FACULTAD DE INGENIERÍAS Y CIENCIAS AGROPECUARIAS

DESARROLLO DE UN PROTOTIPO DE SISTEMA DE CATALGACIÓN AUTOMÁTICA DE BOTELLAS PET JUNTO CON LA INTERFAZ GRÁFICA QUE PERMITA MANEJARLO

AUTOR

Juan Francisco Rosero Pozo

AÑO

2018

FACULTAD DE INGENIERÍAS Y CIENCIAS AGROPECUARIAS

DESARROLLO DE UN PROTOTIPO DE SISTEMA DE CATALOGACIÓN AUTOMÁTICA DE BOTELLAS PET JUNTO CON LA INTERFAZ GRÁFICA QUE PERMITA MANEJARLO

Trabajo de Titulación presentado en conformidad con los requisitos establecidos para optar por el título de Ingeniero en Electrónica y Redes de Comunicación.

Profesor Guía

Mtr. Jorge Luis Rosero Beltrán

Autor

Juan Francisco Rosero Pozo

Año

2018

DECLARACIÓN DEL PROFESOR GUÍA

“Declaro haber dirigido el trabajo, Desarrollo de un prototipo de sistema de catalogación automática de botellas PET junto con la interfaz gráfica que permita manejarlo, de Juan Francisco Rosero Pozo, en el semestre 2018-1, orientando sus conocimientos y competencias para un eficiente desarrollo del tema escogido y dando cumplimiento a todas las disposiciones vigentes que regulan los Trabajos de Titulación”

_____

Jorge Luis Rosero Beltrán

Master en Ciencias con Especialidad en Automatización.

CI: 1803610185

DECLARACIÓN DEL PROFESOR CORRECTOR

“Declaro haber revisado este trabajo, Desarrollo de un prototipo de sistema de catalogación automática de botellas PET junto con la interfaz gráfica que permita manejarlo, de Juan Francisco Rosero Pozo, en el semestre 2018-1, dando cumplimiento a todas las disposiciones vigentes que regulan los Trabajos de Titulación.”

_____

David Fernando Pozo Espín

Máster universitario en automática y robótica.

CI: 1717340143

DECLARACIÓN DE AUTORÍA DEL ESTUDIANTE

“Declaro que este trabajo es original, de mi autoría, que se han citado las fuentes correspondientes y que en su ejecución se respetaron las disposiciones legales que protegen los derechos de autor vigentes”.

_____

Juan Francisco Rosero Pozo

CI: 0401584909

AGRADECIMIENTOS

Quiero agradecer a todos los verdaderos maestros que he tenido a lo largo mi vida. A esas personas que no sólo enseñaron, sinó que despertaron las ganas de aprender y el deseo de mejorar el mundo a través del conocimiento.

DEDICATORIA

Quiero dedicar este trabajo a mi familia. A esas cuatro personas con las que tuve y tengo la fortuna de compartir mi vida. Es de ustedes que saco el deseo de superarme a mí mismo. Mi corazón siempre estará donde sea que se encuentren.

RESUMEN

El siguiente documento presenta el desarrollo de un sistema automático de clasificación de envases. El prototipo a desarrollar se encarga de recibir los envases y cuenta con una interfaz mediante la cual el usuario puede ingresar su cédula y recibir una recompensa por los envases entregados. Este prototipo se encuentra conectado a un servidor escrito en Python y desplegado en Google Compute Engine que cuenta con una red neuronal convolucional capaz de reconocer entre cinco tipos diferentes de envases y llevar el control de recompensas para cada usuario. Finalmente se implementó un servidor desarrollado en Node Js y desplegado en Heroku, encargado de presentarle al usuario una página web en la que pueda consultar sus recompensas. Los dos servidores se encuentran conectados a una base de datos MongoDB hosteada en mLab.

Los envases son principalmente botellas PET, aunque en el desarrollo se demostró que el sistema es capaz de clasificar otros tipos de envases como latas.

El documento también contiene los análisis de las posibles soluciones en cuanto a la clasificación de los envases y la construcción del prototipo con un enfoque en la reducción de costos de producción del mismo. Además cuenta con una sección donde se discute un posible modelo de negocio que aproveche las ventajas del sistema.

ABSTRACT

The following document presents the development of an automatic container classification system. The prototype to develop is capable of receiving the containers and has an interface which allows users to register their contributions in order to receive a reward later. This prototype is connected to a Python server hosted in Google Compute Engine which runs a Convolutional Neural Network capable of recognizing five different types of containers and register the rewards for each user. Finally, a NodeJs server hosted on Heroku was developed to allow users to see their rewards via a web page. Both servers share a MongoDB database hosted on mLab.

The containers to be classified are mainly PET bottles. Furthermore, development showed that other types of containers, like aluminium cans, can be classified as well.

This document also contains the análisis on the possible solutions to the classification problem and the construction of the prototype focusing on reducing production costs. Also, a discussion about a business model that could take advantage of the system's benefits is included.

ÍNDICE

1. INTRODUCCIÓN

Es deseable desarrollar sistemas que puedan catalogar objetos de la misma manera en la que un ser humano puede hacerlo. Actualmente, existen varios acercamientos para solucionar este problema que varían en el tipo de software que se utiliza y los sensores que proveen los datos sobre los cuales tomar la decisión. Está el caso de Ohtani y Baba en 2012, quienes lograron con éxito reconocer clases botellas transparentes y su posición utilizando sensores ultrasónicos junto con una red neuronal que interpretara los datos obtenidos.

El uso de una Red Neuronal, un software que emula las capacidades del cerebro humano al replicar el funcionamiento de varias neuronas conectadas entre si, es de especial interés pues además de haber probado ser una eficaz forma de construir inteligencia artificial, cuenta en la actualidad con herramientas que facilitan su implementación.

Específicamente, las Redes Neuronales Convolucionales, un tipo especial de red neuronal que replica el funcionamiento de nuestra corteza visual, muestran gran precisión a la hora de catalogar imágenes. Es el caso de Krizhevsky, Sutskever y Hinton, quienes, también en 2012, entrenaron una Red Neuronal Convolucional que clasificara las imágenes del ImageNet. Sus avances les permitieron obtener resultados ganadores en los concursos ILSVRC-2010 e ILSVRC-2012.

Szegedy, Toshev y Erhan (2013) fueron un paso más adelante y las utilizaron para reconocer no solo el objeto dentro de una foto sinó también su posición. Más recientemente Luís A. Alexandre (2017) las utilizó para reconocer imágenes en formato RGB-D (Imágenes RGB con una medida de profundidad). Estas aplicaciones le permitirían a un robot tener una mejor percepción del ambiente que lo rodea.

Por otro lado, la transferencia de aprendizaje descrita por Yosinski, Clune, Bengio y Lipson(2014) y por Razavian, Azizpour, Sullivan y Carlsson (2014) les permite a las Redes Neuronales Convolucionales utilizar el aprendizaje de otros modelos para obtener resultados consistentes con relativamente muy pocos datos.

A continuación, se describe la aplicación de estas tecnologías y técnicas para el desarrollo de un prototipo de bajo costo que permita clasificar envases de cinco tipos diferentes. El prototipo cuenta con una interfaz que le permite al usuario entregar un envase y recibir una recompensa. Este se encuentra conectado con un un servidor que corre la Red Neuronal y registra las recompensas para cada usuario. Finalmente se implementó un servidor web mediante el cual el usuario pueda consultar sus ganancias.

Al final del documento se discuten los resultados de la implementación y se propone un posible modelo de negocio que aproveche las características del sistema.

1.1 Objetivos

El objetivo principal del proyecto es crear el prototipo de un sistema de catalogación automática de botellas PET junto con la interfaz gráfica que permita manejarlo. Además, el usuario debe poder consultar los beneficios obtenidos por entregar la botella.

Los objetivos secundarios son los siguientes:

- Que el sistema de catalogación sea lo más rápido y preciso posible.
- Crear una interfaz fácil de manejar para el usuario
- Reducir lo más posible el costo de construcción del sistema.

- Realizar un análisis sobre la posible aceptación en el mercado teniendo en cuenta los costos aproximados de producción.
- Crear un sistema fácil de integrar para empresas que deseen ubicar una de estas máquinas en sus establecimientos.

1.2 Alcance

Se pretende diseñar un sistema capaz de catalogar botellas entregadas por el usuario de manera automática. Luego, el sistema debe permitirle consultar las ganancias obtenidas por la entrega de las mismas. El usuario maneja el dispositivo mediante una pantalla táctil y una interfaz gráfica fácil de controlar.

2. MARCO TEÓRICO

2.1 Redes Neuronales

Si se compara las capacidades de una computadora con las de un humano se puede ver claramente que mientras la computadora puede procesar información a grandes velocidades, no tiene la capacidad de resolver problemas complejos como la de un ser humano.

¿Y si se pudiera replicar las habilidades del cerebro humano en una máquina?. Hace ya algunos años, investigadores han trabajado en desarrollar maneras de adaptar el funcionamiento de las neuronas del cerebro a una computadora. A estos sistemas se los denomina Redes Neuronales.

2.1.1 Motivaciones para el uso de Redes Neuronales

Durante los últimos años, el desarrollo de sistemas de computación ha visto como deseables las características propias de un cerebro humano. Si bien para nosotros es realmente sencillo reconocer un rostro entre una multitud, para nuestras computadoras basadas en la arquitectura de Vonn Neuman, es una tarea sumamente ardua (Jain, A.K. , Moohiuddin, K & Mao, J. , 1996).

La efectividad de las redes neuronales en dichas tareas es superior gracias a la arquitectura bajo la que se construye. Dicha arquitectura intenta repartir el trabajo de procesamiento entre varios nodos (neuronas), mientras que la arquitectura de Vonn Neuman concentra el trabajo en uno o varios procesadores de alta velocidad. La tabla número 1 (Jain et al., 1996) muestra algunas otras diferencias entre las dos arquitecturas.

Tabla 1.

Una computadora Vonn Neumann vs un Sistema Neurológico

	Computadora Von Neumann	Sistema neuronal biológico
Procesador	Complejo Alta velocidad Uno o algunos	Simple Baja velocidad Una gran cantidad
Memoria	Separada del procesador Localizada	Integrada en el procesador Distribuida
Cómputo	Centralizado Secuencial Programas almacenados	Distribuido En paralelo Autodidacta
Confiabilidad	Vulnerable	Robusta
Campo de acción	Operaciones numéricas y simbólicas	Problemas perceptuales
Ambiente operativo	Bien definido y delimitado	No muy bien definido y sin limitaciones

Adaptado de Jain, 1996

Desarrollar este tipo de sistemas requiere conocimientos de muchas materias y resulta ser muy complicado. Sin embargo, las ventajas que estas pueden brindar sobrepasan las dificultades de su desarrollo. Además, mientras se desarrollan, las mismas se convierten en herramientas que facilitan su desarrollo.

2.1.2 Funcionamiento de una neurona

Es importante resaltar el funcionamiento de las neuronas del cerebro humano para entender la manera en la que una Red Neuronal funciona.

Una neurona es una célula del cuerpo especializada en procesar información. Está constituida principalmente por un núcleo, dendritas, cuerpo y axioma.

El cuerpo y el núcleo de esta célula se encargan de las funciones de supervivencia de la célula. Por otro lado, el axioma y las dendritas funcionan como transmisores y receptores respectivamente. Una sinapsis es la unión de las dendritas de una neurona con el axioma de otra contigua. Mediante esta unión las neuronas pueden transmitir impulsos eléctricos y procesar información.

Cuando un impulso eléctrico alcanza una sinapsis, esta se llena de químicos llamados neurotransmisores que bien pueden reforzar o atenuar la señal dependiendo del tipo de sinapsis que se encuentre. Cuando un camino ha sido recorrido varias veces por un impulso eléctrico, el resultado es la capacidad del cerebro de aprender o recordar algo, pues las sinapsis entre esas neuronas se ve reforzada.

Nuestro cerebro cuenta con aproximadamente 10^{11} neuronas (hay un número parecido de estrellas en la vía láctea). Cada una de estas puede estar conectada de 10^{3} a 10^{4} neuronas diferentes. En total unas 10^{14} a 10^{15} conexiones en total. Por otro lado, por cada una de ellas no pasan más de unos cientos de pulsos por segundo, que es mucho menos que las velocidades de transmisión de nuestras redes más avanzadas.

Estos números nos ayudan a entender que el sistema neuronal no se basa en la complejidad de sus componentes, ni en la velocidad de transmisión de los mismos, sino en la capacidad de trabajar de manera conjunta.

2.1.3 Funcionamiento de una Red Neuronal

2.1.3.1 Modelos computacionales

McCulloch y Pitts (1943) propusieron un modelo computacional binario para describir el funcionamiento de una neurona.

En la Figura 1 se puede ver que lo que en realidad se tiene es la suma de n entradas a las cuales se les asigna un peso w. El resultado de esta suma es introducido a una función que tiene como salida un 0 o un 1 dependiendo de si la entrada sobrepasa un cierto límite u.

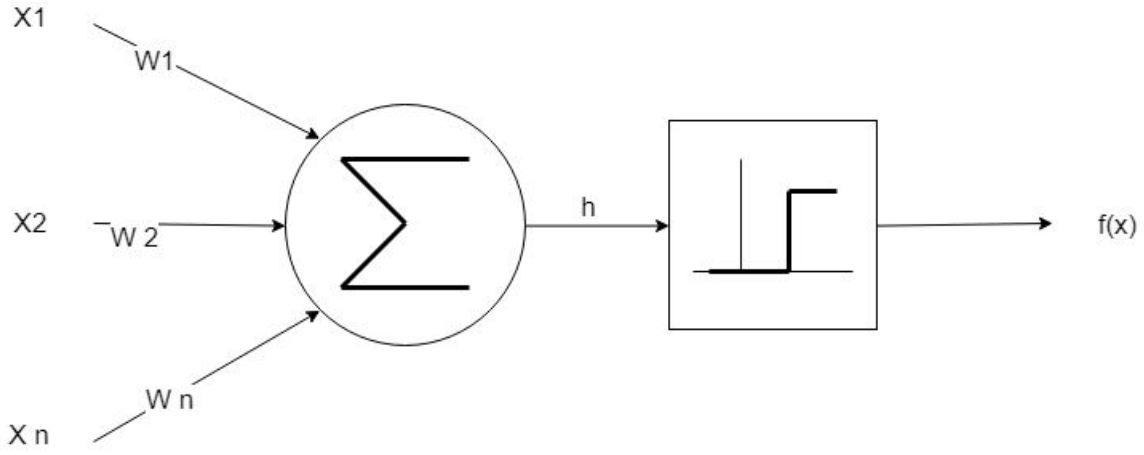


Figura 1. Modelo computacional binario para representar una neurona.

Adaptado de: Jain, 1996

La figura anterior se puede representar mediante la siguiente ecuación:

$$f(x) = sign\left(\left(\sum_{i=1}^{n} w_i\, x_i\right) + u\right)$$

En la cual x_i representa cada una de las entradas a la neurona, que multiplicadas por un peso w_i. Si el resultado de la sumatoria de estos

productos sobrepasan un umbral u , el resultado es 1 (la neurona se activa). En caso contrario se obtendría un 0 (la neurona no se activa).

Este modelo puede ser considerado una sobre simplificación de lo que en realidad pasa dentro de una neurona. Sin embargo, es útil para entender el los principios de funcionamiento de la misma.

2.1.3.2 Arquitectura de una Red Neuronal

Una red entonces no es más que un grupo de neuronas conectadas entre sus entradas (dendritas) y salidas (axiomas) por enlaces a los cuales se les asigna un peso. Dependiendo de cómo se den estas conexiones se puede dividir a las Redes Neuronales en dos grupos.

- *Feed-foward Networks.* Este tipo de redes son las más comunes. Si se dibujara un gráfico de las mismas, este no tendría lazos. Sus conexiones se agrupan en capas conectadas unidireccionalmente. Son estáticas en el sentido de que no pueden aprender de los resultados que generan. Su cantidad de salidas es fija.
- *Recurrent Networks.* El gráfico de este tipo de redes contiene lazos. Esto implica que los resultados que la red genera son utilizados para modificar los pesos de la red dependiendo de una función que determine si el resultado fue correcto o no. Este proceso les permite aprender con la práctica y de ser necesario, cambiar el número de salidas.

La Figura 2 muestra los gráficos para los dos tipos de redes.

Hay otros parámetros que también son utilizados para describir la arquitectura de una red neuronal. A estos se les denomina Hiperparámetros. La cantidad y tipo de hiperparámetros puede variar en función del tipo de red que se esté construyendo.

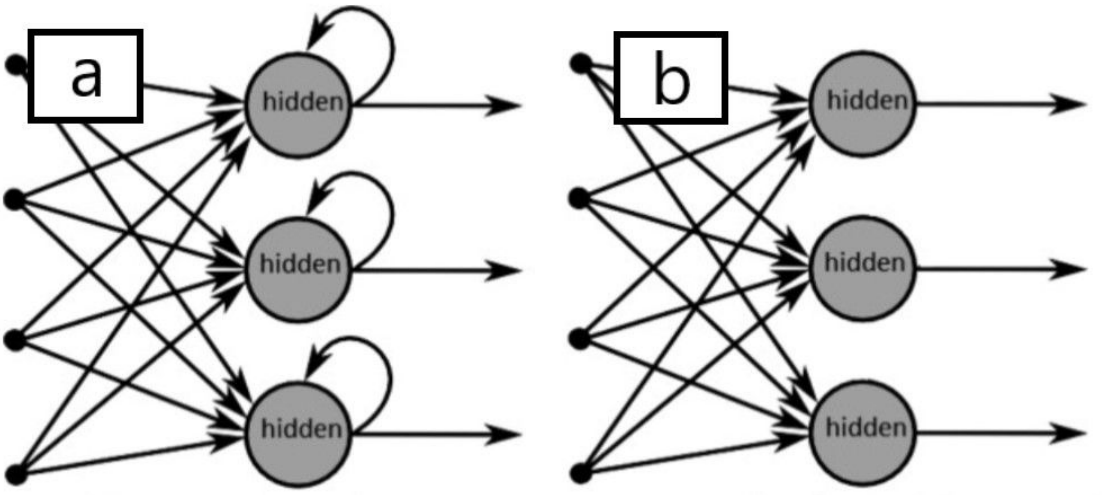


Figura 2. Tipos de redes neuronales

a) *Recurrent*

b) *Feed Forward*

Adaptado de: Mulder, 2005

2.1.3.3 Aprendizaje de las Redes Neuronales

Si intentamos retratar el concepto de aprendizaje aplicado a una Red Neuronal, se puede definir como la modificación de la arquitectura de la red y de sus pesos asociados con el fin de realizar una nueva tarea (Jain et al., 1996).

A breves rasgos, para entrenar la red se le proporciona datos y una respuesta correcta. Si la red acierta, entonces los pesos de las neuronas se actualizan de tal manera que ese resultado se vuelva a dar en el futuro.

Los primeros pesos con los que cuenta cada neurona de la red pueden ser asignados aleatoriamente. Mientras más se entrena la red, más aprende a realizar determinada tarea.

Esto le da la capacidad de adaptarse sin seguir ninguna regla preestablecida, lo cual resulta en extremo útil e interesante.

2.1.3.4 Funciones de pérdida

En el proceso de aprendizaje de una red neuronal, no es suficiente saber si la respuesta que se obtuvo está bien o mal. Se tiene que poder determinar que tan bien o que tan mal respondió la red. Es para ello que se utiliza una función de pérdida. Las entradas de esta función son la valoración de la red para un caso particular y la respuesta correcta.

Dependiendo del valor que se obtenga con esta función, se pueden modificar los pesos de la red de una manera más drástica o leve.

Algunas funciones famosas para ser utilizadas como funciones de pérdida son:

1. Multiclass Support Vector Machine loss (SVM)
2. Softmax

2.1.3.5 Backpropagation

Backpropagation (propagación en reversa), es el proceso mediante el cual las redes neuronales actualizan sus pesos con el fin de obtener los resultados deseados. Se puede decir que es la manera en la que estas aprenden a resolver el problema para el que fueron diseñadas.

Este proceso se puede dividir en 4 pasos distintos: el paso hacia adelante, la función de pérdida, el paso hacia atrás y la actualización de los pesos.(D. Adit, 2016).

El primero se refiere al paso de una muestra por la red, que en una primera instancia tendrá sus pesos asignados de manera randómica. Es de esperarse que el mismo sea poco preciso, pero lo que interesa por ahora es corregir ese error. Tómese como ejemplo una red encargada de clasificar 3 tipos de imágenes (las dimensiones de la imagen no son relevantes por el momento). El resultado final de la red después de realizar todo el procesamiento sería un vector con las probabilidades de que esa imagen pertenezca a determinada

clase. Puede que después de que la red analice la primera imagen dé como resultado un vector [0.33,0.33,0.33]. Esto por supuesto no brinda mucha información sobre la clase a la que pertenece la imagen, pero teniendo el vector respuesta [1,0,0] (es decir, la imagen pertenece a la primera clase), podemos empezar a actualizar los pesos de nuestra red.

El siguiente paso, es la función de pérdida. Esta se encarga de determinar que tan errado o acertado es el resultado obtenido. Hay varias funciones que se pueden utilizar para calcular este valor.

Con ese valor inicia el paso hacia atrás, que lo que busca es determinar qué tan culpable es cada uno de los pesos de la red en el resultado obtenido.

Este problema se puede abordar como uno de optimización, en el cual lo que queremos encontrar son los pesos que nos dan el menor error posible. Matemáticamente, lo que se haría es obtener la derivada del error con respecto a los pesos. El gradiente obtenido representará cuanto contribuyó ese peso al resultado final.

La Figura 3, muestra una neurona simple que, en el paso hacia adelante (izquierda), calcula un resultado z, dados los parámetros x y y . Luego, se utiliza z para calcular un error L. Finalmente en el paso hacia atrás (derecha), podemos minimizar el error con respecto a z utilizando una derivada. Ya que los valores importantes son los de x y y para los cuales L es mínimo, se puede utilizar la regla de la cadena para representar dicha derivada en función de las derivadas de x y y.

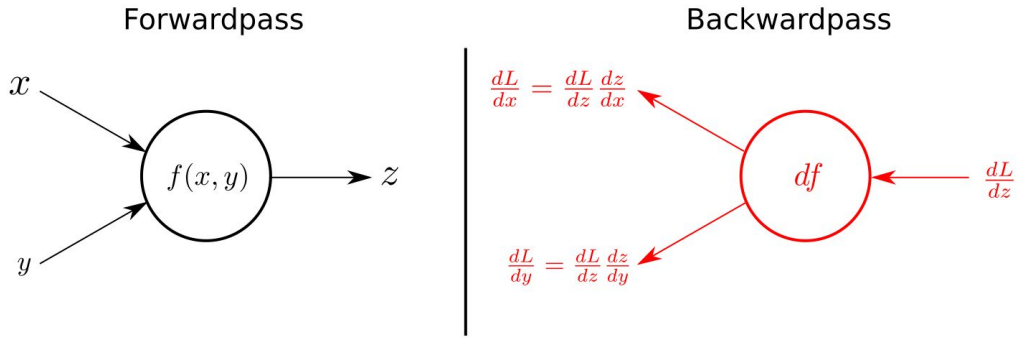


Figura 3. Paso hacia adelante y hacia atrás en el proceso de *backpropagation.*

Tomado de: M. Agarwa, 2017

Esta forma de calcular dicho valor se denomina **gradiente descendiente**. Como se puede esperar, cada vez que se calculan los gradientes para las capas inferiores, el valor de los mismos desciende de manera exponencial. Cuando el gradiente desciende demasiado rápido y los pesos que se encuentran más profundos en la red no se actualizan apropiadamente se produce el problema del **gradiente desvaneciente**.

Finalmente, una vez que se tiene el gradiente por cada peso de cada nodo se lleva a cabo el paso final que es modificar los pesos de la red neuronal, lo que concluye el proceso de entrenamiento para determinado caso.

Un hiperparámetro importante en este proceso es el **factor de aprendizaje**, el cual modifica de manera lineal cuanto se modifican los pesos en el proceso de *backpropagation*. Mientras más grande, la red puede aprender más rápido, pero puede darse también el caso en el que los pasos sean demasiado grandes como para llegar al punto en el que se minimiza el error. (D. Adit, 2016).

2.1.3.6 Problemas relacionados con las redes neuronales

La efectividad de una red neuronal se puede ver afectado por algunos problemas comunes que se describen a continuación:

Cantidad de datos

Cualquiera que sea la red neuronal que se pretende implementar, necesita datos de entrenamiento y prueba con los que aprender y posteriormente validar ese aprendizaje.

Se puede decir que la calidad de las predicciones de una Red Neuronal es directamente proporcional a la cantidad y calidad de los datos que se le proporciona. Esto presenta un reto pues en algunos casos conseguir el grupo de datos deseado presenta inconvenientes.

Es por ello que se ha vuelto común la práctica de tomar un grupo de datos y procesarlo para obtener más muestras. En el caso de sistemas de reconocimiento visual esto incluye insertar modificaciones aleatorias a las muestras. A esto se le denomina *Image Augmentation*. Otra solución es la validación cruzada (*Crossvalidation),* en la cual se divide el grupo de datos en secciones diferentes, para luego entrenar con cada una de ellas y realizar pruebas con el resto sucesivamente.

Linealidad

En general, el funcionamiento de una red neuronal se puede representar por una función $f(x,w)$, donde x representa las entradas y w sus pesos. Debido a la naturaleza de las predicciones que se desean alcanzar con las redes neuronales, es poco probable que la función de una red bien diseñada sea lineal. Esto significa que un incremento en cierta característica de la entrada

corresponde directamente a la obtención de un resultado u otro, lo cual no siempre es el caso en las tareas de reconocimiento.

Hay varias técnicas para reducir la linealidad del sistema. Uno de ellos es utilizar funciones de activación que no sean lineales (como *sigmoid* o *tanh*) en los nodos de la red.

Overfitting

Este problema se da cuando la red neuronal es eficiente solamente para un grupo determinado de datos. Esto es totalmente contrario a lo que se desea pues una red neuronal debería poder emitir una decisión sin importar el origen de los datos a evaluar.

El causante principal de este problema es el de tener un solo grupo de datos para entrenar y probar la red, lo que hace que el ajuste de los hiperparámetros naturalmente favorezcan a dichos datos.

La solución más sencilla es separar a nuestro grupo de datos en dos, un grupo de entrenamiento y otro de prueba. Normalmente, los porcentajes varían entre un 50-90% para entrenamiento y el resto para pruebas.

Otras soluciones más complicadas incluyen dividir aún más los datos de entrenamiento para conseguir una red aún más imparcial. Todo depende de la cantidad de datos disponibles e hiperparámetros a configurar.

2.1.4 Redes Neuronales Convolucionales

Si bien las Redes Neuronales originales utilizaban el concepto de la neurona como base para su implementación, sus hijas, las Redes Neuronales Convolucionales utilizan la corteza visual como inspiración para solucionar el problema del reconocimiento de imágenes.

Esta capacidad especial del cerebro de reconocer imágenes con precisión y sin esfuerzo se obtiene por la manera en la que la corteza visual procesa información. Básicamente, lo que hace es construir información de alto nivel utilizando otra de bajo nivel. Para ello, se divide a sí misma en varias regiones especializadas en reconocer una característica específica, bien sea un color, una orientación, una tonalidad, etc.

Luego, sobre esta información de bajo nivel se puede construir abstracciones de más alto nivel hasta llegar al resultado que es determinar la clase a la cual pertenece determinada imagen.

2.1.4.1 Capas convolucionales

La manera en la que los humanos perciben una imagen es muy diferente de la de una computadora. Los primeros ven colores, tonalidades y formas. Por otro lado, para las computadoras una imagen no es más que un arreglo de valores que representan el color que tiene la imagen en determinado punto.

Con este concepto de la abstracción y el hecho de que la imagen no es más que un arreglo de valores, podemos crear un mecanismo que emule nuestra corteza cerebral. Este es el objetivo de una capa convolucional.

La manera más sencilla de imaginar el funcionamiento de esta capa es como una luz que ilumina una sección de una imagen. Esta luz empieza iluminando la esquina superior izquierda y se mueve ordenadamente sobre toda la imagen hasta haberla recorrido completamente. Como si la estuviera escaneando poco a poco. (D. Adit, 2016).

Con el fin de ejemplificar este concepto se puede suponer que la imagen cuenta con un área de 32x32. La luz por otro lado puede iluminar un área de 5x5. En términos técnicos, a esta luz se le llama **filtro**, y el área que ilumina se

denomina **campo receptivo**. Ahora, desde el punto de vista de las matemáticas, el filtro es en realidad un arreglo de valores, que corresponden a los **pesos** o **parámetros** de la red neuronal. Al pasar el filtro por la imagen, lo que en realidad pasa es que se calcula el producto punto entre los valores del filtro y los valores correspondientes de la imagen.

El resultado de esta operación es otro arreglo de 28x28 (asumiendo que el filtro se mueve un pixel a la vez) de valores que muestran la similitud entre cada posible posición del filtro (que encierra valores correspondientes a la imagen) y el filtro en si. Es por este proceso que se denominan convolucionales, pues el filtro "convoluciona" alrededor de toda la imagen.

A este nuevo arreglo se lo denomina **mapa de activación o identificador de características.** Si se quisiera detectar más características de la imagen es necesario incrementar el número de filtros. La Figura 4 muestra cómo una imagen es procesada por varias capas convolucionales generando mapas de activación cada vez más pequeños.

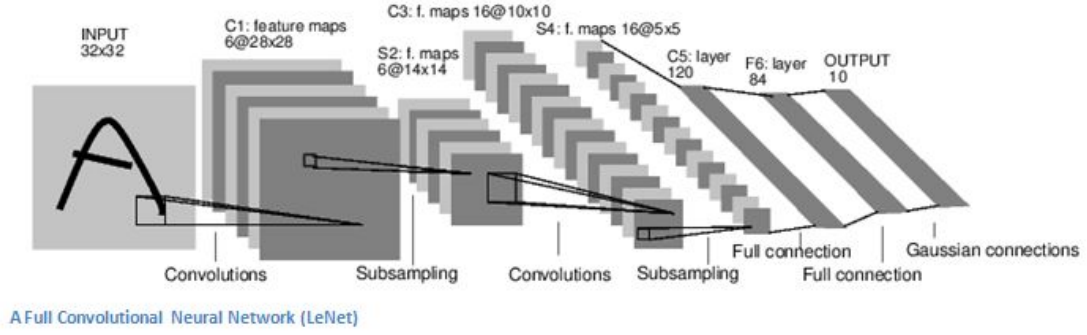


Figura 4. Funcionamiento de un grupo de capas convolucionales.

Tomado de: D. Adit, 2016

Hay que recordar que el principio de funcionamiento de las redes neuronales es la abstracción de características complejas partiendo de características simples. Una Red Neuronal Convolucional puede contar con muchas capas

convolucionales, lo que le permite obtener características cada vez más complejas. Por ejemplo, la primera capa puede ser la encargada de encontrar líneas rectas o curvas. El mapa de activación que ésta genere es utilizada por la segunda para reconocer una silueta que luego se convertirá en una rueda, lo que podría indicar que el objeto es una bicicleta, etc… El ejemplo anterior es bastante simple comparado con lo que en realidad pasa aunque el principio el es mismo.

Otros dos parámetros importantes dentro de este proceso son:

1. El *stride,* que se refiere a cuánto se mueve el filtro hasta alcanzar la siguiente posición sobre la imagen. En el ejemplo anterior se utiliza un *stride* de 1. Nótese que entre mayor sea el *stride,* más pequeño el mapa de activación resultante.
2. El *padding.* Teniendo en cuenta que cada mapa de activación es de menor tamaño que la imagen o mapa de activación del cual se obtiene, puede darse el caso en el que la velocidad a la que estas dimensiones se reducen resulten poco convenientes. Para evitar esto, se puede agregar un "borde" de valores nulos alrededor de la fuente de información para que al terminar el proceso se obtengan las dimensiones adecuadas. El ancho de este borde se denomina *padding.*

2.1.4.2 Otras capas de una CNN

Además de las capas convolucionales que le dan el nombre a este tipo de redes neuronales, existen capas intermedias y finales que controlan la linealidad del modelo y proveen un resultado utilizando el mapa de activación resultante de las convoluciones.

Capa ReLU

Las capas ReLU (Rectified Linear Units) se ubican entre capas convolucionales con el fin de remover la linealidad de los mapas de activación proporcionados por las mismas. Estas simplemente aplican la función f(x) = max(0,x) a los resultados.

El uso de este tipo de capas ha probado contribuir con la rapidez de aprendizaje de la red (en comparación con otras funciones como *sigmoid*) debido a su sencillez a la hora de calcular. Además, reduce el problema del **gradiente desvaneciente**.

Capa Dropout

Las capas de dropout se encargan de descartar aleatoriamente resultados obtenidos por la capa anterior a esta. Esto previene el problema de overfitting pues los resultados no dependen completamente de los datos que se le entregan a la red durante el entrenamiento.

Capa *Fully Connected*

Este tipo de capas forman el final de la red convolucional. Varias capas Fully Connected (Completamente conectadas) son las encargadas de tomar los mapas de activación producto de las capas convolucionales anteriores y decidir a qué clase pertenecen. La Figura 5 muestra cómo se acoplan al resto de la arquitectura de una red neuronal convolucional.

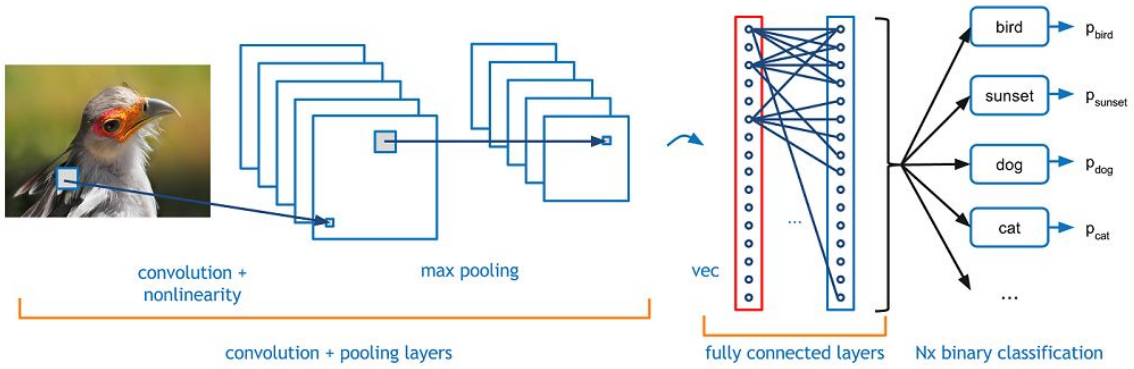


Figura 5. Arquitectura de una CNN.

Tomado de: D. Adit, 2016

2.1.4.3 Transferencia de aprendizaje

Muchas de las redes entrenadas utilizando imágenes muestran un curioso fenómeno: las características aprendidas por las capas inferiores de diferentes modelos muestran gran similitud sin importar que los datos con los que se les entrene sean diferentes. Parecería, por otro lado, que son generales en el sentido de que se pueden aplicar para diferentes tareas. (Yosinski et al, 2014).

Sin embargo, entre más arriba se encuentra una capa, mayor es la especialización de la misma con respecto al caso para el que se le haya entrenado. Por ello, no es factible transferir su conocimiento.

Lo ideal entonces es utilizar el entrenamiento de las capas inferiores y entrenar las capas superiores con el set de datos específico para el dominio de problema a resolver.

Esta característica resulta extremadamente útil a la hora de entrenar una red neuronal convolucional, pues reduce enormemente el tamaño del set de datos necesario para conseguir un modelo que solucione X problema.

2.2 IaaS y PaaS

Las siglas IaaS (Infraestructura como servicio) y PaaS(Plataforma como servicio) se han vuelto muy famosos en los últimos años. La idea principal de estos servicios es liberar a sus usuarios de la necesidad de comprar y mantener un servidor. Ahora, por el valor de una suscripción, un desarrollador puede adquirir poder de procesamiento en la nube para implementar sus aplicaciones sin preocuparse por ninguno de los temas administrativos propios de un servidor propietario (mantenimiento, disponibilidad, reparaciones, etc...)

Con la reducción en los costos de hardware e incremento en sus capacidades, compañías como Heroku, Google o Amazon han podido sacar al mercado soluciones como estas a precio realmente convenientes para cualquier emprendedor. Cuentan además con escalabilidad integrada, por lo cual, cada quien paga dependiendo de cuánto poder de procesamiento necesite.

2.3 Python

Python es un lenguaje de programación creado por Guido van Rossum en 1991. Es un lenguaje de “scripting”, lo que significa que los programas escritos en Python no necesitan compilarse para ejecutarse. Además, gracias a que obliga al programador a separar bloques de código utilizando espacios en blanco (identación) es fácil de entender cuando se lee.

Python se puede usar para una gran variedad de proyectos y gracias a que es un proyecto de código libre es utilizado extensamente alrededor del mundo.

2.3.1 Utilidad en el campo de la investigación

Python cuenta con muchas herramientas que facilitan el manejo de datos. Por ejemplo, *numpy*, una de las librerías que incluye el paquete original, permite,

entre otras cosas, realizar operaciones sobre matrices directamente, lo cual es un poco más complicado en otros lenguajes.

Por otro lado, proyectos como Jupyter permiten crear un ambiente interactivo para programar, al cual llaman “cuaderno” (notebook en inglés). Este cuaderno permite almacenar código para ejecutar, comentarios, datos y cualquier otro tipo de recurso pertinente para el proyecto en el que se trabaje.

2.3.2 Keras

Keras es una librería escrita en Python que provee una API de alto nivel para construir redes neuronales. Esta funciona encima de otras librerías como TensorFlow o Theano, que también pueden ser utilizadas para construir una red neuronal, pero con mucho más trabajo. El objetivo de Keras es proporcionar a los programadores herramientas que les permitan crear prototipos de redes neuronales lo más rápido posible.

Es una librería que se enfoca en la modularidad, pues cada componente de la red neuronal puede ser reemplazado fácilmente. Si una de las capas de nuestro sistema no funciona como se espera, se puede reemplazar por otra cambiando un par de líneas en el código.

Además, es fácil de extender, es decir, el programador no está limitado a las herramientas propias de la librería, sino que puede crear sus propios módulos dependiendo de sus necesidades.

2.4 Docker

Docker es una plataforma de código abierto pensada para automatizar la creación y despliegue de ambientes virtuales. Provee todos los beneficios de crear máquinas virtuales pero de una manera eficiente, ahorrando recursos de memoria y procesamiento.

2.5 SBC - Single Board Computer

Un SBC es una computadora cuyos componentes pueden entrar en una sola tarjeta. Su tamaño, costo y capacidad las hacen atractivas para cualquier proyecto electrónico.

La mayoría de SBCs pueden correr sistemas operativos complejos como Linux y cuentan con puertos digitales que permiten el control de periféricos.

3. DISEÑO

3.1 Consideraciones Iniciales

Se desea desarrollar un sistema mediante el cual los usuarios puedan entregar envases reciclables con el fin de obtener una recompensa. El sistema debe poder reconocer de entre cinco tipos de envases diferentes y guardar esa información sobre el tipo de envase y la recompensa otorgada al usuario. Además, el usuario debe poder consultar las recompensas que ha acumulado.

Cinco tipos obviamente no cubre todas las clases de envases que se desearía reconocer finalmente, así que se debe poder agregar más clases o tipos de envases reconocibles fácilmente. Para ello podría utilizar la información de los envases que no se reconocieron.

Los envases son principalmente botellas PET. Para el desarrollo de este prototipo se propuso agregar un envase extra que no sea una botella con el fin de demostrar la adaptabilidad del sistema a otros tipos de envases.

Entre los tipos iniciales de envases a reconocer se encuentran los siguientes productos:

1. Sprite de 400ml
2. Güitig de 500ml
3. Dasani de 600ml
4. Pepsi de 400ml
5. Lata de 355ml de cualquier producto

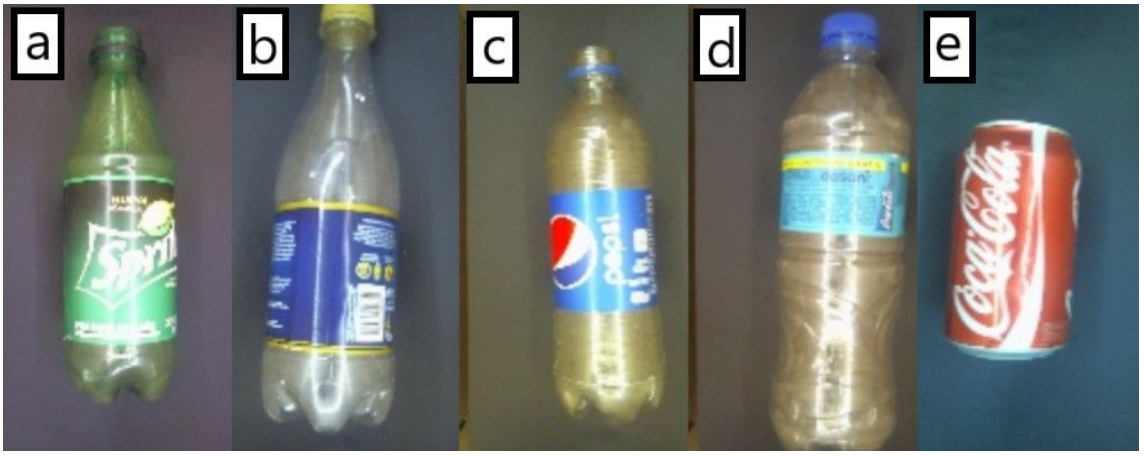


Figura 6. Tipos de botellas a reconocer.

a) Sprite de 400ml

b) Güitig de 500ml

c) Dasani de 600ml

d) Pepsi de 400ml

e) Lata de 355ml de cualquier producto

3.2 Reconocimiento de la botella

La selección del método de reconocimiento de la botella pasó por varias iteraciones que se describen a continuación.

3.2.1 Reconocimiento mediante sensores de distancia

El primer acercamiento intentó utilizar la forma del envase como su principal fuente de información. Para obtener esta información se ubicarían varios sensores de distancia alrededor de la botella para obtener las medidas de la botella en altura y diámetro en varios puntos.

Mientras que la forma de la botella resulta ser un diferenciador clave, recolectar esa información mediante sensores de distancia traía los siguientes inconvenientes:

1. La precisión de los sensores no era la indicada. Los sensores ultrasónicos, por ejemplo, cuentan con una precisión de 3 milímetros que en casos específicos no es suficiente para diferenciar una botella de otra.
2. La distancia de detección de los sensores. Dentro de la cámara de detección el envase puede estar demasiado cerca del sensor. Lo que lo hace inservible.
3. La dificultad de obtener los sensores para el prototipo.
4. Interferencia entre sensores dentro de la cámara.

3.2.2 Reconocimiento mediante el análisis del perfil

Ya que la forma del envase sigue siendo un factor importante para el reconocimiento del mismo se propuso capturar una imagen, para luego, mediante el procesamiento de la misma, extraer el perfil correspondiente al envase. Este perfil luego se entrega a una Red Neuronal Convolucional que se encarga de analizar la información del perfil y determinar una clase.

Esta iteración tiene éxito solucionando los problemas relacionados con los sensores de distancia. Además, el procesamiento necesario para obtener el perfil del envase es bajo.

Por otro lado, la iluminación de la cámara de reconocimiento juega un papel crucial en el perfil que se obtiene, por lo que en ciertos casos el perfil de un envase tiene una forma relativamente similar al de otra clase. Además, es sensible a deformaciones en el envase.

Otro factor a tomar en cuenta es la cantidad de procesamiento necesario para correr la red neuronal. Hay que tener en cuenta que el dispositivo que funciona como interfaz para el usuario cuenta con limitados recursos en cuanto a procesamiento y memoria RAM.

Por último, al procesar la imagen para obtener únicamente el perfil del envase, se desprecian otras fuentes de información que pueden ser útiles para reconocer el tipo de envase, como su color.

3.2.3 Reconocimiento por análisis visual completo

El último y final acercamiento utiliza también una red neuronal convolucional para reconocer el envase, pero a diferencia de su predecesor, no realiza ningún procesamiento sobre la imagen original (a excepción de un redimensionamiento).

De esta manera se aprovecha completamente las capacidades de una red neuronal convolucional. A pesar de que también introduce otros inconvenientes, la precisión y flexibilidad que brinda hacen que sea viable.

Entre los inconvenientes antes mencionados se destacan:

1. Las necesidades de capacidad de procesamiento requeridos por la nueva red neuronal.
2. La cantidad de datos necesarios para entrenar una red de este tipo.

Se decidió resolver el primer punto utilizando un servidor de reconocimiento. Al principio puede parecer que implementar un servidor para realizar esta tarea es un esfuerzo exagerado. Hay varias razones que refutan esta idea:

1. El bajo costo y facilidad de despliegue. El mercado actual de Infraestructuras como servicio *(IaaS)* se ha vuelto extremadamente competitivo en la actualidad. Esto permite no solo desplegar servidores de cualquier clase a bajos costos, sino también con gran facilidad. Proveedores como Google y Amazon hacen que levantar un servidor sea cuestión de un par de clicks.
2. Las posibilidades que se abren con un servidor en línea. Entre las primeras, se puede listar el hecho de que el usuario puede acceder a

sus recompensas en tiempo real. Además, se pueden recolectar datos que sean útiles en otros aspectos del negocio. Una discusión más extensa de este tema en el Capítulo 6.

3. La escalabilidad del sistema. Mientras más clases se agreguen al modelo, más pesado se vuelve. El servidor está listo para soportar esta carga y en el caso de necesitarlo, desplegar más capacidad de procesamiento es sumamente simple. Eso de otra manera supondría que el hardware del prototipo se tenga que actualizar constantemente o que se deba invertir en un sistema mucho más costoso. Dividiendo el trabajo de esta manera, las interfaces del usuario pueden enfocarse en ser lo más baratas posibles, pues lo único que tienen que hacer es tomar una foto y enviarla al servidor para que sea reconocida.

Otros argumentos en contra de esta solución pueden ser los siguientes:

1. Si la interfaz no tiene internet, no funciona el sistema: Hay que tener en cuenta que las interfaces están pensadas para instalarse en zonas urbanas, en las cuales, según datos del INEC (2016), el porcentaje de hogares que cuentan con internet es del 44.6%. Esto apunta a que el acceso al internet desde la interfaz no propone un impedimento real. En una última instancia, las interfaces pueden ser adaptadas para utilizar la red celular.
2. Al tener que realizar un requerimiento a través de internet, el tiempo para reconocer una botella incrementa. Si bien es cierto que el tiempo de reconocimiento incrementa, el tiempo total en el que se reconoce el envase sigue siendo bajo. Más información en el capítulo 5 (Resultados)

El segundo problema a resolver, la cantidad de datos para entrenar una red de este tipo, se sustenta utilizando la técnica descrita por François Chollet (2016) en su publicación “Construyendo modelos de reconocimiento de imágenes con muy pocos datos”. En este artículo, se plantea utilizar los pesos de las capas convolucionales de un modelo de reconocimiento visual ya entrenado (VGG16

entrenado con el set de datos *Imageset)* para luego ser adaptado a una serie de capas completamente conectadas (*Fully connected layers*) entrenadas con nuestros propios datos con el fin de conseguir una alta precisión.

Imageset cuenta con miles de imágenes que, mediante el entrenamiento, dotan al modelo de la arquitectura VGG16 con la capacidad de transformar una imagen en un grupo de valores que representan características de alto nivel. A este grupo de valores se los denomina Mapa de activación, el cual funciona como entrada para las capas completamente conectadas que se agregan al final.

En resumen, a pesar de que la última solución presenta retos mayores a la hora de la implementación, los resultados muestran que los beneficios exceden a las inconveniencias.

3.3 Servidor de reconocimiento

El servidor de reconocimiento se escribe en Python para poder utilizar la librería Keras, la cual facilita la implementación de la red neuronal. El uso de otras librerías encargadas de la comunicación HTTP con el cliente y la base de datos se detallan en la sección 4.1. El propósitio principal de este servidor es el de utilizar la imagen que el prototipo envía para determinar la clase a la que pertenece el envase. La Figura 7 muestra el funcionamiento general del sistema.

Este servidor se levantará en Google Compute Engine. Se seleccionó este proveedor pues cuenta con herramientas que facilitan el despliegue de un servidor de este tipo. Entre ellas se cuenta el repositorio de imágenes de Docker y la consola de administración y despliegue.

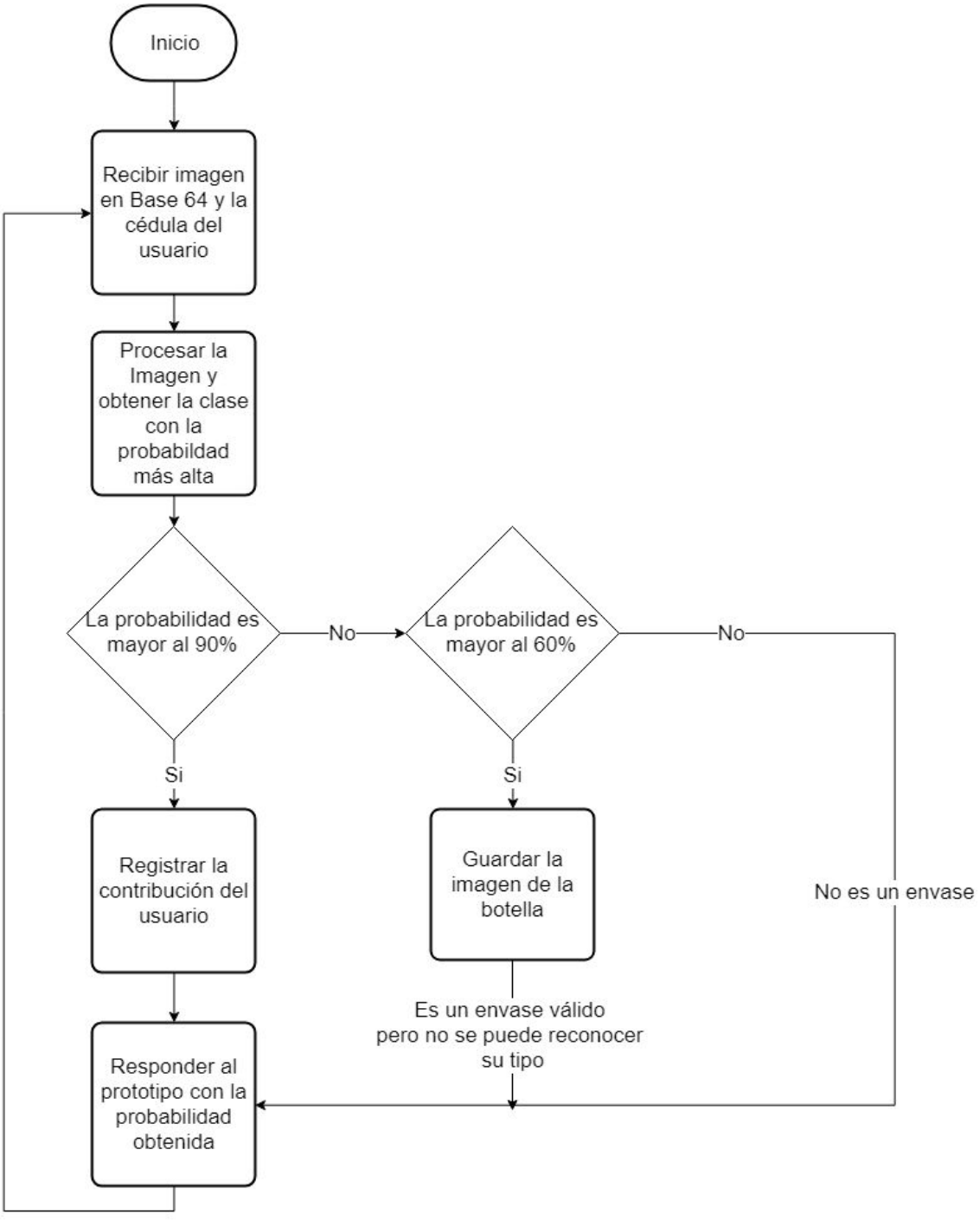


Figura 7. Diagrama de flujo del servidor de reconocimiento.

3.4 Base de datos

Los datos que recoja el servidor de reconocimiento se deben almacenar en algún lugar. Para esto se decidió utilizar una base de datos de MongoDB. Este tipo de base de datos tiene un excelente desempeño en grandes volúmenes de registros. Además, al no tener un esquema definido, los registros de la base pueden aceptar nuevos campos que se crean necesarios después de su implementación sin causar problemas.

mLab proporciona instancias de MongoDB como un servicio. Las bases de datos de prueba son gratuitas y para desplegar una instancia se necesita solamente una cuenta.

3.5 Servidor de consulta

Ya que los datos sobre las recompensas que consiga el usuario se van a almacenar en una base de datos en línea, es preciso implementar una solución mediante la cual el usuario pueda consultar sus ganancias. El funcionamiento general de este servidor se detalla en la Figura 8.

Para construir esta plataforma se decidió utilizar NodeJs y Heroku. NodeJs es liviano y eficiente. El hecho de que sea Javascript lo hace ideal para cualquier aplicación que tenga que ver con web.

Una de las librerías más utilizadas de NodeJs es Express, la cual le permite a Node construir un servidor HTTP fácilmente. Para la comunicación con la base de datos se utiliza Node Mongo, la librería oficial de Node para la comunicación con una base de datos MongoDB.

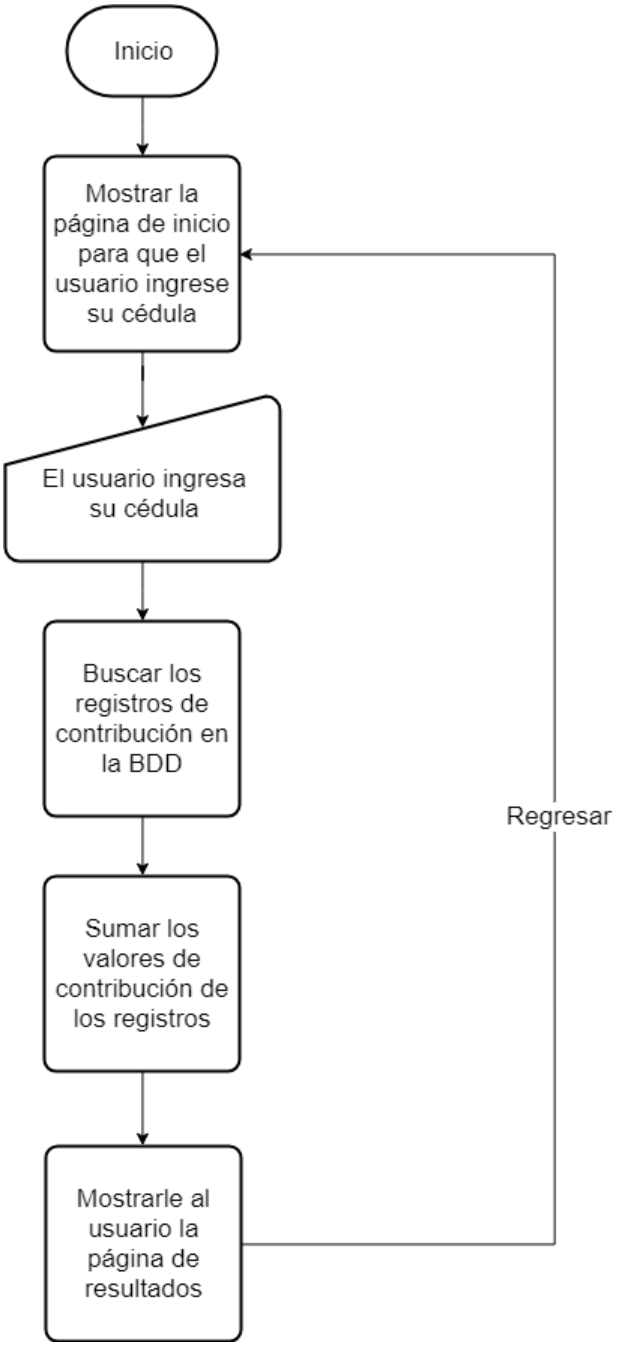


Figura 8. Diagrama de flujo del servidor de consultas.

Del lado del servidor, se utilizó Heroku debido a que tiene herramientas que aceleran los procesos de desarrollo y actualización del servidor. A diferencia del servidor de reconocimiento, Heroku no precisa utilizar Docker, pues sus

instancias virtuales están configuradas para alojar aplicaciones web. Para actualizar el sistema por ejemplo, solo se debe actualizar el repositorio en línea del código. Heroku, mediante las configuraciones que se le proporcione, se encarga del resto.

3.6 Diseño del prototipo

Este prototipo es el que le permite al usuario entregar sus envases. Cuenta con una cámara de reconocimiento. Es dentro de esta cámara que se toman las fotos para enviarse al servidor. También cuenta con una pantalla táctil que le permite al usuario ingresar su número de cédula y dar la orden para que se escanee el envase que se encuentre presente dentro de la cámara de reconocimiento. El número de cédula le permitirá luego al usuario revisar los beneficios obtenidos por entregar los envases en el servidor de consulta.

Una vez se obtenga una respuesta del servidor, el prototipo toma la decisión de almacenar o desechar la botella. Esto lo hace mediante un servomotor que hace girar la plataforma en la que se dispone el envase. El funcionamiento general del programa que controla el prototipo se detalla en la Figura 9.

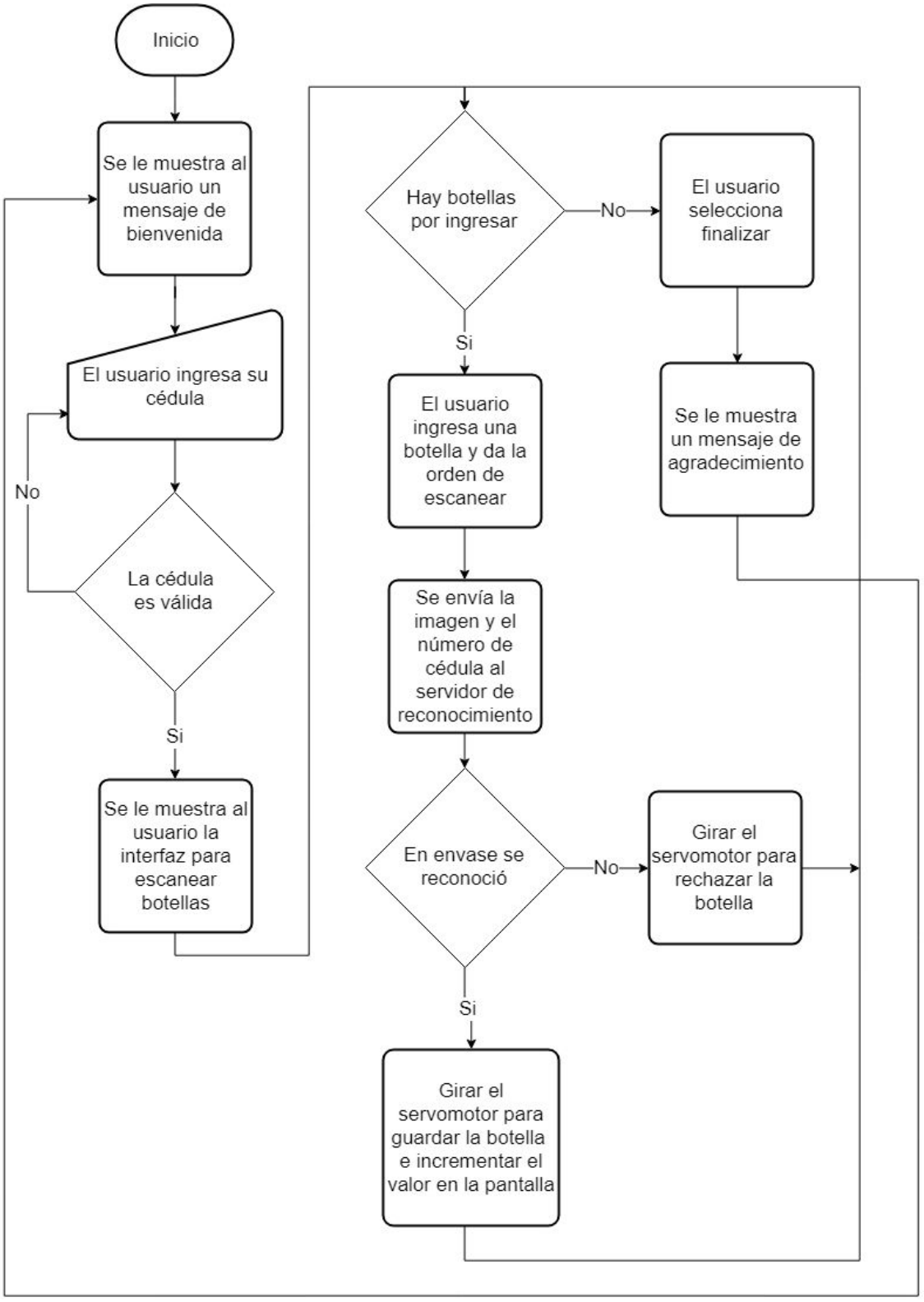


Figura 9. Diagrama de flujo del programa que controla la interfaz.

3.6.1 Selección del Sistema embebido

Para el sistema embebido responsable de controlar las funciones del prototipo se decidió utilizar SBCs. Esto debido a que brindan varias facilidades a la hora de implementar el prototipo mientras que su costo se mantiene bajo. Las funciones que debe cumplir se listan a continuación:

1. Control de un servomotor mediante PWM
2. Control de una pantalla táctil
3. Comunicación HTTP con el servidor
4. Captura de imagen

Para el presente proyecto, se tomaron en cuenta las siguientes variables a la hora de seleccionar el dispositivo:

- Memoria RAM. El microcontrolador debe tener suficiente memoria como para correr el programa de control de la interfaz.
- Capacidad de procesamiento.
- Periféricos y puertos. El microcontrolador debe poseer preferiblemente un puerto USB que permita conectar la cámara encargada de capturar las fotos y un puerto PWM que permita controlar el servomotor
- Precio. Uno de los objetivos del proyecto es reducir en lo más posible los costos de construcción del prototipo.
- Características extras. Algunos de los candidatos cuentan con características que las hacen realmente interesantes para nuestro caso específico.

Entre las opciones disponibles en el mercado se listan las siguientes:

Raspberry Pi

El Raspberry Pi es una computadora compacta, de bajo costo y alto rendimiento desarrollada en el Reino Unido por la Fundación Raspberry Pi. Ya

en su sexta iteración (que entregó el Raspberry Pi 3), la misma fundación continúa mejorando un producto que ya ha calado su espacio dentro del mundo de los electrónicos y de la programación.

Desde su creación, varias compañías han ido desarrollando periféricos que permitan extender las capacidades del Pi. Desde pantallas táctiles hasta cámaras y monturas, las posibilidades de extender la funcionalidad de la minicomputadora es realmente extensa.

El candidato a tomar en cuenta de esta marca es el Raspberry Pi Zero W que cuenta con las siguientes características:

- 802.11 b/g/n wireless LAN
- Bluetooth 4.1
- Bluetooth Low Energy (BLE)
- 1GHz, single-core CPU
- 512MB RAM
- Puertos Mini HDMI y USB On-The-Go
- Puerto Micro USB para alimentación
- Header de 40 pines compatible con HAT

El costo de este modelo es de 10$. Sin embargo, al solamente contar con un puerto Mini USB es necesario comprar un adaptador extra que permita tener suficientes puertos USB (10$ extra).

Esta familia de minicomputadoras no solo cuentan con una comunidad extensa de desarrolladores dedicados a crear guías para quienes deseen aprender a utilizarla, sino que también con una larga lista de distribuidores de accesorios para el mismo.

Arduino

Arduino es la popular plataforma open-source para prototipos electrónicos basado en un hardware y software fáciles de utilizar. Está pensado para

artistas, diseñadores, aficionados y todo quien esté interesado en crear dispositivos interactivos. Se enfoca en brindar soluciones para la mayor cantidad de proyectos posibles.

El modelo de Arduino a tomar en cuenta para nuestro análisis es el UNO R3, que cuenta con las siguientes características:

- Microcontrolador ATmega 328 @ 16Mhz
- Memoria flash de 32Kb
- 20 Pines I/O digitales
- 2Kb de SRAM
- Puerto USB
- El costo de uno de estas tarjetas es de 25$.

Estos dispositivos facilitan el control de sensores y de salidas digitales. Es por ello que sus aplicaciones normalmente no son realmente complicadas.

Omega 2

El Omega2 de Onion es la competencia directa de Raspberry Pi en cuanto a precios y extensibilidad. Su enfoque principal es el de ser utilizado en proyectos de hardware conectados con internet. Sus desarrolladores lo describen como la combinación del eficiente consumo y lo compacto de Arduino junto con el poder y la flexibilidad de un Raspberry Pi. Su tamaño es realmente impresionante pues no es mucho más grande que una cereza.

El Omega2 corre una distribución de Linux basada en OpenWrt. Además, cuenta con una selección de aplicaciones que pueden ser descargadas de su tienda en línea, Onion App Store.

A diferencia del Raspberry Pi, que está desarrollado alrededor de Python, puede ser programado con múltiples lenguajes de programación, incluyendo Ruby, NodeJS e incluso PHP.

Omega2, que es el modelo que se tomó en cuenta, tiene las siguientes características:

- CPU @ 580Mhz
- Memoria RAM de 64Mb
- 16Mb de memoria integrada
- Puerto USB 2.0
- 802.11 b/g/n wireless LAN
- 15 puertos GPIO

Este dispositivo es modular, es decir, hay varios componentes extras que se pueden instalar fácilmente sobre la tarjeta para brindar funcionalidad extendida. Estos pueden incluir conectividad Bluetooth o GPS por ejemplo.

NanoPi

La familia de tarjetas NanoPi son, al igual que Onion, un competidor directo de Raspberry Pi. Son desarrolladas y producidas por Friendly ARM. Tienen un tamaño reducido en comparación al Raspberry Pi. Es de código libre y puede correr Ubuntu MATE, Debian y hasta algunas versiones de Android.

El modelo que se tomó en cuenta es el NanoPi M1 H3, que cuenta con las siguientes características:

- CPU: Allwinner H3, Quad-core Cortex-A7@1.2GHz
- GPU: Mali400MP2@600MHz, Supports OpenGL ES2.0
- DDR3 RAM: 512MB
- Conectividad: 10/100M Ethernet
- Audio: 3.5mm audio jack/Via HDMI
- Micrófono sobre la tarjeta
- Receptor Infrarojo
- 3 Puertos USB 2.0
- Ranura MicroSD

- Puerto MicroUSB para transmisión de datos y energía.
- Salida de video: HDMI 1.4 1080P, CVBS
- 40 Pines GPIO compatibles con Raspberry
- Tamaño : 64 x 50mm
- Alimentación: DC 5V/2A
- El precio de este es de 15$.

Al igual que las placas anteriores, el NanoPi tiene la capacidad de extender sus funcionalidades mediante periféricos fáciles de instalar. Sin embargo, los accesorios para esta placa resultan ser de menor costo y se adaptan a las necesidades del proyecto.

Compración

A continuación se muestra una tabla que resume los pros y contras de cada una de las opciones:

Tabla 2.

Comparativa entre sistemas embebidos (primera parte)

Modelo	**Ventajas**	**Desventajas**
Raspberry Pi Zero W	● Gran capacidad de procesamiento. ● Corre una distribución de Linux ● Facilidad para integrar periféricos ● Cuenta con una comunidad que ayuda a los desarrolladores ● Ambiente de desarrollo fácil de implementar ● Facilidad para integrar con una pantalla táctil. ● Facilidad para adquirirlo en el mercado. ● Cuenta con conectividad a Wifi Integrada	● Cuenta solamente con un puerto USB mini. ● Accesorios relativamente caros. ● Cierta dificultad manejando los puertos de entrada y salida.
Arduino UNO R3	● Eficiencia energética. ● No necesita de un sistema operativo para funcionar ● Eficiente manejando puertos digitales de entrada y salida. ● Facilidad para adquirirlo en el mercado. ● Cuenta con una comunidad que ayuda a los desarrolladores.	● La programación para nuestro caso se vuelve más extensiva. ● Su capacidad de procesamiento cae muy por debajo de sus competidores

Tabla 3.

Comparativa entre sistemas embebidos (segunda parte)

Modelo	**Ventajas**	**Desventajas**
Onion Omega2	● Gran capacidad de procesamiento. ● Puede correr varias distribuciones de Linux ● Tamaño reducido ● Facilidad para integrar periféricos. ● Cuenta con conectividad a Wifi Integrada. ● Bajo precio. ● Trabaja con varios lenguajes de programación. ● Cuenta con soporte para una tienda de aplicaciones en ● línea.	● Cuenta solamente con 15 Pines GPIO. ● No tiene un puerto dedicado que permita conectar una pantalla. ● Es difícil de conseguir en el mercado pues no hace mucho tiempo que se lanzó.
NanoPi M1 H3	● Gran capacidad de procesamiento. ● Cuenta con varios puertos USB. ● Se puede conectar una pantalla táctil fácilmente. ● Tamaño reducido. ● Periféricos de bajo costo. ● Ambiente de desarrollo de fácil implementación. ● Es de código abierto. ● Corre varias distribuciones de Linux, incluso Android. ● Fácil de adquirir en el mercado.	● No cuenta con tanto soporte en el internet como sus competidores. ● No cuenta con conectividad Wifi integrada.

Después de realizar el análisis respectivo podemos concluir que el mejor dispositivo para nuestro proyecto es el NanoPi M1 H3 por las siguientes razones:

- Cuenta con los puertos USB y pines GPIO necesarios para el prototipo.
- El hecho de que corra una distribución de Linux elimina la necesidad de implementar nuestra propia base de datos, lo que en general mejora la calidad del proyecto.
- Su costo total, incluido periféricos, es el menor de todos.
- Las pantallas táctiles que se producen para este dispositivo son fáciles de instalar y cuentan con controladores ya instalados en la placa.

3.6.2 Periféricos

En la lista de los periféricos necesarios para construir el prototipo de la interfaz se encuentran:

1. Una cámara USB. La calidad de la imagen no es de especial interés pues las fotos a tomar no tienen alta resolución, así que una cámara VGA basta. Lo que sí es importante es el ángulo de apertura de la cámara, es decir, qué tan ancha es la imagen que puede capturar. Encima de 70 grados de apertura, a una altura de 30cm, se puede capturar un ancho aproximado de 42 centímetros, suficiente para encajar cualquiera de los envases propuestos.
2. Un servomotor. Se decidió utilizar un servomotor de 15kg/cm, suficiente para soportar cualquiera de los envases incluso cuando se encuentren llenos, teniendo en cuenta que la plataforma tiene 7,5cm de ancho a cada lado.
3. Iluminación para la cámara de reconocimiento. Para solventar esta necesidad se utilizó un arreglo de leds blancos adheridos al rededor de las paredes del prototipo.
4. Una fuente de poder de 5V 3A con una salida microUSB para proveer energía al microcontrolador.
5. Una segunda fuente de poder de 5V 3A para alimentar el servomotor.

La Figura 10 muestra el diagrama de bloques para las conexiones entre el SCB y sus periféricos.

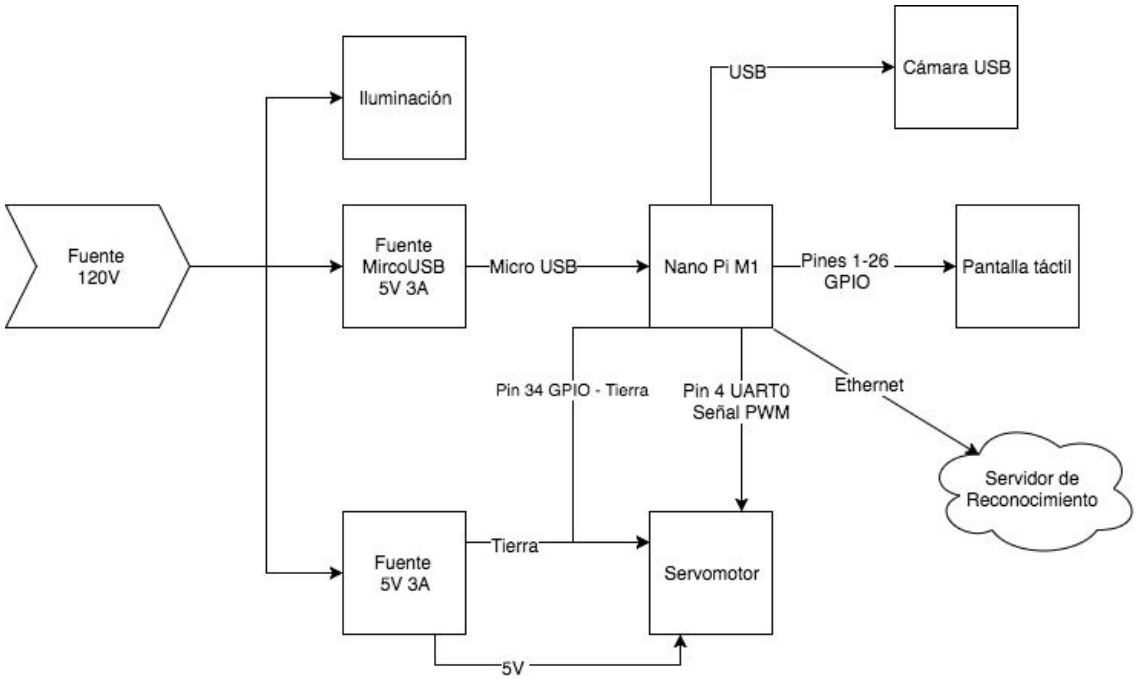


Figura 10. Diagrama de bloques del sistema.

3.6.3 Armazón

Todo el prototipo se monta sobre un cuerpo de MDF (*Medium Density fibreboard*). Se seleccionó este material por su facilidad para ser cortado con láser.

Otra parte importante del sistema son los rodamientos y acoples que le permiten al servomotor girar la plataforma en la que se disponen los envases para aceptarlos o rechazarlos. La mayoría de estos componentes fueron adaptaciones de elementos varios para producir un prototipo funcional.

3.7 Integración de servidores y prototipo

En esta sección se explica más a fondo la manera en la que interactúan todos los componentes del sistema para brindar el servicio al usuario.

El proceso inicia cuando el usuario recibe el mensaje de bienvenida en el prototipo e ingresa su número de cédula. A continuación, el prototipo le muestra la interfaz para entregar los envases. Cuando el usuario le da la orden al prototipo de escanear el envase, este le toma una foto al envase y la envía al servidor de reconocimiento en un formato Base64 junto con el número de cédula ingresado. Una vez en el servidor de reconocimiento, el modelo de predicción se encarga de determinar la clase a la que pertenece la botella de la imagen. Dependiendo de este resultado, se escriben los registros adecuados en la base de datos. Si el resultado fue favorable, se acredita ese saldo al usuario. En caso de no ser favorable, el sistema decide si guardar esa imagen para un futuro procesamiento o simplemente desechar dicha imagen. Después de analizar los resultados, el servidor le responde al prototipo indicando si el envase se reconoció o no. Ya con este resultado, el prototipo puede tomar la decisión de devolver o guardar el envase.

Por otro lado, cuando el usuario desea consultar sus ganancias, debe acudir al servidor de consulta, donde puede ingresar su número de cédula para revisar los beneficios acumulados.

La Figura 11 muestra un resumen gráfico del procedimiento que se desarrolla entre el cliente y el servidor de reconocimiento.

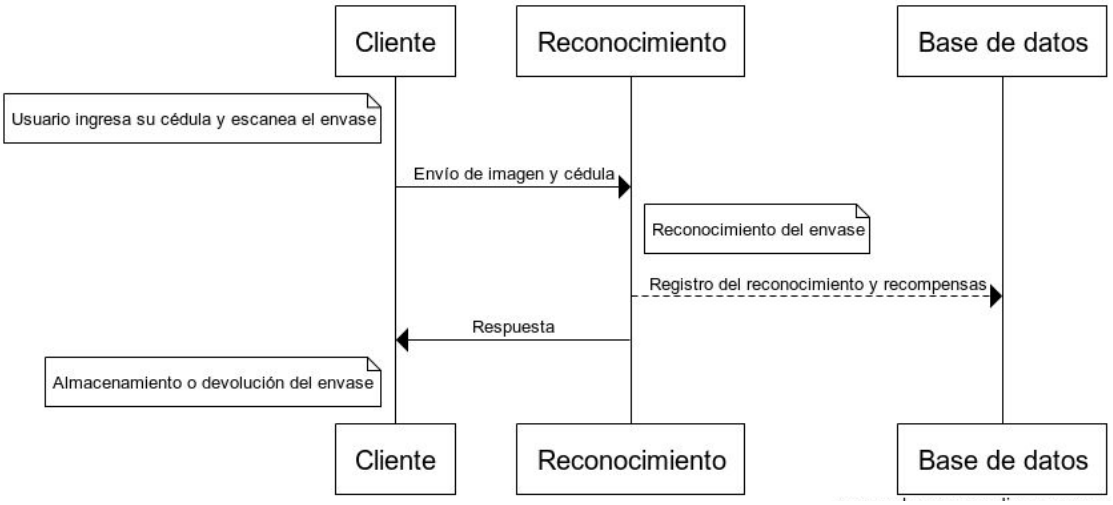


Figura 11. Reconocimiento del envase.

La Figura 12 resume por otro lado el proceso que se desarrolla entre el usuario y el servidor de consultas.

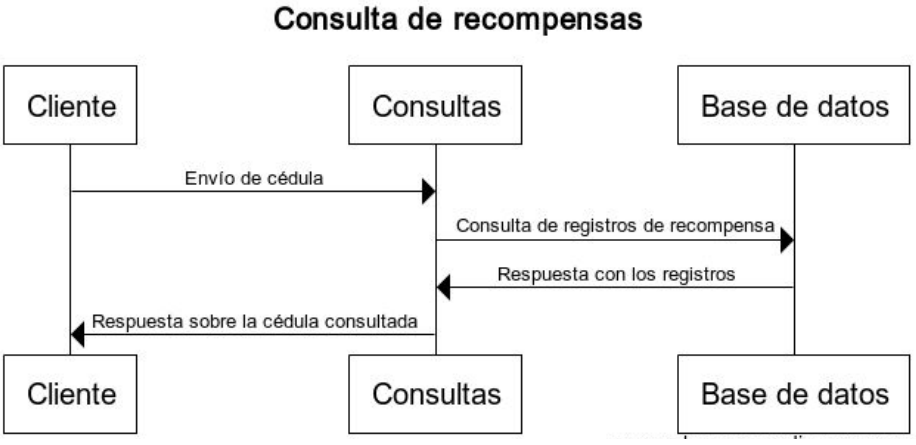


Figura 12. Consulta de recompensas

4. DESARROLLO

4.1 Construcción de la Red neuronal.

Esta sección está dedicada a explicar el proceso necesario para obtener el modelo de predicción que utiliza el servidor.

Antes de empezar a entrenar el modelo es necesario obtener el set de datos que se va a utilizar. Para ello, dentro del prototipo de interfaz se instaló la cámara web a utilizar y se tomaron 262 fotos por cada clase de envase. 200 de ellas servirán para entrenar el modelo mientras que las 62 restantes para validación.

Lo primero es partir de la arquitectura VGG16 entrenada bajo el set de datos Imageset. En la Figura 13 se presenta a la red completa con todas sus capas.

La librería que se utiliza para desarrollar el modelo, Keras, tiene utilidades que permiten importar fácilmente el modelo ya entrenado bajo Imageset. La misma librería permite además obtener el modelo sin el último bloque (las últimas cuatro capas), el cual se reemplazará luego por un bloque propio de clasificación.

Otra modificación que se debe hacer a la hora de importar el modelo es cambiar la dimensión de las entradas. Originalmente acepta entradas de dimensiones 224x224x3. Las imágenes del sistema que clasifica envases tienen dimensiones 100x200x3. Por dentro, lo que se realiza la librería Keras es aplicar un padding para completar los pixeles faltantes. Esto no afecta de ninguna manera el resultado.

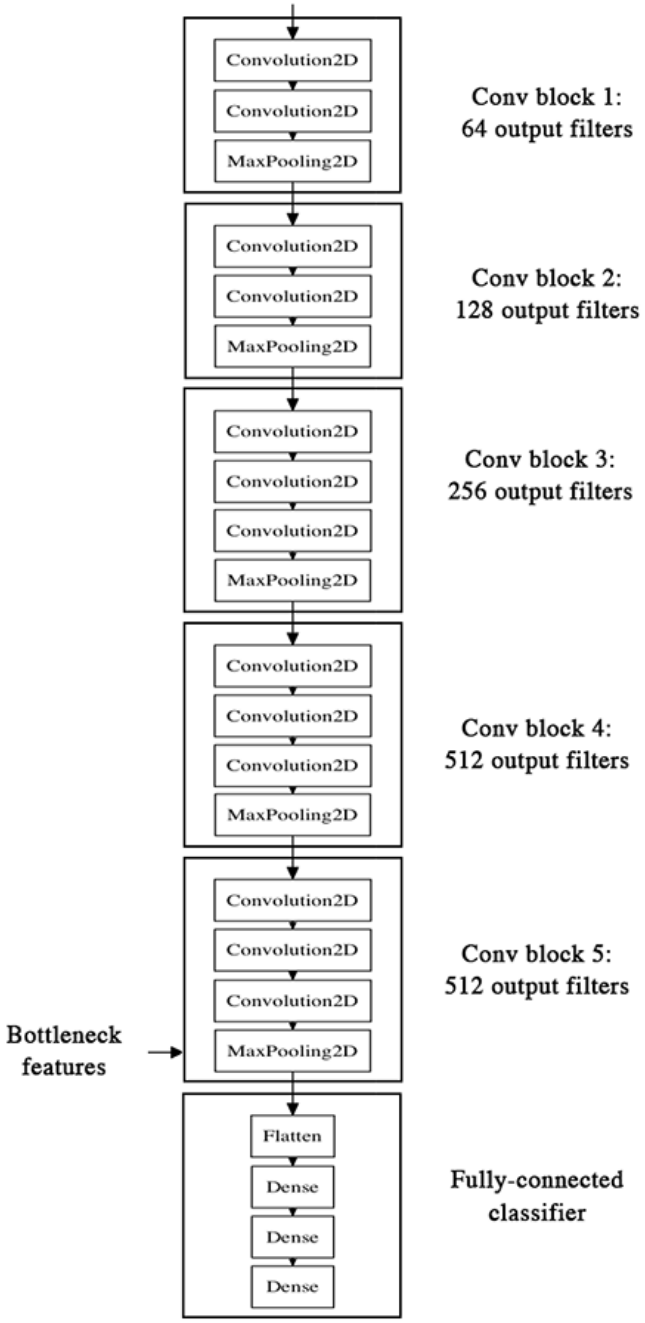


Figura 13. Arquitectura VGG16.

Tomado de: D. Adit, 2016

Los valores RGB las imágenes varían en un rango de 0 a 255. Estos se normalizan dividiendo para 255, lo que hace que varíen en un rango de 0 a 1.

Esto para asegurarse de que el factor de aprendizaje no resulte demasiado alto cuando los valores se acerquen al tope.

Con el modelo cargado, lo primero es realizar varias predicciones utilizando el set de datos de los envases. El proceso anterior nos deja varios mapas de activación que luego serán utilizados para entrenar el clasificador. Este consiste de las siguientes capas:

1. Una capa de aplanamiento (Flatten). Esta se encarga de transformar los mapas de activación en un vector unidimensional que pueda se interpretado por el clasificador.
2. Estas entradas aplanadas pasan a una capa completamente conectada.
3. El resultado para a una capa de *dropout,* la cual elimina aleatoriamente el 50% de los parámetros que ingresan a ella. Esto con el objetivo de evitar linealidad en el modelo además del overfitting.
4. Finalmente estas entradas se conectan con unas últimas 5 neuronas en una capa completamente conectada. Esta capa cuenta con una función de activación softmax, lo que da como resultado un vector de 5 posiciones con las probabilidades de que determinada entrada pertenezca a una clase.

Utilizando entonces el bloque de clasificación y los mapas de activación que se consiguieron anteriormente se procede a entrenar dicho bloque.

Una vez entrenado, el último paso es conectarlo con las capas convolucionales del primer paso y entrenar todo el modelo con el fin de hacer un ajuste final a los pesos del modelo. Ya que no queremos alterar de manera significativa la capacidad de obtener características de bajo nivel de las capas convolucionales, se congelan los pesos de las primeras 14 capas. El optimizador a utilizar es un gradiente descendiente estocástico con un factor de aprendizaje bajo.

El pseudo código a continuación explica el funcionamiento del programa utilizado para entrenar el modelo.

```
Procedimiento ObtenerPrediccionesPreClasificador
    Cargar las Capas convolucionales del Modelo VGG16
    Cargar las imágenes para el entrenamiento
    Obtener las predicciones para las imágenes de
    entrenamiento
    Guardar las predicciones de entrenamiento
    Cargar las imágenes para la validación
    Obtener las predicciones para las imágenes de validación
    Guardar las predicciones de validación
Fin Procedimiento
Procedimiento EntrenarClasificador
    Cargar las predicciones de entrenamiento y generar
    arreglos categóricos a partir de ellos
    Cargar las predicciones de validación y generar arreglos
    categóricos a partir de ellos
    Instanciar el modelo del Clasificador
    Entrenar el modelo del Clasificador y generar una imagen
    del entrenamiento
    Guardar los pesos del Clasificador
Fin Procedimiento
Procedimiento AcoplarClasificador
    Cargar las Capas convolucionales del Modelo VGG16
    Instanciar el modelo del Clasificador
    Cargar los pesos guardados en el Clasificador
    Acoplar el clasificador encima de las capas
    convolucionales
    Bloquear el aprendizaje de las capas convolucionales
```

```
        Compilar el modelo completo con un índice bajo de
        aprendizaje

        Cargar nuevamente las imágenes de entrenamiento y
        validación

        Insertar modificaciones aleatorias en las imágenes de
        entrenamiento

        Entrenar el modelo completo y guardar una imagen con sus
        resultados

        Guardar el modelo final

Fin Procedimiento


ObtenerPrediccionesPreClasificador()

EntrenarClasificador()

AcoplarClasificador()
```

Al final del proceso se obtiene un archivo con extensión **f5** que contiene todos las definiciones y pesos de la red ya entrenada.

4.2 Despliegue del servidor de reconocimiento

Una vez entrenado el modelo para reconocer las imágenes, se puede levantar el servidor de reconocimiento que se encargará de recibir la imágenes y emitir una predicción sobre la clase a la cual cree que pertenece. Mucha de la arquitectura de este servidor fue inspirado por el artículo de Murray (2017).

Se utiliza Docker para crear un contenedor virtual que corre la versión Zesty de Ubuntu. Sobre ella se instala Python 3.6 con todas las dependencias necesarias para correr el aplicativo.

Para la construcción del servidor se utilizaron las siguientes herramientas:

1. Flask. Esta librería permite levantar una API capaz de comunicarse mediante HTTP de manera rápida. Basta solamente con definir los

endpoints (combinaciones de ruta y método) y la función que maneja ese llamado para empezar a atender.

2. PyMongo. Esta es la librería encargada de comunicarse con la base de datos, la misma que se encuentra alojada en los servidores de mLab.
3. Gunicorn. Esta librería permite conectar la aplicación creada en Flask con el exterior. Además permite levantar varios hilos al mismo tiempo, por lo que si ocurre algún error en la aplicación, el usuario no percibe el error.
4. Ngnix. Este servidor HTTP permite exponer finalmente el servicio hacia la web.
5. Supervisord. Este aplicativo permite definir tareas que se ejecutarán en un servidor constantemente mientras las supervisa. Esto automatiza el proceso de inicio de las aplicaciones.

Docker nos permite encapsular esta imagen para ser transportada y desplegarla en cualquier otro ambiente sin tener problemas de compatibilidad.

La máquina virtual a desplegar en Google Compute Engine cuenta con 1 CPU virtual dedicado (2.0 GHz Intel Skylake) y 3.75 GB de memoria RAM. El costo mensual aproximado del servidor es de $24 dólares al mes aunque. Este costo se ve cubierto por los 300$ que Google le regala a sus nuevos desarrolladores por el primer més.

Por último, Google cuenta con un servicio de repositorios para imágenes de Docker en línea. Utilizando su herramienta de línea de comandos se puede subir la imagen al repositorio para luego, mediante una conexión SSH con la máquina virtual, ser descargada y finalmente ejecutada en la misma.

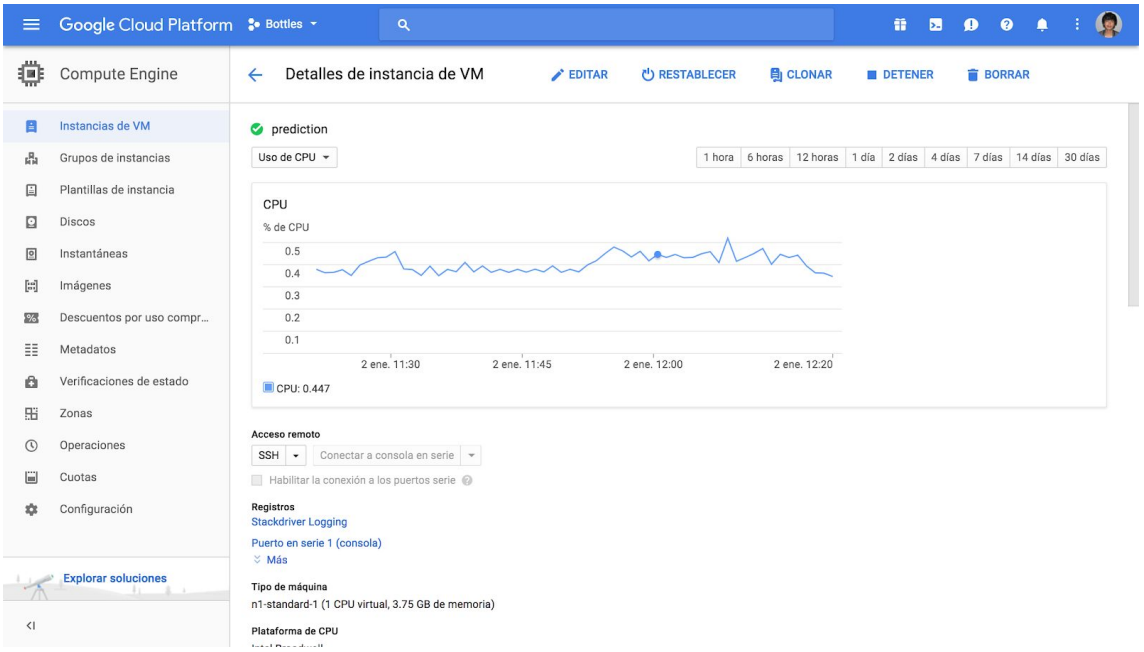


Figura 14. Consola de administración de Google Compute Engine.

Se debe también configurar una IP estática para el servidor. Esto se puede hacer desde la consola de configuración de la máquina virtual que se muestra en la Figura 14. Esto no tiene ningún costo adicional.

4.3 Despliegue del servidor de consultas

El servidor de consultas es aún más fácil de desplegar pues no necesita de Docker. La misma plataforma de Heroku provee una plantilla de servidor web en NodeJs lista para ser desplegada. Lo único necesario es contar con una cuenta para crear la instancia en el cual va a ser desplegado el servidor y subir el código del mismo al repositorio de Git asociado que contenga el archivo de configuración de Heroku. La Figura 15 muestra la consola de administración de Heroku con la instancia ya desplegada.

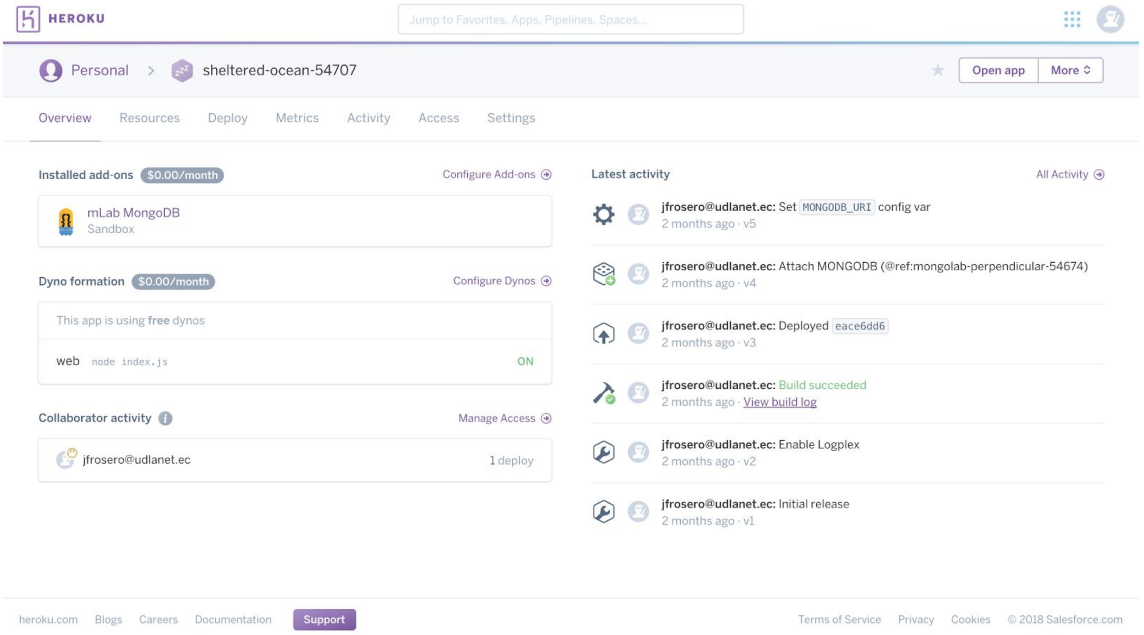


Figura 15. Consola de administración de Heroku.

4.4 Construcción del prototipo

Antes de construir el prototipo es necesario preparar el SBC para ser utilizado. Esto incluye tareas como instalar el sistema operativo que se va a utilizar y el programa en python encargado de controlar el servomotor, capturar la imagen y comunicarse con el servidor.

El sistema operativo a utilizar es la distribución de Debian que FriendlyARM, el fabricante del Nano Pi M1, provee para el mismo. Específicamente la versión 4.11 de Jessie. Para ello, es necesario descargar la imagen de la fuente oficial de FriendlyARM y formatear una memoria Micro SD con la misma. Se puede utilizar la herramienta Win 32 Disk Manager para esta tarea.

Una vez formateada la tarjeta e instalada en dispositivo ya se puede encender el SBC y visualizar la interfaz Gnome de Debian. Ya dentro del sistema, es necesario utilizar la línea de comandos para ejecutar la utilidad *npi-config*.

Mediante la misma se deben habilitar las características de SPI (Necesario para conectar la pantalla táctil) y PWM (necesario para controlar el servomotor) a utilizarse luego en la implementación.

Esta versión del sistema operativo viene con la versión 2.7 de Python ya instalada. Además, también tiene instalada la interfaz de OpenCv para Python. Las últimas dependencias de Python necesarias son: Tkinter (con la cual se construye la interfaz gráfica) y Request (utilizada para realizar peticiones HTTP al servidor). Las dos se pueden instalar mediante el comando pip, incluido junto con la distribución de Python.

Para el control del servomotor se utiliza la librería WiringNP, también desarrollada por FriendlyARM. Una vez instalada, se puede controlar todo el GPIO del dispositivo utilizando el comando *gpio.* Para utilizar la librería desde un programa de python se utiliza el módulo subprocess.

Finalmente se instala el programa de la interfaz y se configura a Debian para que cada vez que se inicie el sistema, se inicialice el programa de la interfaz.

Con el SBC listo, estos fueron los pasos a seguir para armar el prototipo:

1. Diseño del armazón utilizando un asistente de dibujo por computadora (SketchUp). Se planeó utilizar MDF de 6mm para la construcción del mismo. Esto facilitó su construcción pues del modelo en 3D del prototipo bastó sacar los perfile y cortarlos sobre una lámina de MDF para conseguir las piezas. Los esquemáticos de las piezas se encuentran en el Anexo 1.
2. Otros componentes como la barra que sostiene y hace girar la plataforma en la que se dispone la botella y el adaptador que conecta la misma con el servomotor tuvieron que ser fabricados de manera personalizada. Una vez listos, se juntaron las tres piezas: la plataforma, la barra y el adaptador para formar una sola pieza. Ésta en adelante se denominará plataforma.

3. Con estas partes listas: las piezas del armazón y la plataforma construida se puede armar el prototipo. Las tapas laterales del armazón cuentan con viñetas que permiten pegar las piezas fácilmente. Se pegan todas las piezas excepto la tapa. La plataforma se acopla al armazón mediante un rodamiento del lado opuesto al servomotor. El otro lado se conectará directamente con el servomotor.
4. Se adapta el servomotor al armazón y se conecta con la plataforma.
5. Se agrega la iluminación para la cámara de reconocimiento. Para ello se utilizó una cinta led que recorriera el perfil interior del prototipo, a 10 centímetros de la parte superior.
6. Se instalan las fuentes de poder para el servomotor y el SBC. Dentro del mismo circuito que conecta estas fuentes de poder, se alimenta la cinta led.
7. Se instala la pantalla táctil, la cámara y el SBC en la tapa del prototipo. La cámara se conecta mediante USB con el SBC. La pantalla, al ser diseñada precisamente para este SBC, se debe conectar mediante un cable bus con los pines del GPIO correspondientes.
8. Finalmente se conecta el pin que controla el servomotor con la entrada de señal del mismo y tierra para el GPIO.

A continuación se muestran algunas imágenes con el prototipo ya terminado

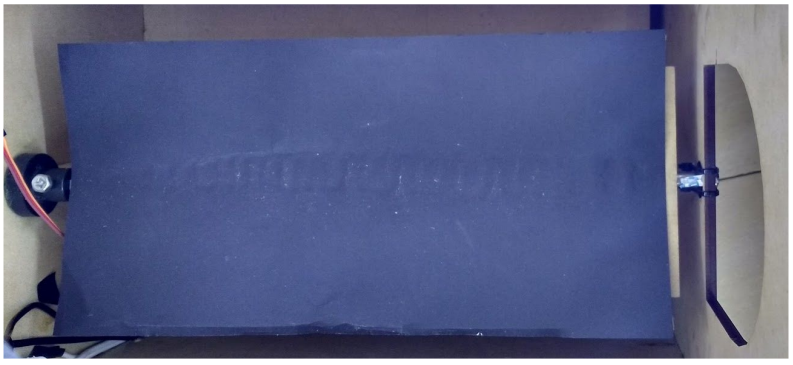


Figura 16. Plataforma de recepción al interior del prototipo.

Figura 17. Vistas laterales del prototipo

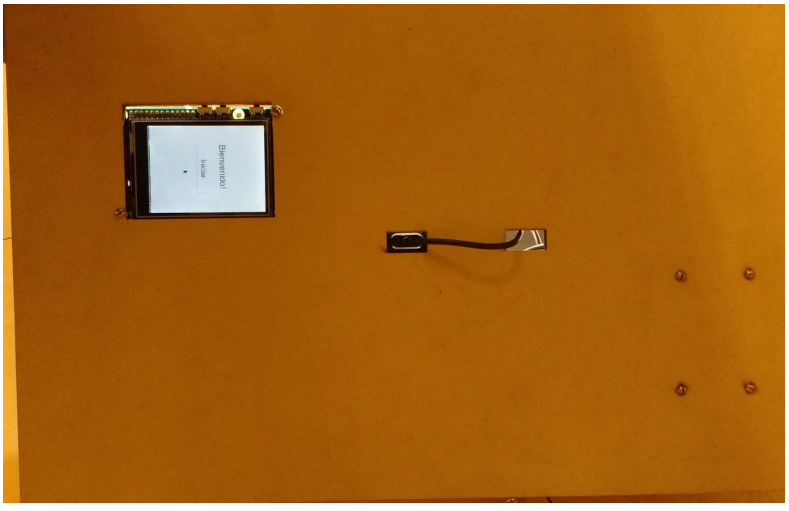


Figura 18. Vista superior del prototipo.

5. RESULTADOS

5.1 Resultados del entrenamiento de la red neuronal

Al utilizar el método de transferencia de aprendizaje, los resultados teóricos que arrojó la red neuronal en su entrenamiento fueron satisfactorios desde el principio.

Se hicieron 4 pruebas variando la cantidad de EPOCHs (ronda de entrenamiento con el set de datos), y el número de neuronas en la última capa completamente conectada.

Los siguientes gráficos muestran la precisión versus el número de Epoch en dos pasos. La gráfica de la izquierda muestra los resultados del entrenamiento del clasificador que se instala sobre las capas convolucionales de la red mientras que el de la derecha el resultado del ajuste final del modelo completo para cada caso:

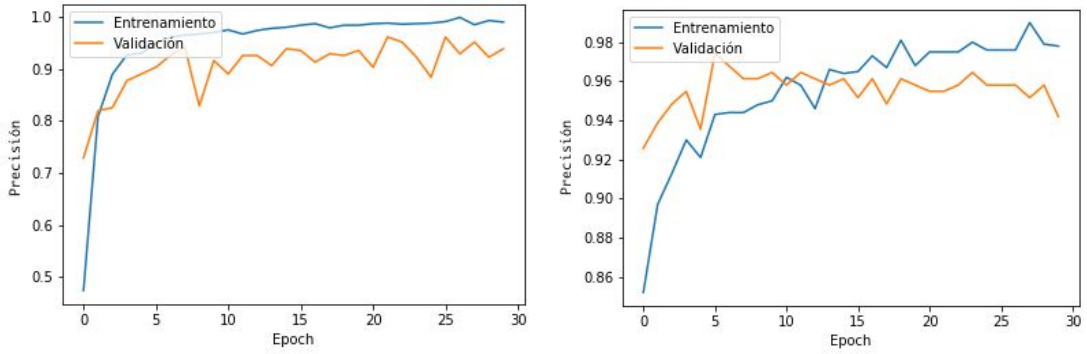


Figura 19. Entrenamiento con 30 epoch y 160 neuronas finales.

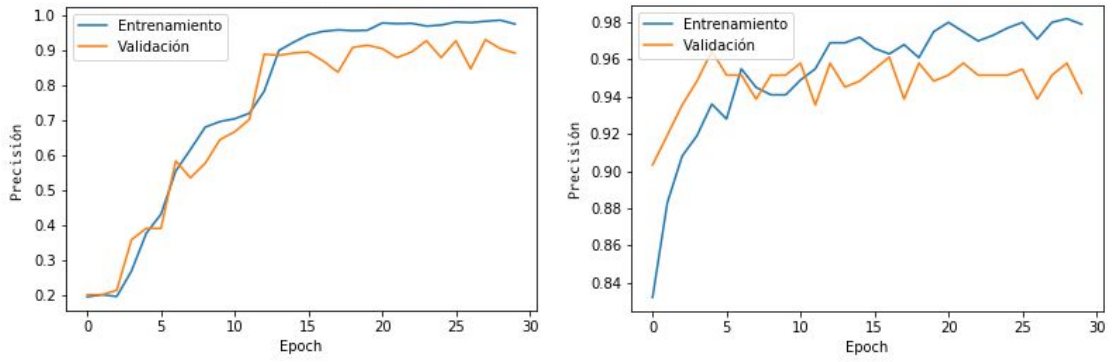


Figura 20. Entrenamiento con 30 Epoch y 640 neuronas en el último nivel.

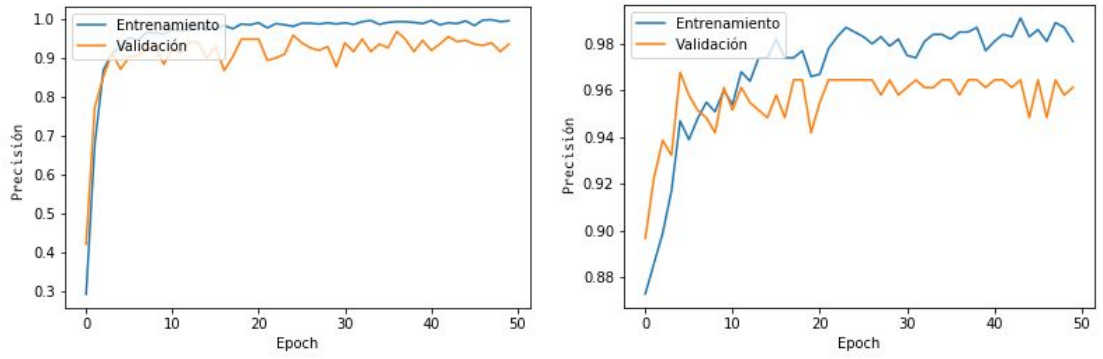


Figura 21. Entrenamiento con 50 Epoch y 160 neuronas en el último nivel.

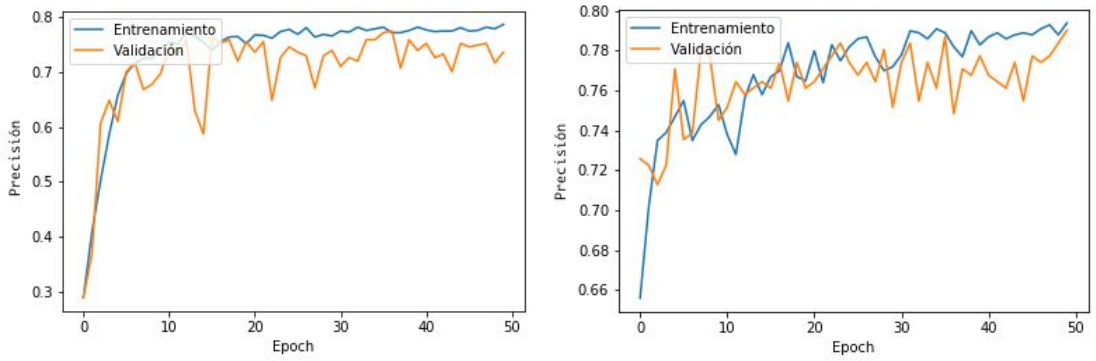


Figura 22. Entrenamiento con 50 Epoch y 640 neuronas en el último nivel.

Como se puede apreciar, el mejor resultado se da cuando se utilizan 160 neuronas en el último nivel y 50 Epoch. La precisión teórica alcanzada es del 96.6%.

5.2 Resultados del prototipo

La interfaz para la recepción de la botella (cuyas pantallas se muestran en la imagen 23), a pesar de que no muestra una interfaz altamente estética, es funcional. Los tiempos de respuesta que dependen del servidor son bajos. El promedio de respuesta en proceso de escaneo de la botella es de 3.8 segundos en las pruebas realizadas.

Figura 23. Vistas de la interfaz.

La pantalla táctil se integra bien con el prototipo. Esto da paso a que otras funciones sean añadidas sin necesidad de aumentar hardware. El tamaño es su único inconveniente.

Se realizaron otras pruebas para determinar la precisión práctica del prototipo. Estas pruebas son pertinentes pues el uso del prototipo por parte de un humano puede hacer que la precisión varíe. Después de 200 intentos con envases diferentes, 179 (89.5%) fueron exitosos al primer intento. De los restantes 21, 17 (80%) tuvieron éxito en el segundo intento. Los 4 envases

restantes tuvieron éxito al tercer intento. Ninguno de los envases se quedaron sin ser clasificados.

Por otro lado, se hicieron pruebas con envases que no se encontraban dentro de la muestra a analizar. Se tomaron 14 tipos de envases diferentes a los 5 iniciales. De los 14, 4 (28,5%) fueron aceptados como válidos, lo que resalta la necesidad de ampliar la cantidad de envases a catalogar.

Finalmente, en cuanto al precio de construcción del prototipo, la siguiente tabla resume los costos totales, resaltando si los componentes se compraron dentro del país o fuera de él, pues la segunda opción resulta menos costosa.

Tabla 4.

Costos finales del prototipo.

Cant	**Item**	**Precio (USD)**	**Nacional o Internacional**
1	Nano Pi M1	15	Internacional
1	Pantalla táctil 2,8”	10	Internacional
2	Plancha MDF	30	Nacional
1	Fuente de poder 5v 3A MicroUSB	15	Internacional
1	Fuenete de poder 5v 3A	10	Internacional
1	Servomotor	15	Nacional
1	Cámara web	10	Nacional
	Corte de láminas	15	Nacional
	Otros(Pegamento, tornillos, adaptadores, etc...)	10	Nacional
	Total	**$ 120**	

6. DISCUSIÓN

Si bien la idea de hacer que los usuarios entreguen sus envases reciclables mediante una máquina para obtener recompensas no es novedosa, esta implementación de esa misma idea trae consigo los siguientes beneficios:

1. El costo de la máquina es bajo. Una búsqueda rápida en internet muestra precios de máquinas de vending reverso que pueden costar hasta 10 veces más. Hay que tener en cuenta también que si se decidiera producir el prototipo, su costo sería incluso menor.
2. La capacidad de recolectar datos. El tener un servidor que se encargue del reconocimiento de la botella abre un mundo de posibilidades sobre los datos que se pueden recolectar al mismo tiempo que se brinda el servicio de reconocimiento y recolección.
3. La versatilidad de la interfaz. Al utilizar una pantalla táctil, las posibilidades de aplicaciones que se puedan correr sobre esta plataforma son extensas.

Estas ventajas pueden ser utilizadas para generar un modelo de negocio sustentable que potencie el reciclaje de envases en áreas urbanas.

6.1 Modelo de negocio

Aprovechando las ventajas que el prototipo brinda se generó un modelo de negocio utilizando el modelo Canvas. Este se encuentra adjunto en el Anexo 2.

En resumen, el modelo pretende obtener ingresos de la recolección y venta de envases reciclables hasta conseguir una posición en el mercado que permita generar relaciones con empresas envasadoras que estén interesadas en los datos que se obtienen de la recolección.

7. CONCLUSIONES Y RECOMENDACIONES

7.1 Conclusiones

El uso de redes neuronales convolucionales para el reconocimiento de envases PET y de cualquier otro tipo es completamente viable. Los beneficios de su uso incluyen: la baja necesidad de datos de entrenamiento (gracias a las técnicas de transferencia de aprendizaje), la alta precisión, la flexibilidad para agregar nuevas clases.

A pesar de que este sistema está construido para reconocer envases, la tecnología con la que se construye puede adaptarse para resolver una variedad inmensa de problemas que tengan que ver con el reconocimiento de objetos.

Las iteraciones en el diseño del sistema permitieron llegar a una solución que encaje en las necesidades del proyecto. Esto no solo ayudó a solucionar el problema sinó también a expandir el campo de conocimiento de este estudiante.

Se implementó exitosamente un sistema de recolección de botellas que le permita al usuario obtener una recompensa y consultarla en tiempo real. El mismo incluye el prototipo de bajo costo encargado de la recolección de los envases con una interfaz gráfica que permite controlarlo.

El incremento en la cantidad de neuronas en la última capa completamente conectada no refleja una mejora en la capacidad de reconocimiento de la red. Esto puede ser consecuencia de que solo haya 5 clases distintas.

El mercado actual de IasS y PaaS facilita inmensamente que propuestas como estas se hagan realidad ya que reducen el costo de desarrollo e implementación de los servidores necesarios.

Se cumplieron los objetivos propuestos en el anteproyecto además de otros retos que surgieron a partir de la solución propuesta, como el levantamiento de los servidores.

7.2 Recomendaciones

Después del desarrollo de este proyecto se pueden brindar las siguientes recomendaciones:

Aprovechar las facilidades que brinda la librería de Keras para el desarrollo de Redes Neuronales. Su API es extremadamente fácil de entender y utilizar.

En lo posible utilizar la versión 3 de Python. A pesar de que existe soporte extenso para la versión 2.7, las versiones que se actualizan con más regularidad y que contienen nuevas características son las 3.x

Utilizar librerías de control de versiones para Python como *venv.* Esto acelera el proceso de desarrollo gracias a que permite replicar un ambiente de desarrollo de Python en cualquier otro lugar.

Tomar en cuenta el soporte que se le dé actualmente al SBC con el que se decida trabajar. En el caso de este proyecto, surgieron algunos inconvenientes en cuanto a librerías desactualizadas o inexistentes que podrían haber sido sustentados fácilmente si se hubiese trabajado con otro proveedor más popular como Raspberry.

Referencias

Adit D. (2016). A Beginner's Guide To Understanding Convolutional Neural Networks. Recuperado el 17 de Noviembre del 2017, de: https://adeshpande3.github.io/adeshpande3.github.io/A-Beginner%27s-Guide-To-Understanding-Convolutional-Neural-Networks.

Agarwa, M (2017). Back Propagation in Convolutional Neural Networks — Intuition and Code. Recuperado el 18 de Enero del 2017 de: https://becominghuman.ai/back-propagation-in-convolutional-neural-networks-intuition-and-code-714ef1c38199

Alexandre, L. A. (2015). 3D Object Recognition Using Convolutional Neural Networks with Transfer Learning Between Input Channels. Intelligent Autonomous Systems 13 Advances in Intelligent Systems and Computing,889-898. doi:10.1007/978-3-319-08338-4_64

Chollet, F. (2016). Building powerful image classification models using very little data. Recuperado el 28 de Noviembre, 2017, de https://blog.keras.io/building-powerful-image-classification-models-using-very-little-data.html

Jain, A. K., Mao, J., & Mohiuddin, K. M. (1996). Artificial neural networks: a tutorial. Computer, 29(3), 31-44. doi:10.1109/2.485891

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). ImageNet Classification with Deep Convolutional Neural Networks. In Advances in Neural Information Processing Systems 25 (NIPS 2012). Recuperado el 28 de Noviembre del 2017 de:

https://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf

INEC. (s.f.). Tecnologías de la Información y Comunicaciones (TIC´S) 2016[PDF]. Recuperado el 28 de Noviembre del 2017 de: http://www.ecuadorencifras.gob.ec/documentos/web-inec/Estadisticas_Sociales/TIC/2016/170125.Presentacion_Tics_2016.pdf

Mulder, W. D., Bethard, S., & Moens, M. (2015). A survey on the application of recurrent neural networks to statistical language modeling. Computer Speech & Language, 30(1), 61-98. doi:10.1016/j.csl.2014.09.005

Murray, C. (2017, 13 de Febrero). Deep Learning with Keras on Google Compute Engine – Google Cloud Platform - Community – Medium. Recuperado el 28 de Noviembre del 2017, de https://medium.com/google-cloud/keras-inception-v3-on-google-compute-engine-a54918b0058

Ohtani, K., & Bab, M. (2012). Shape Recognition and Position Measurement of an Object Using an Ultrasonic Sensor Array. Sensor Array. doi:10.5772/36115

Razavian, A. S., Azizpour, H., Sullivan, J., & Carlsson, S. (2014). CNN Features Off-the-Shelf: An Astounding Baseline for Recognition. 2014 IEEE Conference on Computer Vision and Pattern Recognition Workshops. doi:10.1109/cvprw.2014.131

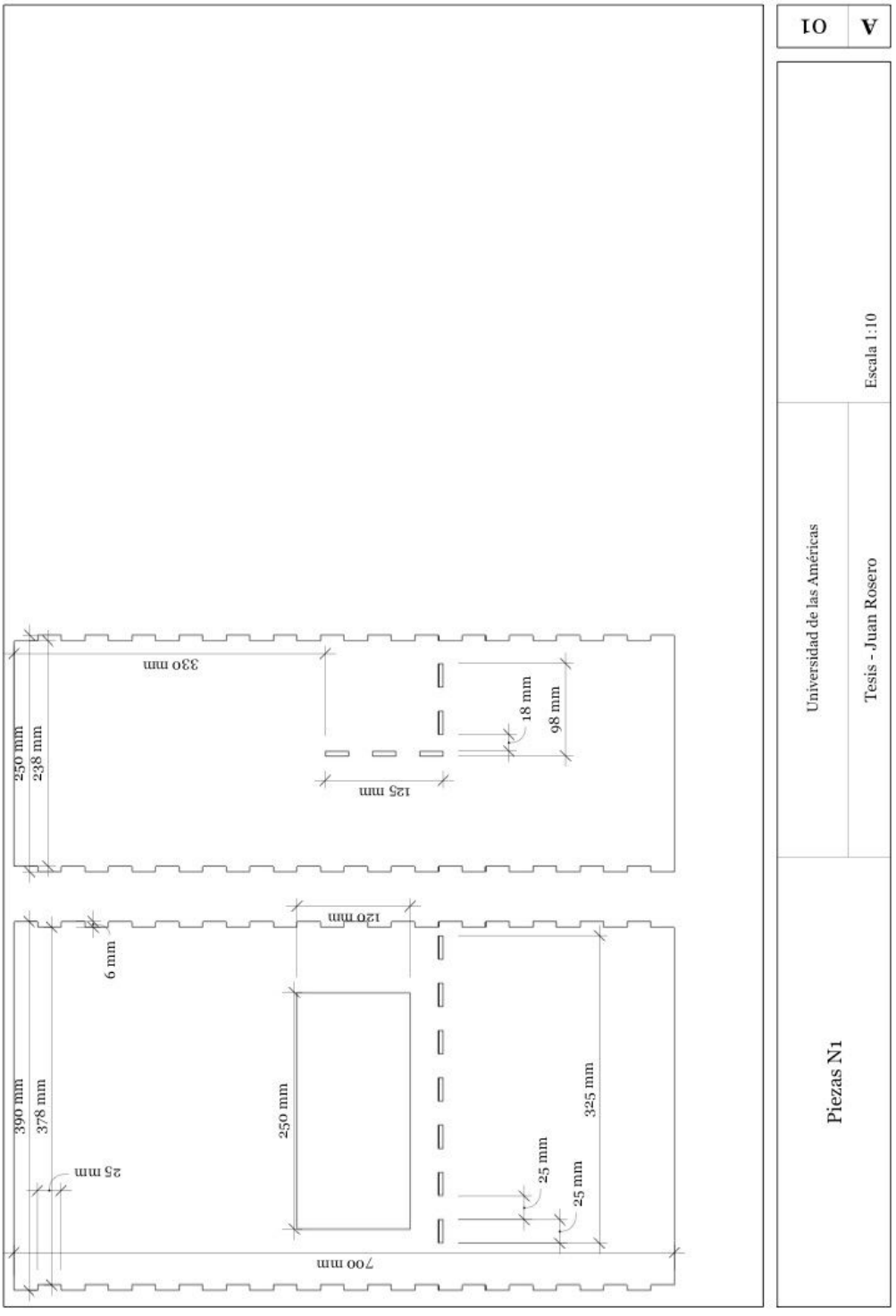
Szegedy, C., Toshev, A., & Erhan, D. (2014). Deep Neural Networks for Object Detection. In Advances in Neural Information Processing Systems 26 (NIPS 2013). Recuperado el 28 de Noviembre del 2017 de: https://papers.nips.cc/paper/5207-deep-neural-networks-for-object-detection.pdf

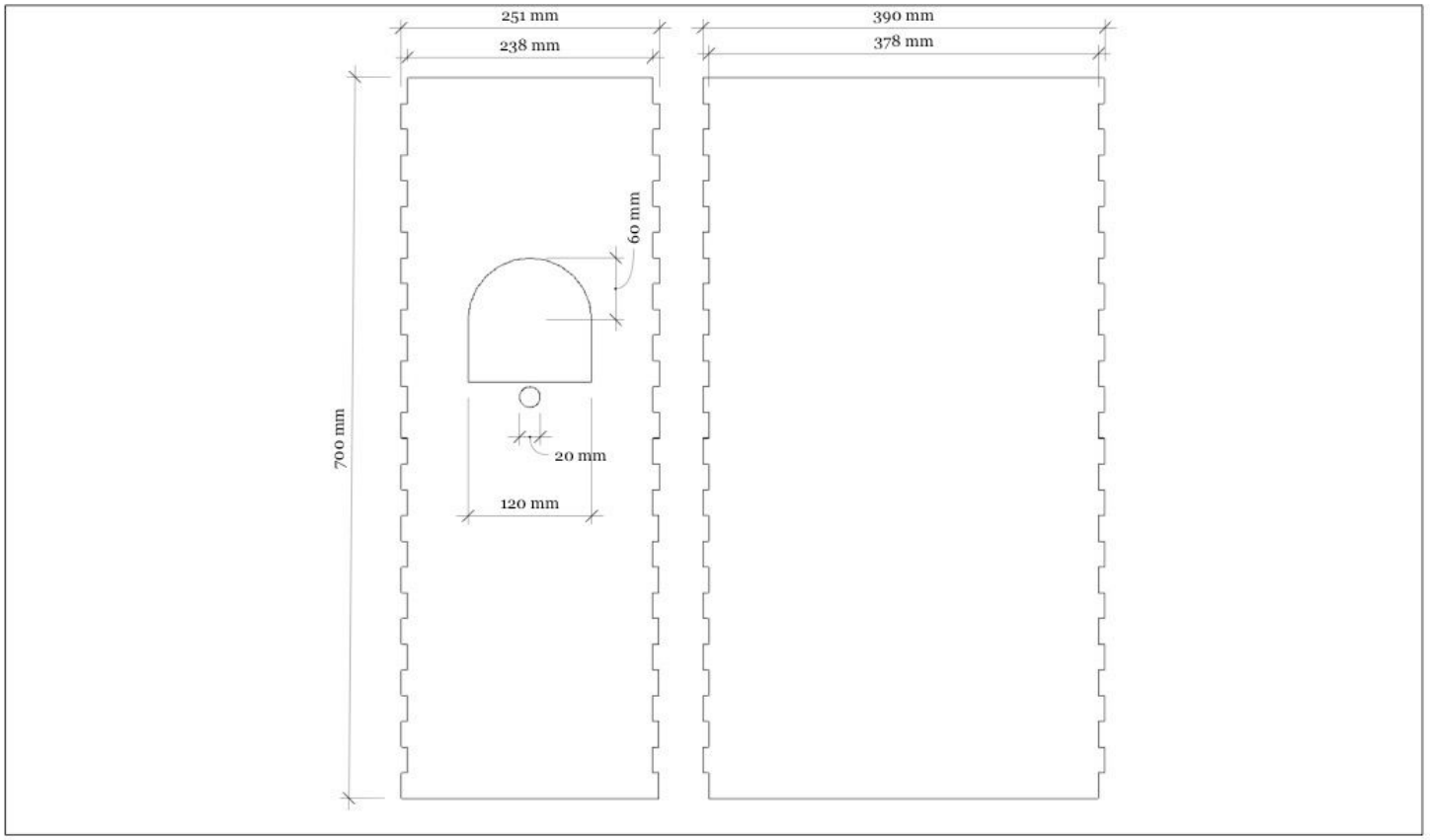
W.S. McCulloch , W. Pitts (1943). A Logical Calculus of Ideas Immanent in Nervous Activity. Bull. Mathematical Biophysics, Vol 5. pp 115 -135

Yosinski , J., Clune, J., Bengio, Y., & Lipson, H. (2014). How transferable are features in deep neural networks? . InIn Advances in Neural Information Processing Systems 27 (NIPS 2014). Recuperado el 28 de Noviembre del 2017 de: https://arxiv.org/pdf/1411.1792.pdf

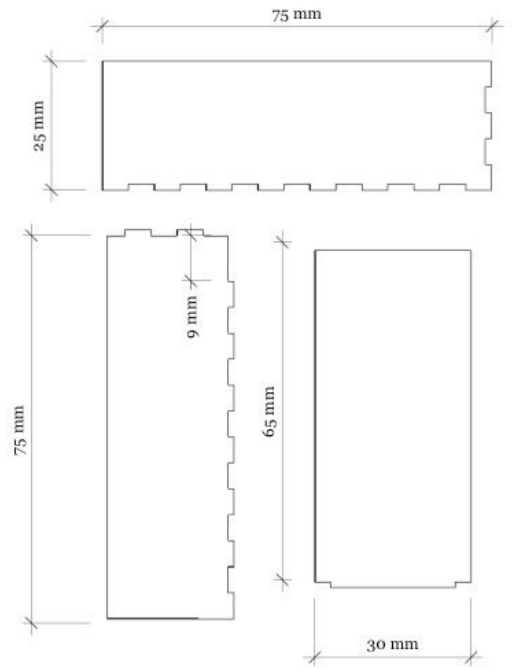
ANEXOS

Anexo 1. Esquemáticos de las piezas del prototipo.





Piezas N2	Universidad de las Américas		A 01
	Tesis - Juan Rosero	Escala 1:10	



Piezas N3	Universidad de las Américas		A 01
	Tesis - Juan Rosero	Escala 1:10	

Anexo 2. Canvas del modelo de negocio

Socios Clave

Empresas envasadoras de bebidas

Empresas dedicadas al reciclaje de envases a los cuales les interese comprar los envases reciclados.

Dueños de tiendas de barrio pequeñas que puedan alojar una máquina recolectora.

Actividades Clave

La fabricación en masa de los recolectores.

El mejoramiento continuo de las interfaces.

El marketing de nuestro servicio.

La instalación de los recolectores al rededor de las zonas urbanas.

La recolección y venta de los envases entregados.

La generación de relaciones con las empresas envasadoras.

Recursos Claves

La tecnología utilizada para el reconocimiento de envases.

Los datos que se obtengan de la recolección de los envases.

El capital para empezar la producción y distribución de las máquinas recolectoras.

Una lista de tiendas interesadas en convertirse en tiendas recolectoras.

Una fuente de ingreso de la recolección y venta de los envases.

Propuesta de Valor

Mediante nuestros servicios, nuestros usuarios obtienen una manera gratuita y conveniente de reciclar los envases que ya no les son utiles.

Esta recolección se realiza mediante máquinas automáticas instaladas en tiendas pequeñas dentro de la zona urbana. Las tiendas reciben un pequeño ingreso por alojar la máquina, en función de los envases recolectados por la misma.

Por otro lado, gracias a los datos recolectados, las empresas envasadoras de bebidas pueden obtener información valiosa sobre sus clientes, además de un metodo para reforzar la fidelidad de los mismos.

Relaciones con los clientes

Se espera construir un grupo de usuarios que se acerquen a los puntos de recoleccion a entregar sus envases a cambio de incentivos económicos o establecidos por las empresas envasadoras.

Ademas, se deben mantener relaciones con los dueños de las tiendas en las que se recolecten los envases.

La entrega de envases es totalmente gratuita, mientras que la información que se brinda a las empresas envasadoras, y la oportunidad de aprovechar nuestro canal tiene un costo. Finalmente, los dueños de las tiendas recolectoras obtienen un beneficio económico por alojar la máquina.

Al inicio del proyecto ninguna de ellas se habrá establecido.

Canales

Se debe llegar a los usuarios mediante medios como radio, televisión, redes sociales. También a través de medios publicitarios impresos (vallas, posters) al rededor de la ciudad y en las tiendas recolectoras.

Por otro lado con las tiendas recolectoras se deben establecer relaciones personalmente, así como con las empresas recolectoras.

Segmentos de Mercado

Creamos valor primeramente para las personas que entregan los envases. Deben ser personas que residan dentro de un area urbana, principalmente entre 18 y 50 años, familiarizados con el uso de un dispositivo electrónico e interesados en el reciclaje.

En segundo lugar estan los dueños de tiendas pequeñas dentro de zonas urbanas.

Finalmente, tambien creamos va lor para las empresas envasadoras interesadas en utilizar nuestros servicios.

Estructura de costo

El costo más importante a cubrir es el de la producción y distribución de las máquinas recolectoras. Este costo se reduce si se producen más unidades.

Luego está el costo de marketing de la marca y el servicio. Este es el costo más representativo.

Después, está el costo de mover los envases de las tiendas recolectoras hacia nuestros compradores. Este costo aumenta en función del número de máquinas recolectoras activas.

Finalmente, el costo de relacionarse con las empresas envasadoras. Este costo es fijo en el tiempo.

Fuentes de ingreso

La fuente inicial de ingresos viene de la venta de los envases recolectados.

Eventualmente, cuando la empresa se encuentre en la posición de negociar con las empresas envasadoras, se pretende levantar acuerdos en los cuales se venda la información obtenida por la recolección y la posibilidad de utilizar las máquinas recolectoras como un medio para reforzar la fidelidad a determinada marca.