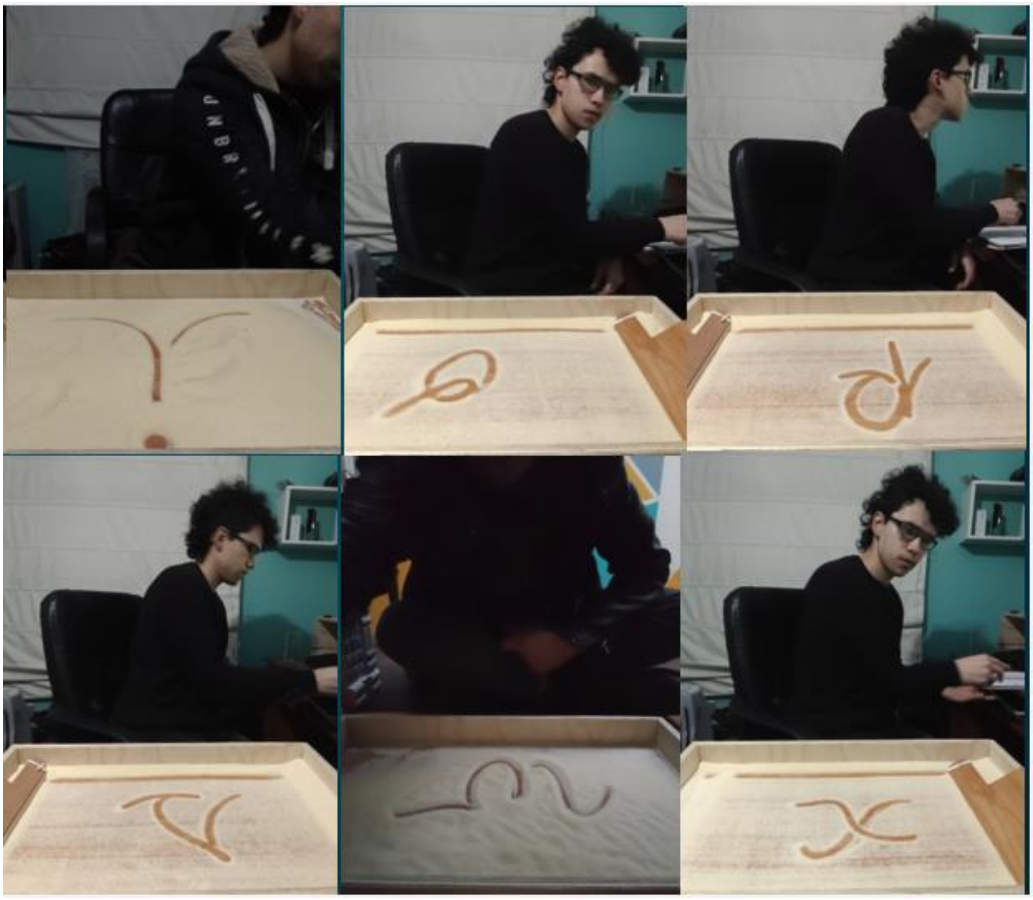
2.2.1 Captura y procesamiento del conjunto de datos

Con lo anteriormente descrito, se deben recopilar muestras de lo que el sistema debe detectar.

En un inicio, se recopilarán datos para un universo acotado, en el que sólo estarán las letras del abecedario (minúsculas) de la escritura manuscrita. Esto, debido a que la cantidad de imágenes para que el modelo sea efectivo depende de la cantidad de etiquetas que se utilizarán. Siguiendo esta regla, a mayor cantidad de etiquetas (objetos a detectar) se debe aumentar la cantidad de imágenes por cada una. Por consiguiente, se obtendrá un mínimo de 50 fotos por etiqueta (letra del abecedario), y aplicando este principio a un universo en el que se incluyan formas, palabras o números, el tiempo para la toma de muestras y la cantidad de datos que se requieren se extiende en gran medida.

Es importante que para cada letra se obtengan fotografías en distintas condiciones de luz, distintos tamaños de las letras, distintos fondos, etc. para que así el modelo sea lo suficientemente flexible y pueda funcionar en cualquier condición. Se obtuvieron aproximadamente 30 fotografías por cada letra, invirtiendo tiempo mayor al esperado. En total se obtuvo un conjunto de datos de 810 fotos.

En la figura 2-10 se observa una muestra de las fotografías que se obtuvieron para la creación del conjunto de datos.



Fuente: Fotografía realizada por el estudiante.

Figura 2-10: Muestra del conjunto de datos.

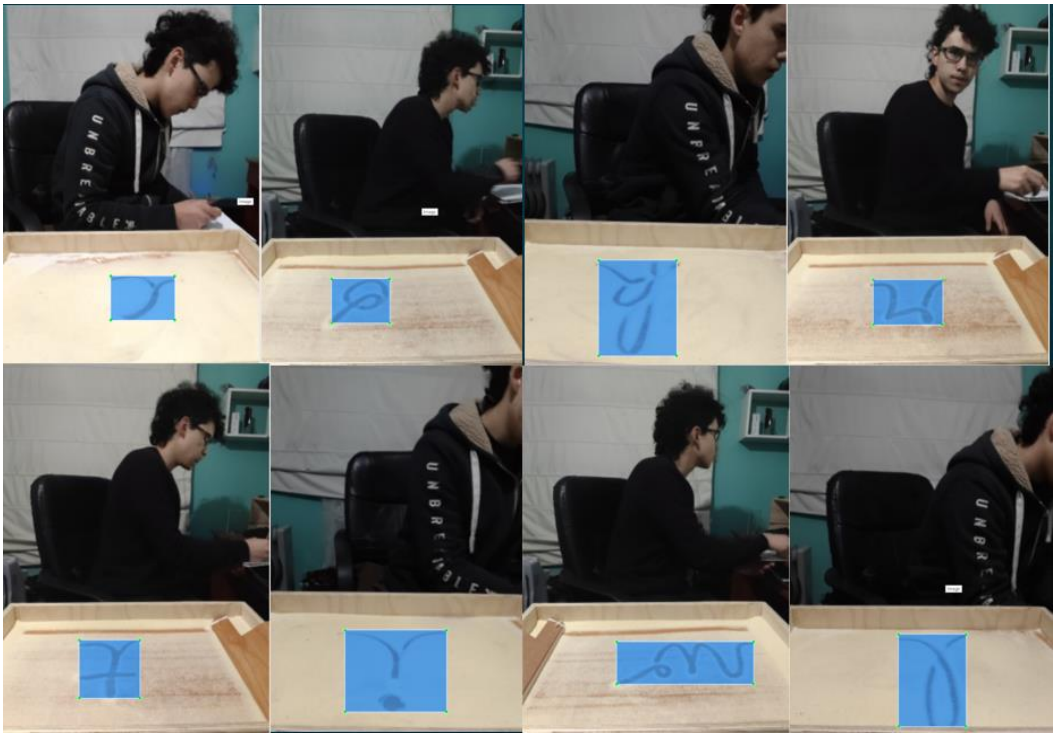
Una vez realizado y agrupado el conjunto de datos, se debe disminuir el tamaño de las imágenes para reducir el uso de recursos al momento de entrenar el modelo. Para esto se utilizará un script de código Python por lo que es necesario instalar el software intérprete de código Python (versión 3.7 para la compatibilidad con los otros softwares). El script itera entre las imágenes dentro de la carpeta en la que se ejecuta y las ajusta a 1/8 del tamaño original.

Ver anexo 1: Script para el reajuste del tamaño de las imágenes.

Luego, se procedió a etiquetar cada imagen con la herramienta de anotación gráfica de imágenes LabelImg (se utilizan cajas para enmarcar los objetos).

El proceso de etiquetado consiste en ubicar una caja/box (área de forma cuadrada o rectangular) sobre el objeto que se desea detectar, junto con el nombre de la etiqueta. En el caso práctico, se ubica una caja sobre cada letra realizada en la bandeja sensorial con su respectivo nombre. Es importante que la caja encierre solamente al elemento que se desea detectar puesto que, si se incluyen otros elementos dentro de ella, es probable que el modelo comience a entender esa parte como propia de la letra.

En la figura 2-11 se observa un grupo acotado de imágenes ya etiquetadas.



Fuente: Fotografía realizada por el estudiante.

Figura 2-11: Muestras del conjunto de datos etiquetado.

Al momento de etiquetar las fotografías, LabelImg crea un archivo con la extensión XML (de su traducción al español, Lenguaje de Marcado Extensible) para cada imagen, con el mismo nombre de esta, que indicará sus especificaciones, la ubicación de la caja y el nombre de la etiqueta.

2.2.2 Instalación de software para el entrenamiento del modelo

Para comenzar con el entrenamiento del modelo de aprendizaje automático es necesario instalar una serie de programas y herramientas de compilación para configurar el entorno de desarrollo en Windows. En la web oficial de Tensorflow se encuentra la documentación que muestra la compatibilidad entre las distintas versiones de los programas. Esto es de suma importancia debido a que, si se instalan las versiones incorrectas, el entrenamiento no se podrá realizar.

TensorFlow puede trabajar tanto en la CPU del equipo como en la GPU (GPU de NVIDIA compatible) para agilizar los tiempos de entrenamiento. En la siguiente tabla se muestra un fragmento de la tabla de compatibilidad para el caso de utilizar la GPU del sistema.

Versión	Versión de Python	Compilador	Herramientas de compilación	cuDNN	CUDA
tensorflow_gpu-2.5.0	3.6 a 3.9	MSVC 2019	Bazel 3.7.2	8.1	11.2
tensorflow_gpu-2.4.0	3.6 a 3.8	MSVC 2019	Bazel 3.1.0	8.0	11.0
tensorflow_gpu-2.3.0	3.5 a 3.8	MSVC 2019	Bazel 3.1.0	7.6	10.1
tensorflow_gpu-2.2.0	3.5 a 3.8	MSVC 2019	Bazel 2.0.0	7.6	10.1
tensorflow_gpu-2.1.0	3.5 a 3.7	MSVC 2019	Bazel 0.27.1-0.29.1	7.6	10.1
tensorflow_gpu-2.0.0	3.5 a 3.7	MSVC 2017	Bazel 0.26.1	7.4	10

Fuente: https://www.tensorflow.org/install/source_windows#gpu

Figura 2-12: Bandeja sensorial con el soporte para celular integrado.

Para la compatibilidad con la última versión de Tensorflow (v2.5.0) se instalaron las siguientes versiones de los programas:

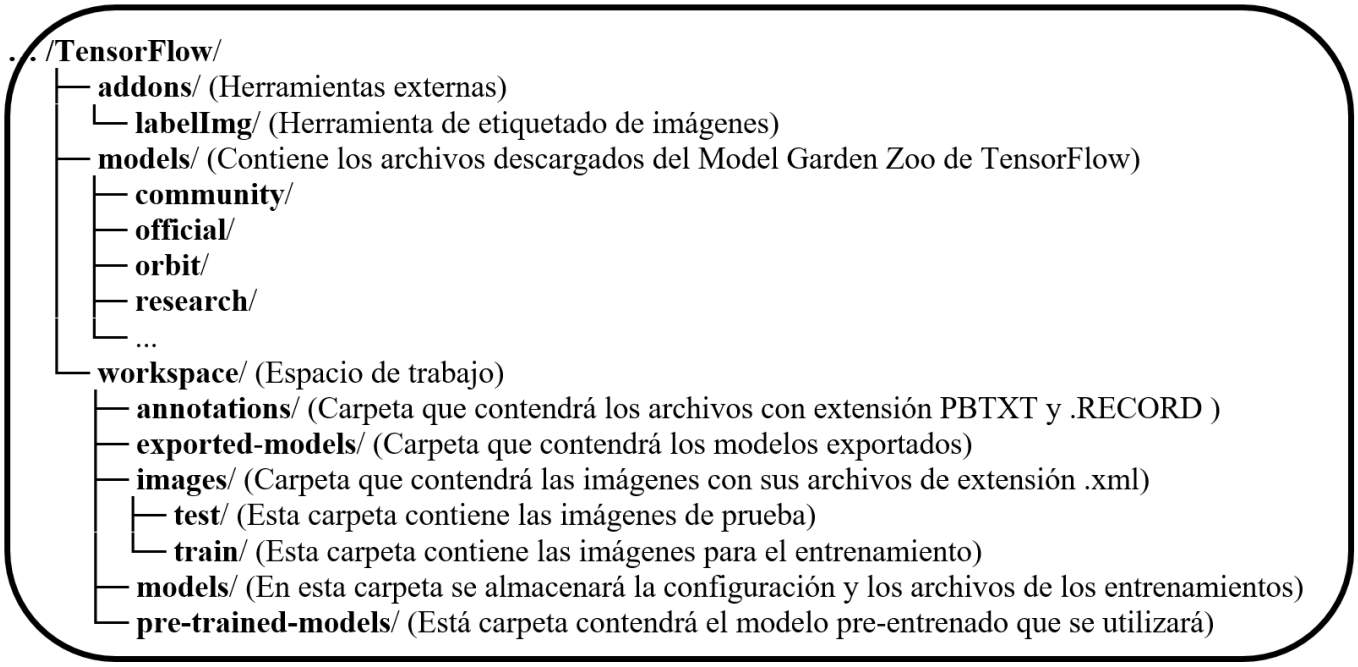
Tabla 2-3: Software necesario para el entrenamiento del modelo

Nombre del Software	**Versión**	**Descripción**
Python	3.7	Software que permite escribir e interpretar código de Python.
Microsoft Visual C++ compiler	MSCV 2019	El compilador de Windows de código C++ es necesario debido a que TensorFlow, en su raíz, está escrito en este lenguaje.
NVIDIA CUDA Toolkit y CuDnn	Cuda toolkit v11.2 y CuDnn 8.1.0	Estos programas permiten que el entrenamiento del modelo sea realizado por la GPU del sistema.
Protobuf	-	Protocolos de comunicación
TensorFlow	2.5.0	Es el entorno de trabajo para el aprendizaje automático (machine learning) en el que se desarrollará el entrenamiento del modelo.
TensorFlow Object Detection API	-	Es la API de TensoFlow que facilita la construcción, el entrenamiento y el despliegue de modelos para detección de objetos.

Fuente: Tabla realizada por el estudiante.

2.2.3 Preparación del entorno de trabajo

Para trabajar en forma ordenada se creó un directorio en el que se encontrará todo lo necesario para entrenar el modelo. En la figura 2-13 se muestra la configuración del directorio.



Fuente: Material realizado por el estudiante basado en el TensorFlow Object Detection API tutorial.

Figura 2-13: Disposición de los directorios.

Se optó por nombrar las carpetas en inglés debido a que la documentación se encuentra en ese idioma y así se evitan errores de escritura.

Una vez se tiene configurado el directorio, se dividió el conjunto de datos que se obtuvo anteriormente en un 90% para imágenes de entrenamiento y el 10% restante para las de prueba. Cada conjunto se debe guardar en su respectiva carpeta (…/train/ y …/test/ respectivamente).

El siguiente paso fue crear un archivo de texto con el formato de extensión.pbtxt (ej. label_map.pbtxt) que contendrá los nombres de todas las etiquetas del conjunto de datos y se le asignará un valor entero. El formato para este archivo será el siguiente:

Ver en Anexo 2: Formato del archivo label_map.pbtxt.

Luego de haber creado el archivo, se almacanó dentro de la carpeta annotations. En esta misma carpeta se guardarán dos archivos más con el formato TensorFlow Record (.record) que compilará toda la información del conjunto de datos en un archivo, tanto de las imágenes de entrenamiento como las de prueba. Para esto se utilizó un script que realiza la conversión de los archivos XML a TFRecord.

Ver en anexo 3: Script para la conversión de archivos XML a TFRecord.

En la figura 2-14 se especifican los comandos utilizados para ejecutar el programa para ambos archivos.

En el cmd de Windows:

 # Crear el archivo Data de entrenamiento:
python generate_tfrecord.py -x [RUTA_DEL_DIRECTORIO_DE_IMAGENES]/train -l [RUTA_DEL_DIRECTORIO_ANNOTATIONS]/label_map.pbtxt -o [RUTA_DEL_DIRECTORIO_ANNOTATIONS]/train.record

 # Crear el archivo Data de prueba:
python generate_tfrecord.py -x [RUTA_DEL_DIRECTORIO_DE_IMAGENES]/test –l [RUTA_DEL_DIRECTORIO_ANNOTATIONS]/label_map.pbtxt -o [RUTA_DEL_DIRECTORIO_ANNOTATIONS]/test.record

Fuente: Material realizado por el estudiante basado en el TensorFlow Object Detection API tutorial.

Figura 2-14: Comandos para la ejecución del script de conversión de formato

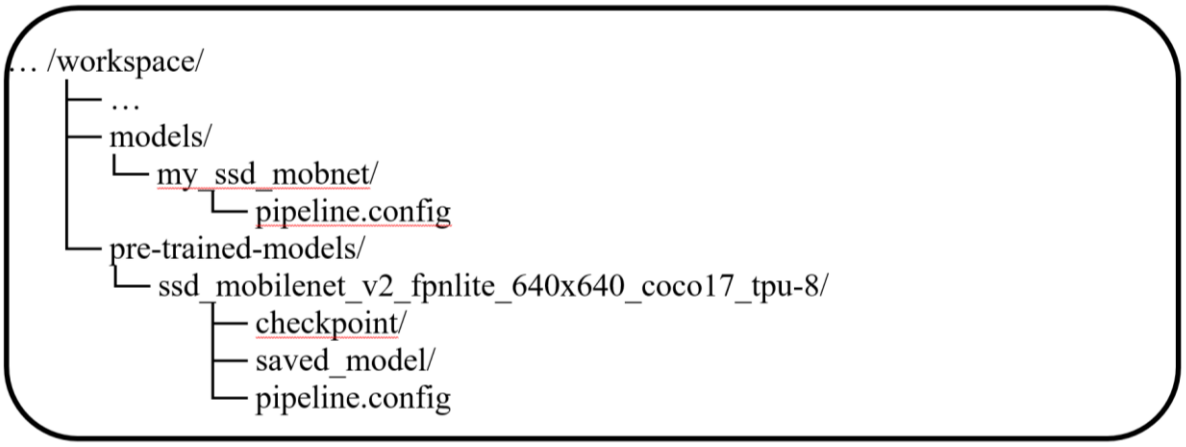
Esto creará dos archivos nuevos dentro de la carpeta annotations que se denominarán train.record y test.record respectivamente.

2.2.4 Configuración del trabajo de entrenamiento

Como se mencionó, se trabajará reutilizando un modelo pre-entrenado ya existente dentro de los modelos que proporciona TensorFlow y se adaptará para su entrenamiento con el conjunto de datos personalizado. Esto disminuye en gran medida la labor de creación del modelo, debido a que ya viene con todos sus parámetros configurados.

Cada modelo del TensorFlow Model Zoo está entrenado con el conjunto de datos llamado COCO 2017 Dataset y tiene un rendimiento que será importante analizar para identificar qué modelo se adapta mejor al trabajo. Con esto en mente, se eligió el modelo SSD mobilenet v2 fpnlite 640x640 que está optimizado para ser implementado en dispositivos móviles y procesa las imágenes en un tamaño de 640x640 pixeles. Dentro de los parámetros del rendimiento del modelo se tiene un tiempo de respuesta de 39ms, con una precisión de un 28.2% evaluado en el conjunto de datos COCO.

El modelo descargado debe ser ubicado en la carpeta pre-trained-models y posteriormente, dentro de la carpeta models, se creará una carpeta con el nombre que se le quiera dar al entrenamiento (ej. my_ssd_mobnet) y se copiará el archivo pipeline.config ubicado en la carpeta antes descargada. En la figura 2-14 se muestra como deberían quedar los directorios.



Fuente: Material realizado por el estudiante basado en el TensorFlow Object Detection API tutorial.

Figura 2-15: Disposición de los directorios con los archivos descargados.

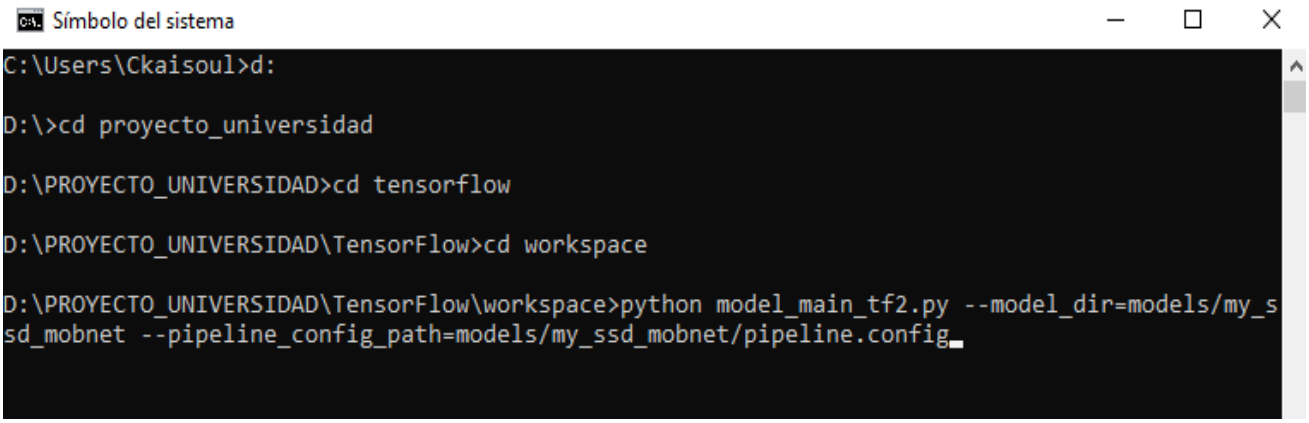
Posterior a esto, se debe configurar el archivo pipeline.config dentro de my_ssd_mobnet para adecuarlos a los datos de trabajo. En el anexo 4, se muestra el contenido del archivo y las líneas de código que se deben modificar están marcadas en amarillo.

Una vez realizado estos cambios al archivo se deben guardar para proceder con el entrenamiento del modelo.

2.2.5 Entrenamiento del modelo

El entrenamiento es la etapa del aprendizaje automático en la que el modelo se optimiza gradualmente para disminuir el error o, en términos más simples, la etapa en la que el modelo *aprende* el conjunto de datos. Esto se realiza con el objetivo que *aprenda* lo suficiente sobre el conjunto de datos de entrenamiento para poder realizar predicciones sobre nuevos datos. Un problema que se puede presentar durante el proceso, es que el modelo *aprenda* demasiado sobre el conjunto de datos y que las predicciones solo funcionen para los datos que ha visto, evitando la generalización del modelo. A este problema se le conoce como sobreajuste (o del inglés, overfitting).

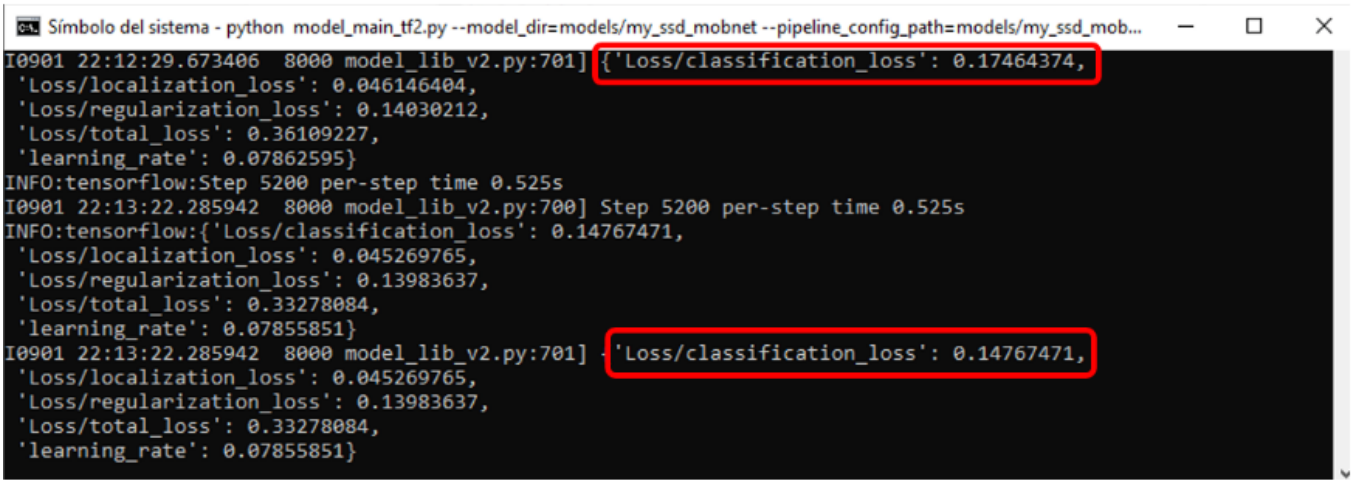
Para comenzar con el entrenamiento se debe copiar el script que se encuentra en TensorFlow/models/research/object_detection/model_main_tf2.py dentro de la carpeta workspace, para así ejecutarlo desde el Command Prompt Windows. Ya en el CMD, se debe acceder al directorio workspace y ejecutar el comando que se muestra en la figura 2.16.



Fuente: Captura de pantalla realizada por el estudiante.

Figura 2-16: Ejecución del comando de entrenamiento.

Con esto ya se debería comenzar a entrenar el modelo y en el momento en que se muestren los valores de la función de pérdida significará que el modelo se está entrenando adecuadamente. En la figura 2.17 se aprecia el valor de la pérdida, que corresponde a la función que determina cuánto se desvían los valores de las predicciones en relación a los valores deseados y por consiguiente el sistema se optimiza para disminuir este valor.



Fuente: Captura de pantalla realizada por el estudiante.

Figura 2-17: Muestra del valor de la función de pérdida.

Si el valor de la función de pérdida es pequeño, es mayor el grado de precisión que ha tomado el modelo para predecir los objetos que están dentro del conjunto de datos, pero es importante evitar valores demasiado pequeños para no generar un problema de sobreajuste del modelo.

Durante el entrenamiento se generan varios archivos que son comunes entre los modelos creados por TensorFlow. Estos son:

- model-ckpt.meta: Este archivo contiene el gráfico completo del modelo.
- model-ckpt.data-0000-of-00001: Este archivo contiene todos los valores de las variables (pesos, biases, placeholders,gradients, hyper-parameters etc).
- model-ckpt.index: Este archivo contiene la MetaData.
- checkpoint: Los puntos clave o checkpoints permiten retomar la tarea al momento de terminar una instancia de entrenamiento o en caso de que ocurra algún error inesperado y así, no se pierda el progreso.

2.2.6 Exportar el modelo a TensorFlow Lite

Para poder exportar el modelo a formato TensorFlow Lite es necesario obtener un archivo que se le conoce como un gráfico de inferencia congelado (o del inglés, frozen inference graph), que

básicamente consiste en congelar el trabajo de entrenamiento para poder realizar inferencias. Este formato de archivo no puede volver a ser entrenado, debido a que solo contiene los parámetros importantes como los datos del gráfico del modelo y de los pesos de los parámetros de éste. Para crear este archivo se utilizaron las siguientes líneas de comando.

Ver Anexo 5: Congelar un gráfico de inferencias.

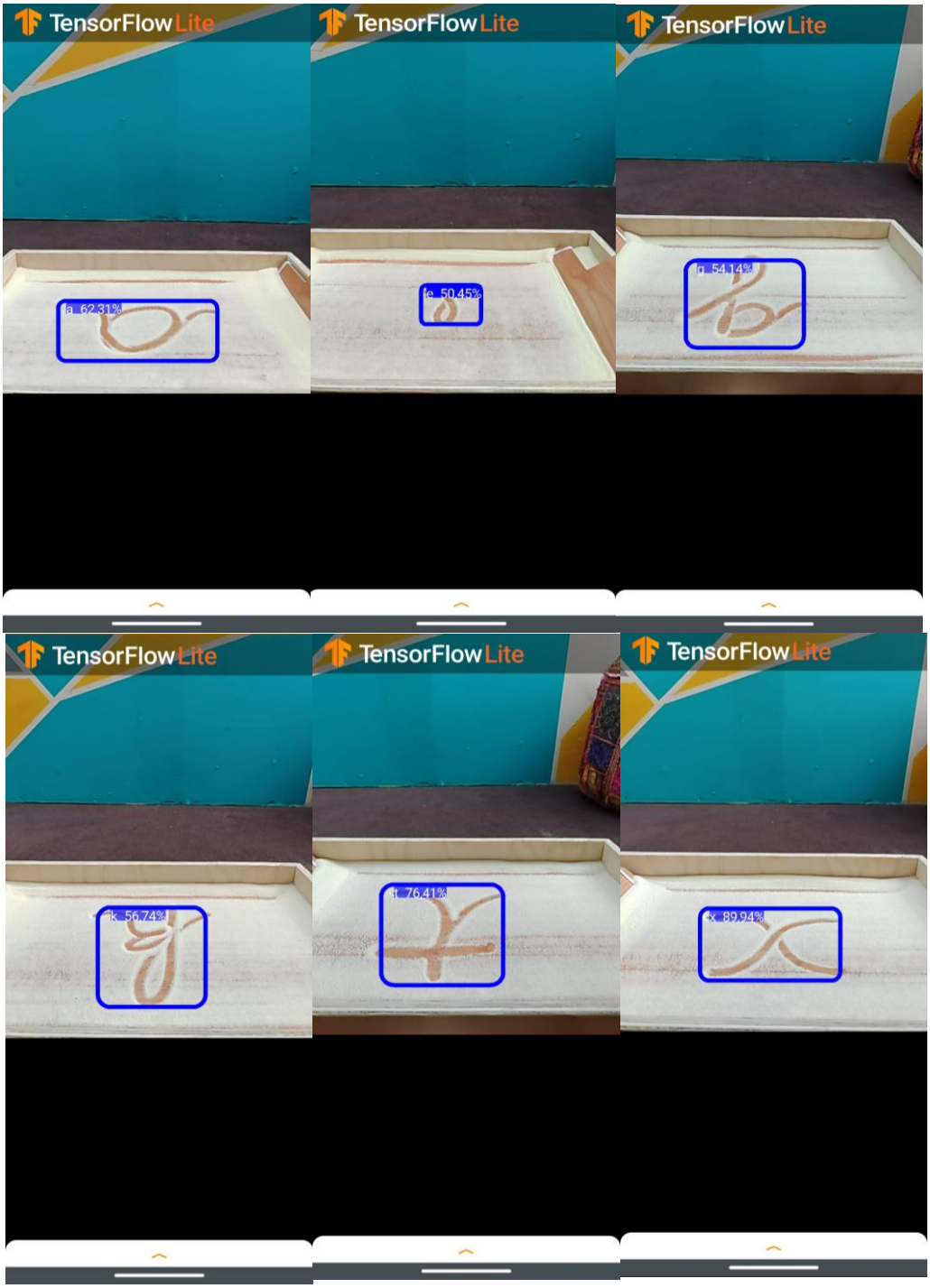
Una vez se tiene este archivo llamado saved_model guardado en la carpeta exported-models, es posible convertirlo a formato TensorFlow Lite con la ayuda de la librería TensorFlow Lite Converter. En el anexo 6: Convertir modelo a formato TensorFlow Lite se encuentra el código del script utilizado para la conversión.

Por último, el modelo necesita estar empaquetado con MetaData de TensorFlow Lite para integrarlo en forma sencilla en aplicaciones móviles usando la librería TFLite Task Library. Esta MetaData permite que el código de inferencia ejecute en forma correcta el pre y post procesado requerido por el modelo. El código utilizado para la creación de la MetaData se encuentra en el Anexo 7: Agregar MetaData al modelo.

2.2.7 Implementar el modelo en una aplicación de Android

Para analizar el funcionamiento del modelo, se utilizará una aplicación de cámara de prueba proporcionada por TensorFlow Lite, que permite realizar detecciones en tiempo real. Se modificó el código de la aplicación para poder implementar el modelo personalizado.

La aplicación cuenta con una interfaz sencilla en la que se muestra una visualización de lo que está capturando la cámara del dispositivo, además de un conjunto de valores como el tiempo de respuesta del sistema, el tamaño de la captura de video, los hilos del procesador que se pueden utilizar. Al momento de realizar una predicción, la aplicación dibuja una caja en el contorno del objeto que se detectó, indicando la etiqueta de la letra que se detectó y su grado de confianza. En la siguiente figura 2-18 se muestran capturas de pantalla realizadas en el celular que permiten observar el funcionamiento de la aplicación.

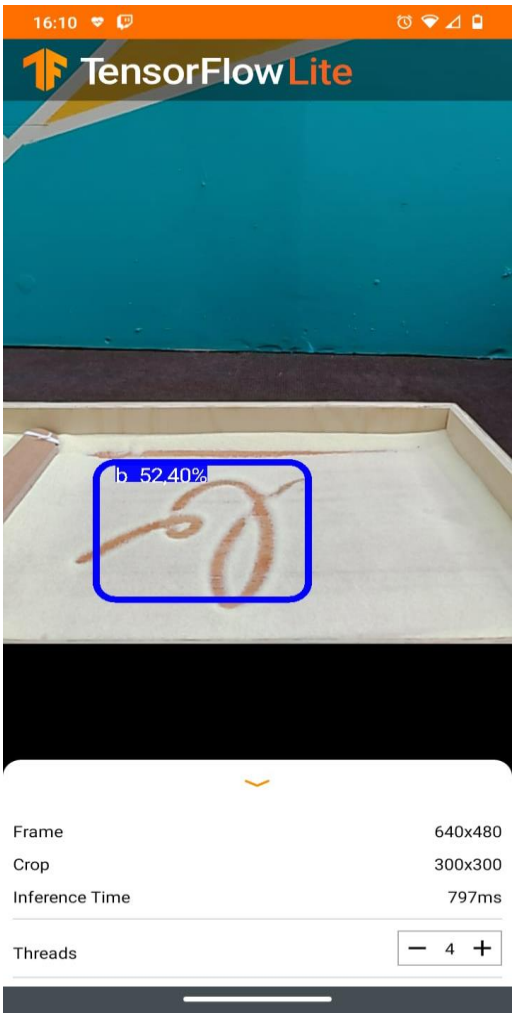


Fuente: Capturas de pantalla realizadas por el estudiante.

Figura 2-18: Funcionamiento de la aplicación de prueba.

Si bien en las capturas de pantalla las detecciones se ven estáticas, el sistema es capaz de hacer seguimiento a los objetos que está detectando en tiempo real.

Al deslizar el botón blanco de la parte inferior de la interfaz de la aplicación hacia arriba, se observan las métricas antes descritas.



Fuente: Captura de pantalla realizada por el estudiante.

Figura 2-19: Interfaz de la aplicación de prueba.

CAPÍTULO 3: ANÁLISIS DE LA IMPLEMENTACIÓN Y EVALUACIÓN DE COSTOS DEL PROYECTO.

En este capítulo se abordará el análisis de la implementación del proyecto y de su desempeño en un entorno real. Se realizarán pruebas que permitan conocer, a grandes rasgos, la precisión del modelo creado y la flexibilidad de éste, encontrando sus fortalezas y debilidades. Además, se realizará una evaluación de costos para determinar la rentabilidad del proyecto y el costo de la HH.

3.1 ANÁLISIS DE LA IMPLEMENTACIÓN DEL PROYECTO.

Para realizar un análisis completo del proyecto, esta sección se dividirá en 5 partes que contemplarán tanto el desarrollo de la bandeja sensorial como la obtención y etiquetado del conjunto de datos, programación y entrenamiento del modelo de aprendizaje automático.

3.1.1 Desarrollo de la bandeja sensorial.

El modelado y diseño de la bandeja sensorial se realizó en dos etapas de elaboración. El primer modelado del diseño se creó con la intención de asentar las bases de lo que sería el producto final y presentar la cohesión que existiría entre la bandeja sensorial y el soporte para el celular. Analizando este prototipo, se consideró la necesidad de tener un sistema con la suficiente flexibilidad para que se desempeñara en la forma correcta. Por esto, el primer prototipo fue modificado, con la intención de atender a estas necesidades y plasmar lo que sería el producto final. Este proceso fue uno en el que el análisis y la proyección del proyecto en un entorno real permitieron la mejora constante del prototipo, otorgando así un trabajo más flexible y a la vez estructurado.

Ya en el proceso de elaboración de la bandeja sensorial no se presentaron mayores inconvenientes, debido a que el estudiante poseía experiencia en el manejo de las herramientas eléctricas utilizadas.

3.1.2 Captura de las imágenes para el conjunto de datos

La creación del conjunto de datos que se utilizaría para el entrenamiento del modelo fue un proceso que demandó mucho tiempo y esfuerzo, debido a la cantidad de imágenes que se debían obtener por cada letra para que el modelo fuera competente, era muy alta. En la práctica se obtuvieron alrededor de 30 imágenes por cada letra, lo que constituye un conjunto de datos de un total de 810 fotografías. Esta cantidad es menor de la que se tenía estipulada pero debido a que el tiempo de la captura de imágenes se extendió más de lo considerado, se optó por disminuir la cantidad de fotos por letra. Teniendo esto en consideración, una importante mejora para el rendimiento del modelo sería aumentar la cantidad imágenes por etiqueta (letra).

3.1.3 Etiquetado del conjunto de datos

El etiquetado de las imágenes también fue un proceso que demandó una gran cantidad de tiempo y esfuerzo, debido a que cada imagen debió ser etiquetada manualmente y al inicio algunas de las fotos quedaron mal etiquetadas, esto implicó volver a etiquetar cada imagen que estaba errónea.

Del mismo modo, otro error que se cometió en el proceso de etiquetado fue no disminuir el tamaño de las imágenes. Esto generó un problema al momento de entrenar el modelo, puesto que el computador simplemente no podía procesar los archivos con el tamaño original de la fotografía, derivando en que éste se congelara e indicara un error en el entrenamiento debido a la falta de hardware para procesar tal cantidad de información. Las dimensiones originales de las imágenes eran de 2448x3264 píxeles, con un tamaño aproximado de 1.5MB. Considerando el total de las imágenes que componen el conjunto de datos y su tamaño total, era un valor demasiado grande para poder ser procesadas por el computador. Por esto fue que se pre-procesaron las imágenes para disminuir su tamaño a 1/8 del original, quedando con unas dimensiones de 306x408 píxeles y un tamaño aproximado de 35KB, un valor significativamente más pequeño que el original.

Una vez se completó el proceso anterior, se volvió a etiquetar todo el conjunto de datos con las consideraciones antes mencionadas.

3.1.4 Instalación del software necesario para el entrenamiento del modelo

Otra etapa en el desarrollo del proyecto que demandó más tiempo del esperado fue la instalación del software. Para esto se siguió el tutorial de instalación oficial encontrado en el TensorFlow Object Detection API tutorial. En el primer intento de realizar la instalación, surgieron errores de compatibilidad y de algunas librerías de Python que no se encontraban o que no eran posibles de instalar. Por consiguiente, se optó por desinstalar todo el software he instalarlo nuevamente, teniendo en consideración los elementos de compatibilidad de las versiones y librerías que anteriormente se encontraron problemas para instalar.

3.1.5 Entrenamiento del modelo

El primer intento de entrenamiento se ejecutó en forma errónea, debido a que se comenzó con el tamaño original de las imágenes y esto implicó que durante el entrenamiento el sistema se quedara sin memoria para poder procesar tal cantidad de datos. Esto generó que el tiempo de entrenamiento fuera demasiado largo y que el computador se congelara en variadas oportunidades, concluyendo en un entrenamiento erróneo y un modelo que no funcionaba.

Una vez que se solucionaron los problemas anteriormente descritos, el proceso de entrenamiento del modelo fue relativamente sencillo, puesto que, al ejecutar el comando para iniciar esta etapa es importante darle tiempo al sistema para que pueda realizar los pasos de entrenamiento de forma correcta. La única condición que puede afectar el funcionamiento del modelo para la predicción de objetos, es dejar que la función de pérdida disminuya a un valor bastante pequeño. Esto es lo que anteriormente se definió como sobreajuste.

3.2 EVALUACIÓN DEL MODELO DE APRENDIZAJE AUTOMÁTICO

Para evaluar el desempeño del modelo de aprendizaje automático y ver qué tan preciso sería en un entorno real, se realizaron pruebas en las que se digitaban todas las letras del abecedario en la bandeja sensorial y se utilizaba la aplicación. Con estas pruebas se pudo obtener una generalización del rendimiento del modelo, dejando evidencias de que hay letras con las cuales tuvo más y menos inconvenientes al momento de reconocerlas.

En la tabla 3-1 se encuentran los datos obtenidos de las pruebas. El índice de confianza es un porcentaje que calcula el sistema indicando la probabilidad de acierto de las predicciones y la dificultad para la detección es una variable que considera el tiempo de demora de la detección y predicciones erróneas.

Tabla 3-1: Datos obtenidos de las pruebas realizadas.

Letra	Índice de confianza	Dificultad para la detección
A	55-65%	Mediana
B	50-60%	Mediana
C	-	Nunca se detectó
D	60-70%	Mediana
E	50-60%	Baja
F	50-60%	Alta
G	50-60%	Alta
H	-	Nunca se detectó
I	55-65%	Baja

J	65-75%	Baja
K	55-65%	Baja
L	50-65%	Baja
M	50-65%	Baja
N	-	Nunca se detectó
Ñ	55-65%	Baja
O	-	Nunca se detectó
P	60-70%	Baja
Q	60-70%	Baja
R	70-80%	Baja
S	55-65%	Mediana
T	70-80%	Baja
U	70-85%	Baja
V	50-60%	Baja
W	55-65%	Baja
X	80-90%	Baja
Y	55-65%	Baja
Z	50-60%	Baja

Fuente: Tabla realizada por el estudiante con los datos obtenidos de las pruebas.

Analizando los resultados de la prueba, se puede concluir que el modelo en su mayor parte se desempeña de forma correcta, pero todavía existe margen de mejora que puede aumentar su precisión. Una forma de mejorar las predicciones del modelo es aumentando la cantidad de fotos por cada letra en el conjunto de datos. Esto permite que exista una mayor cantidad de imágenes por clase (o etiqueta), influyendo positivamente en la cantidad de información que el sistema puede obtener para hace predicciones más precisas.

3.3 SUMARIO DEL DESARROLLO DEL PROYECTO

Considerando los tiempos acotados para el desarrollo del proyecto y la dificultad de éste, se optó por optimizar el trabajo para cumplir con los ejes más importantes. Elementos que influyeron en esta toma de decisiones fueron, en su mayoría, tiempos de trabajos extendidos debido a problemas con el desarrollo del software del proyecto. Teniendo esto en consideración, se trabajó para el cumplimiento de la mayor cantidad de objetivos específicos, los cuales se detallan a continuación:

- Elaboración de la bandeja sensorial junto con el soporte dedicado para el celular.
  Este fue un objetivo específico que se completó en forma exitosa y sin mayores inconvenientes.
- Desarrollar una aplicación en Android que permita activar la cámara del celular para la obtención de datos.
  Este objetivo se logró, ya que se adaptó una aplicación desarrollada por TensorFlow Lite que contenía el código básico que permitía probar el modelo personalizado con la cámara del dispositivo móvil.
- Implementar las librerías y el modelo de aprendizaje automático en la aplicación, para la detección de objetos en tiempo real.
  Este objetivo también se realizó, puesto que se modificó la aplicación para poder implementar el modelo de aprendizaje automático personalizado.

- Desarrollar una interfaz para mostrar la imagen del objetivo a realizar en la bandeja sensorial.

  Este objetivo fue el único que no se completó, debido a que el tiempo de investigación para la creación de aplicaciones en sistema operativo Android no fue el suficiente como para cumplir a cabalidad este objetivo.

Analizando los ejes temáticos del proyecto, se evidencia la necesidad de obtener conocimiento en áreas de trabajo que no fueron parte de la formación del estudiante, implicando un gran esfuerzo en el estudio de nuevos contenidos y la organización de los tiempos de trabajo destinados al desarrollo del proyecto. Uno de los ejes que más tiempo de desarrollo demandó fue el entrenamiento del modelo, puesto que en un principio se consideró el uso de software y trabajo en la nube de Google, que disminuiría en gran medida el proceso de entrenamiento del modelo. Esta opción se descartó por los motivos antes mencionados, destacando el hecho que para acceder a estos servicios era necesario contar con una tarjeta de crédito y posteriormente se realizaban cargos monetarios a la cuenta por hora de entrenamiento del modelo, y que para el estudiante era necesario obtener los conocimientos para el desarrollo de modelos de aprendizaje automático, esto debido que esta plataforma simplificaba bastante el proceso. Este factor fue el que más afectó el desarrollo del proyecto, debido a que se tuvo que invertir mayor tiempo en la investigación de este tópico y en la implementación surgieron errores que afectaron directamente el desarrollo de los objetivos.

3.4 EVALUACIÓN DE COSTOS DEL PROYECTO

El enfoque del proyecto siempre fue velar por la disminución de los costos, tanto en el impacto monetario, como medioambiental y social. Por este motivo, se tomaron decisiones como la selección de la mejor propuesta de desarrollo, que contempló los aspectos del costo monetario y el impacto social de las alternativas. En cuanto al costo monetario, era evidente que entre ambas propuestas destacaba aquella que permitía realizar el proyecto con un costo monetario prácticamente nulo, debido a que todo el hardware necesario para llevarla a cabo se encontraba

a disposición del estudiante. Del mismo modo, todo el software utilizado para el desarrollo del proyecto es gratuito y de libre acceso para todos/as aquellos/as que lo requieran.

En el ámbito social se tuvieron las siguientes consideraciones, realizar un proyecto que sea replicable, accesible y de muy bajo o nulo costo. Esto permite que el producto derivado de este proyecto pueda ser distribuido en forma sencilla, a gran escala y en forma gratuita, para ser utilizado por quien lo necesite.

En el desarrollo de la bandeja sensorial y con el fin de reducir el impacto medio ambiental, se optó por utilizar materiales reciclados que se encontraban en el hogar del estudiante.

La tabla 3-2 muestra los materiales utilizados para la creación de la bandeja sensorial con sus costos.

Tabla 3-2: Materiales utilizados en la elaboración de la bandeja sensorial.

Material	**Cantidad**	**Precio c/u ($)**	**Precio total ($)**
Terciado de pino reciclado	1	-	-
Piso laminado reciclado	2	-	-
Perno ¼ x 1,5 pulgadas	1	150	150
Perno ¼ 2 pulgadas	2	150	300
Perno ¼ 2,5 pulgadas	1	250	250
Tuerca mariposa ¼	4	160	800
Cola de carpintero	-	-	-
Sémola	500g		1.000
Total	-	-	2.500

Fuente: Tabla de materiales realizada por el estudiante.

Para los costos relacionados con la H.H. se estimó un total de 60 horas de dedicación al trabajo, las cuales incluyen tiempo para la investigación, diseño y prototipo, desarrollo de la bandeja y del

entrenamiento del modelo junto con su implementación en la aplicación para móviles. El valor aproximado del sueldo de un Técnico Universitario en Electrónica al primer año de egreso es de $665.716, que equivalen a 18,71UF mensuales (dato obtenido de la página web https://queestudiarenchile.com/, basada en datos entregados por el Servicio de Información de Educación Superior (SIES) vigentes al año 2021). Considerando que el mes cuenta con 4 semanas con una jornada laboral de 45 horas, el precio de la hora de trabajo de un técnico Universitario en Electrónica es de $3.700 aproximadamente, lo que equivale a 0.1UF y se muestra en Fig.3-3

Tabla 3-3: Cálculo del costo HH.

Pago mensual	Pago por hora	Costo HH
18,71 UF	0,104 UF	6,24 UF (60 horas)

Fuente: Tabla realizada por el estudiante con los datos obtenidos en https://queestudiarenchile.com/.

El costo total del proyecto incluye la suma de los costos materiales y los costos de H.H. En la tabla 3-4 se aprecia este valor.

Tabla 3-4: Costo total del proyecto.

Costos materiales	Costo HH	Costo total
0,07 UF	6,24 UF(60 horas)	6,31 UF

Fuente: Tabla realizada por el estudiante con los datos obtenidos en https://queestudiarenchile.com/.

Analizando los costos finales, se evidencian los enfoques del proyecto, que siempre buscaron la disminución del costo de desarrollo e implementación. Esto con el objetivo de realizar un producto que fuera replicable, accesible y gratuito.

CONCLUSIÓN

El proyecto se enfocó en integrar las tecnologías actuales en una herramienta utilizada en la educación, entregando una mirada de como la IA se puede utilizar para dar solución a variadas problemáticas o labores que mejorarían el trabajo educativo, tanto para los profesores como para los estudiantes. Junto con esto, se elaboró un proyecto innovador que permitió al estudiante implementar varias de las habilidades desarrolladas en su formación como profesional, destacando la resolución de problemas, la visión de futuro y adaptabilidad en un mundo que constantemente está cambiando, y reforzar valores personales como la paciencia y la perseverancia, la responsabilidad y alimentando la curiosidad, por eso que aún no se conoce y permite estar a la vanguardia de las tecnologías.

Debido a los recientes avances en el desarrollo de la Inteligencia artificial, y más específico, en Machine Learning y Deep Learning, que permitieron la mejora de estas ciencias, es que son tecnologías relativamente nuevas y que presentan un importante desafío al momento de adquirir conocimientos. Pero tienen un potencial que cada vez se acrecienta y permite generar soluciones en áreas que anteriormente eran indispensables, mejorando la producción o el desempeño en éstas.

Este proyecto, presentó un gran desafío para el estudiante, puesto que, al tener una mirada innovadora en el sector, se contó con muy poca orientación e información relevante al comienzo de su desarrollo. Lo anterior permitió que el desarrollador estuviera siempre atento a las necesidades del proyecto, generando un entorno de exigencia constante que influyó en un desarrollo más ordenado.

Asimismo, se deja la invitación para que nuevos desarrolladores impulsen sus proyectos con este enfoque innovador e integrar conceptos que posiblemente no se encuentren en su dominio de la materia. Es una tarea que demanda tiempo y esfuerzo, pero permite ampliar las habilidades personales y sus conocimientos para mejorar como futuros profesionales.

BIBLIOGRAFÍA

[1] Machine Learning Glosary. [En linea].

Google Developer. (s.f.). Recuperado el 15 de Agosto de 2021, de https://developers.google.com/machine-learning/glossary/#m

[2] Entrenamiento personalizado: tutorial. [En línea]

Tensor Flow. (s.f.). Recuperado el 20 de Agosto de 2021, de https://www.tensorflow.org/tutorials/customization/custom_training_walkthrough

[3] TensorFlow 2 Object Detection API tutorial [En línea]

Tensor Flow. (s.f.). Recuperado el 29 de Agosto de 2021, de https://tensorflow-object-detection-api-tutorial.readthedocs.io/en/latest/index.html

[4] ROUHIAINEN, L. (2018). INTELIGENCIA ARTIFICIAL: 101 cosas que debes saber hoy sobre nuestro futuro. Barcelona, España: Alienta. Obtenido de https://static0planetadelibroscom.cdnstatics.com/libros_contenido_extra/40/39308_Inteligencia_artificial.pdf

ANEXOS

Anexo 1: Script para el reajuste del tamaño de las imágenes:

```
## Bulk image resizer
# This script simply resizes all the images in a folder to one-eigth their
# original size. It's useful for shrinking large cell phone pictures down
# to a size that's more manageable for model training.
# Usage: place this script in a folder of images you want to shrink,
# and then run it.

import numpy as np
import cv2
import os

dir_path = os.getcwd()
print(dir_path)

for filename in os.listdir(dir_path):
   if filename.endswith(".jpg"):
      image = cv2.imread(filename)
      resized = cv2.resize(image,None,fx=0.25, fy=0.25, interpolation=cv2.INTER_AREA)
      cv2.imwrite(filename,resized)
```

Anexo 2: Formato del archivo label_map.pbtxt:

```
item {
 id: 1
 name: 'a'
```

```
}

item {
  id: 2
  name: 'b'
}

item {
  id: 3
  name: 'c'
}

item {
  id: 4
  name: 'd'
}

item {
  id: 5
  name: 'e'
}

item {
  id: 6
  name: 'f'
}

item {
  id: 7
  name: 'g'
```

```
}

item {
  id: 8
  name: 'h'
}

item {
  id: 9
  name: 'i'
}

item {
  id: 10
  name: 'j'
}

item {
  id: 11
  name: 'k'
}

item {
  id: 12
  name: 'l'
}

item {
  id: 13
  name: 'm'
```

```
}

item {
  id: 14
  name: 'n'
}

item {
  id: 15
  name: 'ñ'
}

item {
  id: 16
  name: 'o'
}

item {
  id: 17
  name: 'p'
}

item {
  id: 18
  name: 'q'
}

item {
  id: 19
  name: 'r'
```

```
}

item {
  id: 20
  name: 's'
}

item {
  id: 21
  name: 't'
}

item {
  id: 22
  name: 'u'
}

item {
  id: 23
  name: 'v'
}

item {
  id: 24
  name: 'w'
}

item {
  id: 25
  name: 'x'
```

```
}

item {
  id: 26
  name: 'y'
}

item {
  id: 27
  name: 'z'
}
```

Anexo 3: Script para la conversión de archivos XML a TFRecord:

```
""" Sample TensorFlow XML-to-TFRecord converter

usage: generate_tfrecord.py [-h] [-x XML_DIR] [-l LABELS_PATH] [-o OUTPUT_PATH] [-i
IMAGE_DIR] [-c CSV_PATH]

optional arguments:
  -h, --help            show this help message and exit
  -x XML_DIR, --xml_dir XML_DIR
                        Path to the folder where the input .xml files are stored.
  -l LABELS_PATH, --labels_path LABELS_PATH
                        Path to the labels (.pbtxt) file.
  -o OUTPUT_PATH, --output_path OUTPUT_PATH
                        Path of output TFRecord (.record) file.
  -i IMAGE_DIR, --image_dir IMAGE_DIR
                        Path to the folder where the input image files are stored. Defaults to the same
directory as XML_DIR.
  -c CSV_PATH, --csv_path CSV_PATH
                        Path of output .csv file. If none provided, then no file will be written.
"""

import os
import glob
import pandas as pd
import io
```

```
import xml.etree.ElementTree as ET
import argparse

os.environ['TF_CPP_MIN_LOG_LEVEL'] = '2'    # Suppress TensorFlow logging (1)
import tensorflow.compat.v1 as tf
from PIL import Image
from object_detection.utils import dataset_util, label_map_util
from collections import namedtuple

# Initiate argument parser
parser = argparse.ArgumentParser(
   description="Sample TensorFlow XML-to-TFRecord converter")
parser.add_argument("-x",
                    "--xml_dir",
                    help="Path to the folder where the input .xml files are stored.",
                    type=str)
parser.add_argument("-l",
                    "--labels_path",
                    help="Path to the labels (.pbtxt) file.", type=str)
parser.add_argument("-o",
                    "--output_path",
                    help="Path of output TFRecord (.record) file.", type=str)
parser.add_argument("-i",
                    "--image_dir",
                    help="Path to the folder where the input image files are stored. "
                        "Defaults to the same directory as XML_DIR.",
                    type=str, default=None)
parser.add_argument("-c",
                    "--csv_path",
                    help="Path of output .csv file. If none provided, then no file will be "
                        "written.",
                    type=str, default=None)

args = parser.parse_args()

if args.image_dir is None:
   args.image_dir = args.xml_dir

label_map = label_map_util.load_labelmap(args.labels_path)
label_map_dict = label_map_util.get_label_map_dict(label_map)


def xml_to_csv(path):
```

```
    """Iterates through all .xml files (generated by labelImg) in a given directory and combines
    them in a single Pandas dataframe.

    Parameters:
    ----------
    path : str
        The path containing the .xml files
    Returns
    -------
    Pandas DataFrame
        The produced dataframe
    """

    xml_list = []
    for xml_file in glob.glob(path + '/*.xml'):
        tree = ET.parse(xml_file)
        root = tree.getroot()
        filename = root.find('filename').text
        width = int(root.find('size').find('width').text)
        height = int(root.find('size').find('height').text)
        for member in root.findall('object'):
            bndbox = member.find('bndbox')
            value = (filename,
                     width,
                     height,
                     member.find('name').text,
                     int(bndbox.find('xmin').text),
                     int(bndbox.find('ymin').text),
                     int(bndbox.find('xmax').text),
                     int(bndbox.find('ymax').text),
                     )
            xml_list.append(value)
    column_name = ['filename', 'width', 'height',
                   'class', 'xmin', 'ymin', 'xmax', 'ymax']
    xml_df = pd.DataFrame(xml_list, columns=column_name)
    return xml_df


def class_text_to_int(row_label):
    return label_map_dict[row_label]


def split(df, group):
```

```
    data = namedtuple('data', ['filename', 'object'])
    gb = df.groupby(group)
    return [data(filename, gb.get_group(x)) for filename, x in zip(gb.groups.keys(), gb.groups)]


def create_tf_example(group, path):
    with tf.gfile.GFile(os.path.join(path, '{}'.format(group.filename)), 'rb') as fid:
        encoded_jpg = fid.read()
    encoded_jpg_io = io.BytesIO(encoded_jpg)
    image = Image.open(encoded_jpg_io)
    width, height = image.size

    filename = group.filename.encode('utf8')
    image_format = b'jpg'
    xmins = []
    xmaxs = []
    ymins = []
    ymaxs = []
    classes_text = []
    classes = []

    for index, row in group.object.iterrows():
        xmins.append(row['xmin'] / width)
        xmaxs.append(row['xmax'] / width)
        ymins.append(row['ymin'] / height)
        ymaxs.append(row['ymax'] / height)
        classes_text.append(row['class'].encode('utf8'))
        classes.append(class_text_to_int(row['class']))

    tf_example = tf.train.Example(features=tf.train.Features(feature={
        'image/height': dataset_util.int64_feature(height),
        'image/width': dataset_util.int64_feature(width),
        'image/filename': dataset_util.bytes_feature(filename),
        'image/source_id': dataset_util.bytes_feature(filename),
        'image/encoded': dataset_util.bytes_feature(encoded_jpg),
        'image/format': dataset_util.bytes_feature(image_format),
        'image/object/bbox/xmin': dataset_util.float_list_feature(xmins),
        'image/object/bbox/xmax': dataset_util.float_list_feature(xmaxs),
        'image/object/bbox/ymin': dataset_util.float_list_feature(ymins),
        'image/object/bbox/ymax': dataset_util.float_list_feature(ymaxs),
        'image/object/class/text': dataset_util.bytes_list_feature(classes_text),
        'image/object/class/label': dataset_util.int64_list_feature(classes),
    }))
```

```
    return tf_example


def main(_):

    writer = tf.python_io.TFRecordWriter(args.output_path)
    path = os.path.join(args.image_dir)
    examples = xml_to_csv(args.xml_dir)
    grouped = split(examples, 'filename')
    for group in grouped:
        tf_example = create_tf_example(group, path)
        writer.write(tf_example.SerializeToString())
    writer.close()
    print('Successfully created the TFRecord file: {}'.format(args.output_path))
    if args.csv_path is not None:
        examples.to_csv(args.csv_path, index=None)
        print('Successfully created the CSV file: {}'.format(args.csv_path))


if __name__ == '__main__':
    tf.app.run()
```

Anexo 4: Configuración del archivo pipeline.config:

```
model {

ssd {

num_classes: 27 # Indicar la cantidad de etiquetas

image_resizer {

fixed_shape_resizer {

height: 640

width: 640

}

}

feature_extractor {

type: "ssd_mobilenet_v2_fpn_keras"

depth_multiplier: 1.0

min_depth: 16
```

conv_hyperparams {
regularizer {
l2_regularizer {
weight: 3.9999998989515007e-05
}
}
initializer {
random_normal_initializer {
mean: 0.0
stddev: 0.009999999776482582
}
}
activation: RELU_6
batch_norm {
decay: 0.996999979019165
scale: true
epsilon: 0.0010000000474974513
}
}
use_depthwise: true
override_base_feature_extractor_hyperparams: true
fpn {
min_level: 3
max_level: 7
additional_layer_depth: 128
}
}
box_coder {
faster_rcnn_box_coder {
y_scale: 10.0

```
x_scale: 10.0
height_scale: 5.0
width_scale: 5.0
}
}
matcher {
argmax_matcher {
matched_threshold: 0.5
unmatched_threshold: 0.5
ignore_thresholds: false
negatives_lower_than_unmatched: true
force_match_for_each_row: true
use_matmul_gather: true
}
}
similarity_calculator {
iou_similarity {
}
}
box_predictor {
weight_shared_convolutional_box_predictor {
conv_hyperparams {
regularizer {
l2_regularizer {
weight: 3.9999998989515007e-05
}
}
initializer {
random_normal_initializer {
mean: 0.0
```

stddev: 0.009999999776482582
}
}
activation: RELU_6
batch_norm {
decay: 0.996999979019165
scale: true
epsilon: 0.0010000000474974513
}
}
depth: 128
num_layers_before_predictor: 4
kernel_size: 3
class_prediction_bias_init: -4.599999904632568
share_prediction_tower: true
use_depthwise: true
}
}
anchor_generator {
multiscale_anchor_generator {
min_level: 3
max_level: 7
anchor_scale: 4.0
aspect_ratios: 1.0
aspect_ratios: 2.0
aspect_ratios: 0.5
scales_per_octave: 2
}
}
post_processing {

batch_non_max_suppression {
score_threshold: 9.99999993922529e-09
iou_threshold: 0.6000000238418579
max_detections_per_class: 100
max_total_detections: 100
use_static_shapes: false
}
score_converter: SIGMOID
}
normalize_loss_by_num_matches: true
loss {
localization_loss {
weighted_smooth_l1 {
}
}
classification_loss {
weighted_sigmoid_focal {
gamma: 2.0
alpha: 0.25
}
}
classification_weight: 1.0
localization_weight: 1.0
}
encode_background_as_zeros: true
normalize_loc_loss_by_codesize: true
inplace_batchnorm_update: true
freeze_batchnorm: false
}
}

```
train_config {
batch_size: 4 #
data_augmentation_options {
random_horizontal_flip {
}
}
data_augmentation_options {
random_crop_image {
min_object_covered: 0.0
min_aspect_ratio: 0.75
max_aspect_ratio: 3.0
min_area: 0.75
max_area: 1.0
overlap_thresh: 0.0
}
}
sync_replicas: true
optimizer {
momentum_optimizer {
learning_rate {
cosine_decay_learning_rate {
learning_rate_base: 0.07999999821186066
total_steps: 50000
warmup_learning_rate: 0.026666000485420227
warmup_steps: 1000
}
}
momentum_optimizer_value: 0.8999999761581421
}
use_moving_average: false
```

}

fine_tune_checkpoint:"pre-trained-models\\ssd_mobilenet_v2_fpnlite_640x640_coco17_tpu-8\\checkpoint\\ckpt-0" # Indicar el directorio en el que se encuentra el checkpoint del modelo pre-entrenado

num_steps: 50000

startup_delay_steps: 0.0

replicas_to_aggregate: 8

max_number_of_boxes: 100

unpad_groundtruth_tensors: false

fine_tune_checkpoint_type: "detection"

fine_tune_checkpoint_version: V2

}

train_input_reader {

label_map_path: "annotations\\label_map.pbtxt" #Directorio en donde se encuentra el mapa de etiquetas

tf_record_input_reader {

input_path: "annotations\\train.record" #Directorio en donde se encuentra el archivo TFRecord de las imágenes de entrenamiento

}

}

eval_config {

metrics_set: "coco_detection_metrics"

use_moving_averages: false

}

eval_input_reader {

label_map_path: "annotations\\label_map.pbtxt" #Directorio en donde se encuentra el mapa de etiquetas

shuffle: false

num_epochs: 1

tf_record_input_reader {

input_path: "annotations\\test.record" #Directorio en donde se encuentra el archivo TFRecord de las imágenes de prueba

```
}
}
```

Anexo 5: Congelar un gráfico de inferencia.

```
# Desde el directorio TensorFlow/models/research/
python object_detection/export_tflite_graph_tf2.py \
    --pipeline_config_path ../../workspace/models/my_ssd_mobnet/pipeline.config \ # Directorio del archivo pipeline.config
    --trained_checkpoint_dir ../../workspace/models/my_ssd_mobnet/pipeline.config/checkpoint \ # Directorio del archivo checkpoint
    --output_directory ../../workspace/exported-models\ # Directorio para exporter el modelo
```

Anexo 6: Convertir modelo a formato TensorFlow Lite.

```
#Ejecutar desde el directorio en donde se encuentra la carpeta del modelo guardado como gráfico de inferencia. Para este caso es TensorFlow/workspace/exported-models/
import tensorflow as tf
converter = tf.lite.TFLiteConverter.from_saved_model("prueba2/saved_model") # Directorio del archivo savel_model.
tflite_model = converter.convert()
with open('model.tflite', 'wb') as f:
  f.write(tflite_model)
```

Anexo 7: Agregar MetaData al modelo.

```
#Ejecutar dentro del directorio del modelo TensorFlow Lite. Para este caso TensorFlow/workspace/esported-models/prueba2/
from tflite_support.metadata_writers import object_detector
from tflite_support.metadata_writers import writer_utils
ObjectDetectorWriter = object_detector.MetadataWriter
```

```
_MODEL_PATH = "model.tflite" # Nombre del modelo de TensoFlow Lite.
# Task Library expects label files that are in the same format as the one below.
_LABEL_FILE = "label_map.txt" # Nombre del archivo que contiene el mapa de etiquetas.
_SAVE_TO_PATH = "DeteccionLetras_metadata2.tflite" # Nombre del archivo final.
# Normalization parameters is required when reprocessing the image. It is
# optional if the image pixel values are in range of [0, 255] and the input
# tensor is quantized to uint8. See the introduction for normalization and
# quantization parameters below for more details.
# https://www.tensorflow.org/lite/convert/metadata#normalization_and_quantization_parameters)
_INPUT_NORM_MEAN = 127.5
_INPUT_NORM_STD = 127.5

# Create the metadata writer.
writer = ObjectDetectorWriter.create_for_inference(
    writer_utils.load_file(_MODEL_PATH), [_INPUT_NORM_MEAN], [_INPUT_NORM_STD],
    [_LABEL_FILE])
# Verify the metadata generated by metadata writer.
print(writer.get_metadata_json())
# Populate the metadata into the model.
writer_utils.save_file(writer.populate(), _SAVE_TO_PATH)
```