TECNOLÓGICO NACIONAL DE MÉXICO

Instituto Tecnológico Superior de Teziutlán

DESARROLLO DE UNA APLICACIÓN PARA LA IDENTIFICACIÓN DE PET, CARTÓN Y PAPEL POR MEDIO DE UNA RED CONVOLUTIVA.

TESIS QUE PRESENTA:

Aram Filemón Beciéz Salazar

Como requisito parcial para obtener el título de:

MAESTRO EN SISTEMAS COMPUTACIONALES

ASESOR (A):

M.E.L.: Emmanuel Vázquez Benito

Fracción I Y II S/N, Aire Libre, Teziutlán, Puebla
Agosto 2020

Agradecimientos

Me permito manifestar mi agradecimiento al Maestro y Director de tesis Emanuel Vázquez Benito, no solo por su impecable labor de dirección si no por sus continuas enseñanzas y apoyo en la elaboración de este trabajo. Así mismo a mis amigos y compañeros de Grado quienes recibí su apoyo durante el tiempo de estudio, finalmente pero no por ello menos importante al Instituto Tecnológico Superior de Teziutlán por el apoyo brindado mediante la beca para realizar mis estudios de Maestría en Sistemas Computacionales.

A todos ellos muchas gracias.

Mientras los filósofos discuten
si es posible o no la inteligencia artificial
los investigadores la construyen.
C. Frabetti

Resumen

La presente tesis trata sobre el reconocimiento de imágenes por medio de una red neuronal convolutiva (CNNs) por sus siglas en inglés, las cuales son un modelo de red neuronal, donde las neuronas corresponden a campos receptivos, de una manera muy similar a las neuronas de la corteza visual primaria de un cerebro biológico. Una red neuronal debe aprender cómo procesar la información de entrada antes de que sea utilizada en una aplicación, su proceso de entrenamiento, formado por los datos de cada nodo entrada, y la salida es la respuesta deseada para cada nodo, una vez echo este proceso sólo los datos de entrada son provistos a la red, que recordara la respuesta que aprendió durante el entrenamiento. (Goodfellow, 2012).

La red se compone de múltiples capas, en el principio se encuentra la fase de extracción de características, compuesta de neuronas convolucionales y de reducción. A medida que se avanza en la red se disminuyen las dimensiones activando características cada vez más complejas, al final se encuentran neuronas sencillas para realizar la clasificación de imágenes; de esta manera, el proyecto tiene como finalidad desarrollar una aplicación para la identificación de botellas de pet, y los materiales inorgánicos: papel y cartón. Por otra parte, desde el punto de vista social, se pretende dar solución a la ineficiente clasificación de botellas de pet, papel y cartón por medio de la tecnología, que pretende ser de gran impacto al medio ambiente.

Índice

Dedicatoria

A mis padres y hermanos quienes han sido el pilar más importante durante mi vida, pues si de luchar, trabajar honradamente, respetar y formarme como ser humano se trata, no encuentro mejor ejemplo…

Cartas de aceptación

CAPITULO I

GENERALIDADES DEL PROYECTO

1.9. Introducción

Las redes neuronales por convolución son un elemento importante de las tecnologías de inteligencia artificial, por lo que IA es una rama de la computación que se encarga de los problemas de la percepción, razonamiento y aprendizaje en relación con sistemas artificiales.

Modelos simplificados de estas redes, de las que está constituido el cerebro, y al igual que este intenta aprender a partir de los datos que se suministran. Elementos simples organizados basándose en la emulación por medio de hardware o software de la actividad de las neuronas del sistema nervioso, por lo que la serie de neuronas individuales de las que cada neurona actúa como un elemento procesador independiente, las entradas y pesos de interconexión son procesados por una función sumatoria, el resultado de esta operación es mapeado por una función de transferencia de características no lineal y la salida de esa función no lineal es la salida de la neurona. De modo que modelos matemáticos especializados pueden aplicarse en dominios concretos, así las redes neuronales convolucionales están mostrando su utilidad en muchos problemas reales, y pueden tener cientos de unidades procesadoras, como un sistema computacional hecho por un alto número de elementos procesadores, altamente conectados, que procesan la información por la respuesta dinámica de entradas externas. (Loncomilla, 206)

Pensar en la basura nos genera un rechazo inmediato hacia está, sin embargo, tenemos que convivir con ella y no solo en nuestro hogar, sino a la vuelta de cualquier esquina, en calles, a orillas de las carreteras, en los parques, en las plazas de mercado; en fin, en cualquier lugar. Todo esto es el resultado de las diversas actividades que realiza el hombre en su diario vivir, donde ha generado una producción excesiva de desechos, los cuales se convierten en un inconveniente mayor a la hora de almacenarlos, disponerlos o eliminarlos. Es por eso que se hace necesario aprender a manejar y aprovechar adecuadamente las basuras que producimos, dejarlas de ver como la percibimos y verlas como

residuos que son objetos y que se puede transformar en otro bien, con valor económico; en especial los sólidos.

El documento se encuentra estructurado en cuatro capítulos, así: el primer capítulo contiene la descripción del problema; que va dar una idea clara de la situación en general con relación a la problemática de la clasificación de residuos sólidos, posteriormente encontramos la formulación del problema, las principales preguntas de investigación, así como los objetivos generales, específicos, planteados para el desarrollo del trabajo.

En el segundo capítulo plantea la metodología y desarrollo en la cual se apoya conceptualmente esta investigación, así como el proceso metodológico, se describe la metodología utilizada y sus objetivos, las técnicas de recolección de datos, para desarrollar la aplicación que clasifique por medio de una red neuronal convolutiva los materiales inorgánicos como el pet, papel y cartón. Encontraremos en el tercer capítulo como se llevará a cabo la implementación y pruebas.

Por último, está el capítulo cuatro, corresponde al análisis e interpretación de resultados; este análisis es con respecto a cómo se está llevando a cabo el proceso de la clasificación he identificación de residuos por medio de una red neuronal convolutiva.

1.2. Planteamiento del problema

México es uno de los países en el que su población está menos concientizada en cuestión de basura., en donde la gente le da poca importancia a la clasificación de la misma, en consecuencia, es que la mayoría aún ni siquiera toman conciencia sobre tirar la basura en su lugar.

Día a día se observa como las calles de diferentes ciudades, municipios y pueblos se ven llenas por todos lados de botes, envases, papeles, colillas de cigarro y entre otras que se haya convertido “inservible” para las personas.

Según datos de ECOCE, el acopio de PET en México ha sido el más alto de América en los últimos tres años. En 2014, el consumo aparente nacional de PET virgen para envases rodeó las 700,000 toneladas, de las cuales se recuperaron 57.8%. De lo acopiado en el país, 46.2% se valorizó y se comercializó en México, mientras que 56.8% fue a exportación, principalmente a China y a Estados Unidos. Así mismo, se destaca, que México es líder mundial en reciclado botella a botella grado alimenticio, con 57%, seguido por Estados Unidos con 22% y por la Unión Europea, con 22%, según información de Napcor, PCI y (ECOCE, s.f.).

México es uno de los tres líderes en consumo de papel y cartón reciclado en el mundo, solo detrás de Hong Kong y Malasia, pues 4.9 millones de toneladas de estos materiales se reciclan cada año según datos de la Cámara Nacional de las Industrias de la Celulosa y el Papel , es líder en el continente Americano en consumo de pet con 722 mil toneladas al año, pues el desecho de estos, ha crecido considerablemente, desechando al día un total de 388 mil kilogramos de Pet de tipo 1 Polietileno tereftalato (botellas de agua y refrescos), papel y cartón con 343 mil kg, esto solo en la cd de Puebla, de acuerdo con la Secretaría de Medio Ambiente y Recursos Naturales (SEMARNAT, Google, 2019), son estos materiales los más desechados en la ciudad, ya que por consumo de habitante menciona que 2 botellas al día son desechadas y el 60 por ciento del papel en México se emplea en empaques, 13 por ciento corresponde a la escritura y la impresión, 22 al sanitario y facial, 4 a periódico y 1 por ciento a otros.

Solo en la Unión Europea y por la parte de América, se encuentra que estos sistemas, ya que las leyes hacen obligatorio el reciclaje, en donde empresas como Tomra de Noruega, Wincor Nixdorf de Alemania, reVend Recycling del Reino Unido y Envipco de los Estados Unidos hacen uso de esta tecnología para la clasificación de residuos sólidos. Por lo que en nuestro país encontramos que en el IV congreso internacional de ingeniería mecatrónica y automatización-CIIMA 2015, se presentó un sensor para el reconocimiento de materiales a través de la capacitancia. Según lo expuesto en el IV congreso internacional de ingeniería mecatrónica y automatización, dicho sensor utiliza un oscilador de Colpits en el cual se remplaza uno de sus condensadores por dos placas paralelas en donde se coloca los diferentes materiales, estos generan cambio de frecuencia los cuales son transformados en voltaje para su posterior lectura y procesamiento del sistema embebido (EIA, 2015).

1.3. Justificación

El manejo inadecuado de los residuos sólidos genera una problemática ambiental, en donde la sociedad rompe con el equilibrio ecológico, originándose por la escasa cultura de la clasificación de estos desechos, por lo que se pretende incentivar a la comunidad por medio de una aplicación innovadora, para su clasificación y recolección, es por ello, que en el presente proyecto se desarrollará una aplicación que utilice una red neuronal convolutiva para la clasificación de los residuos inorgánicos: papel, cartón y pet, la red convolutiva funciona con un modelo previamente establecido o entrenado, así una red ya entrenada se puede usar para hacer predicciones o clasificaciones, la aplicación funcionará en conjunto con un sistema mecánico para la clasificación de los materiales, los usuarios introducirán el material en un compartimiento de entrada, posterior a esto se realiza el análisis y la identificación, y por último se clasifica el material.

Para el reconocimiento de patrones, existen diferentes tipos de redes neuronales que realizan diferentes tareas para la identificación de imágenes como lo son, perceptrón, adaline, perceptrón multicapa, neuronal concurrente, la desventaja es, que en estas, cada una de sus neuronas no se une con todas y en cada una de las capas siguientes, si no que solo con un subgrupo de ella, por lo que a diferencia de una red neuronal por convolución consigue reducir el número de neuronas necesarias y la complejidad computacional para su ejecución. Dichas redes constan de múltiples capas diseñadas para requerir un preprocesamiento relativamente pequeño en comparación con las redes descritas anteriormente, así aprenden usando filtros y aplicándolos a las imágenes, tomando un cuadrado pequeño (o 'ventana') y comienza a aplicarlo sobre la imagen. Cada filtro permite que la CNN identifique ciertos patrones en la imagen. La CNN busca partes de la imagen donde un filtro coincida con el contenido de la imagen. A medida que se entrena la red se disminuyen las dimensiones activando características cada vez más complejas. Al final se encuentran neuronas sencillas para realizar la clasificación.

En México, el reciclaje para la sociedad no es obligatorio así la escasa cultura de la misma. De tal manera que se pretende desarrollar este tipo de tecnología más eficiente, ayudando a la conservación del medio ambiente.

1.4. Hipótesis

La implementación de un mecanismo permitirá la identificación y clasificación de pet, cartón y papel por medio de una aplicación basada en redes convolucionales.

1.5. Objetivo general

En conjunto con Inteligencia Artificial, desarrollar una aplicación para la identificación de pet, cartón y papel utilizando redes neuronales convolutivas.

1.6. Objetivos específicos

- Investigar que es una red convolutiva.
- Identificar los requerimientos funcionales de la aplicación.
- Diseñar la interfaz de la aplicación.
- Analizar patrones en los materiales a identificar.
- Entrenar la red convolutiva.
- Documentar de acuerdo a la metodología *OOHDM*
- Determinar la eficiencia de la aplicación.
- Construir un prototipo para la separación de los materiales inorgánicos.

1.7 Alcances

- La aplicación a desarrollar por medio de una red neuronal convolutiva identificará los materiales: pet, papel y cartón.
- Se desarrollará un prototipo mecánico que permita introducir residuos y este los clasifique según su naturaleza, además de reflejar las características y ventajas de integrar las herramientas para lograr el objetivo de identificar los materiales.

1.8 Delimitación

Para la clasificación de los materiales se consideran las siguientes restricciones:

- Las botellas deben estar vacías.
- El papel y cartón debe estar seco.
- El sistema muestra aquellos materiales que no fueron aceptados.
- Rechazar o emitir una alerta en caso de que el objeto depositado no sea una botella, papel o cartón válido.

- La aplicación debe identificar los materiales como el pet, papel y cartón desde el compartimiento de entrada hasta el contenedor, sin arriesgar la integridad del usuario, así como el proceso de validación.
- La aplicación deberá contar con una interfaz para que la interacción con el usuario sea sencilla y confiable, de tal modo que muestre información adicional de acuerdo a las necesidades del cliente.
- Los sistemas deberán integrar el control y programación en lenguajes de programación de alto nivel.

1.9. Estado del arte

Las redes neuronales convolucionales pertenecen a un conjunto de técnicas agrupadas en aprendizaje profundo, una rama del aprendizaje automático, que ha demostrado ser exitosa en los últimos años en tareas de reconocimiento de imágenes. La presente tesis busca implementar el uso de redes neuronales convolucionales en la identificación y clasificación de los desechos: pet, papel y cartón. De modo que se desarrollará una arquitectura de red neuronal convolucional basada en bibliotecas de código abierto para la clasificación de imágenes, se entiende por arquitectura a los artefactos resultantes del seguimiento del desarrollo metodológico de construcción de software, así como los planos y el prototipo funcional del sistema mecánico de clasificación, la red neuronal será entrenada con el mayor número de imágenes para lograr un óptimo reconocimiento.

Para su aplicación los buscadores de imágenes como Google imágenes, se basan principalmente en el nombre del archivo que contiene la imagen, en el nombre del enlace o en el texto que aparece en la página donde se encuentra la imagen. Es decir que la búsqueda se basa en información de texto y no en información gráfica.

Actualmente se han desarrollado aplicaciones como Google Goggles1 basadas en el reconocimiento de imágenes utilizando el contenido de las mismas, para

ello se centran en un subconjunto de las imágenes que Google ha catalogado previamente ya que el análisis y comparación de imágenes digitales requiere de un coste computacional muy alto. Realizar esta labor para todas las imágenes indexadas por un buscador sería una tarea casi imposible, a día de hoy (GogglesAplication, 2015).

El campo de procesamiento de imágenes está continuamente evolucionando, durante los últimos años ha habido un incremento significativo en el interés en campos como morfología de imágenes, redes neuronales artificiales, procesamiento de imágenes en color y/o en escala de grises, comprensión de datos de imágenes, reconocimiento de imágenes y sistemas de análisis basados en conocimiento.

Se puede encontrar una gran cantidad de aplicaciones de reconocimiento de imágenes en el mercado, por destacar algunas, podríamos mencionar:

- Reconocimiento de rostros y expresiones faciales.
- Reconocimiento de firmas.
- Reconocimiento de caracteres.

Otro de los proyectos en los cuales se ha implementado la detección de imágenes por medio de redes neuronales es reconocimiento de imágenes utilizando redes neuronales artificiales desarrollado por la facultad de informática, universidad complutense de Madrid España, el cual describe el proceso de extracción de patrones característicos de imágenes, mediante la ayuda de Redes Neuronales Artificiales. La información de la Red Neuronal junto con datos adicionales de las imágenes, son almacenados en una base de datos y consumidos por un servicio web. Un teléfono móvil con sistema operativo Android consumirá la información almacenada en el servicio web. Posteriormente al realizar una captura de imagen con la cámara del teléfono, este procesará la imagen y junto con los datos consumidos por el servicio web será capaz identificar de qué imagen se trata.

Para finalmente hacer mención de una aplicación por medio de una red convolutiva para la detección facial, descrita por los autores Idelfonso Jiménez

Silva y Sergio Antonio Cruces Álvarez mencionan que el reconocimiento facial representa un importante campo de estudio en la actualidad debido a la variedad de aplicaciones para las que puede ser empleado, que van desde la mejora de la seguridad o aplicaciones industriales hasta utilidades en dispositivos personales. (Antonio-Diego, 2019)

Por otro lado, las redes neuronales han experimentado en los últimos años un incremento de su uso en todo tipo de sectores gracias a su adaptación a diferentes problemas.

CAPÍTULO II

METODOLOGIA Y DESARROLLO

2.1. Fundamentos teóricos

A continuación, se muestran los diferentes pasos llevados a cabo para la consecución de los objetivos marcados en el trabajo:

- ¿Qué es Inteligencia Artificial?
- ¿Qué es una red convolutiva?
- Herramientas para entrenar una red neuronal

2.1.1. ¿Qué es Inteligencia Artificial?

En este capítulo se hace mención al concepto de Inteligencia Artificial, surge en 1950 de la mano del investigador Alan Turing con su artículo “Computing Machinery and Intelligence”, donde se presenta el famoso Test de Turing en el cual se propone demostrar si una determinada máquina es inteligente o no, en base a la capacidad de generación de respuestas a diferentes preguntas. Por lo que en la década de 1990 alcanza el punto de partida de esta ciencia, en la que toma fuerza con grandes aportaciones de los investigadores de la época, grandes inversores apoyan esta tecnología ante la necesidad mejorar la capacidad de procesamiento y análisis de la enorme cantidad de datos con las que ya se trabajaban. (Artificial., 2020) En la actualidad IA es la disciplina científica que trata de imitar el comportamiento del cerebro humano para realizar diferentes tareas por medio de una máquina. Además, la IA tiene diferentes ramas para realizar diferentes tareas, como lo es Machine Learning definido en 1959 por Arthur Samuel como el campo de estudio que da a los ordenadores la habilidad de aprender sin ser explícitamente programados. (Samuel., 2019) Una vez creado el programa, éste es capaz de aprender de forma autónoma a realizar alguna tarea que requiera de cierta inteligencia todo esto mediante un previo entrenamiento, de esta manera se describe como se desarrollará el entrenamiento de una red neuronal convolutiva, las técnicas de procesamiento de imágenes utilizadas, así como las características principales de la red al

plantear la investigación para determinar los factores que inciden en la selección e identificación de los materiales inorgánicos.

El DeepLearning es una de las técnicas más utilizadas dentro del entorno del Machine Learning inspirada en las estructuras básicas del cerebro humano, las neuronas. Este enfoque trata de emular como entendemos el funcionamiento del cerebro, no a la creación de un cerebro. Dichos sistemas, comúnmente llamados Redes Neuronales Artificiales son capaces de realizar tareas de clasificación o predicción tras un entrenamiento previo, lo cual hace que sea un aprendizaje supervisado. (-, 2020)

2.1.2. ¿Qué es una red convolutiva?

Comenzando a definir, ¿qué es una convolución? Se describe como una operación matemática que combina dos señales para producir una tercera señal. A diferencia del resto de redes neuronales, en lugar de tener pesos independientes por cada neurona, los pesos se utilizan para definir un filtro convolucional que se aplica a la información de entrada para que a continuación, esta función se aplique a la activación como resultado de la convolución. Su principal aplicación es la de procesado de imagen o vídeo y su estructura combina las convoluciones con capas de reducción. Hoy en día son muy utilizadas debido a sus buenos resultados, aunque sus requerimientos computacionales son mayores que con otro tipo de redes. (neuronales, 2020)

Las principales capas de las que están compuestas estas redes son:

- Capa convolucional: capa de filtrado para la extracción de características de la imagen. Está formada por un número definido de filtros a los cuales se define su tamaño, estos se desplazarán por toda la imagen. A su salida se generan matrices de datos que contienen las características extraídas, su tamaño dependerá del tamaño del filtro y por lo tanto también de la cantidad de pesos que se incluyan, también dependerá del tipo de desplazamiento del filtro y la posibilidad de incluir un padding externo.

- Capa de reducción: también llamada ’pooling’, se encarga de reducir el tamaño de las matrices anteriormente obtenidas para disminuir el número de datos a procesar. La reducción dependerá del tamaño de ventana seleccionado. Se pueden emplear diversas técnicas como la media o máximo del conjunto de valores de la ventana.

- Capa de entrada totalmente conectada: también llamada “flatten”, su función será transformar las matrices de datos en un único vector que contiene toda la información característica de la entrada.

- Primera capa totalmente conectada: proporciona un nivel mayor de profundidad a la red de predicción de características. Es opcional.

- Capa de salida totalmente conectada: muestra los niveles de probabilidad para cada clase de salida.

En la siguiente ilustración se muestra el esquema de una red neuronal convolutiva

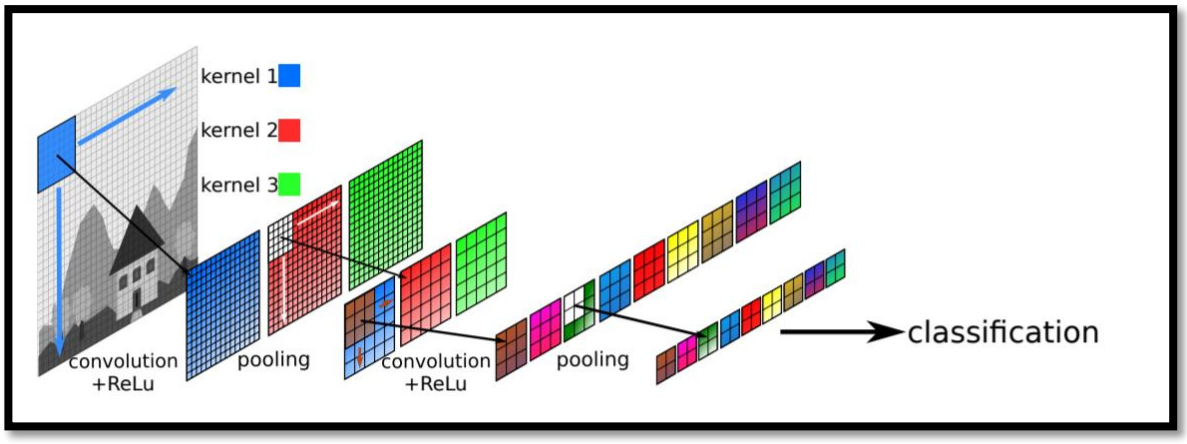


Figura 1. Esquema de una red neuronal convolutiva.

Por lo que podemos definir que la convolución, es una operación de filtrado lineal básica en la visión computacional, la cual utilizando diferentes filtros o kernels se aplica para cometidos como el desenfoque, el aumento de nitidez, acentuar el relieve o la detección de bordes en una imagen que es lo que queremos utilizar para la solución del problema. Se trata por red de convolución en términos de aprendizaje automático a la visión por computador o redes neuronales, de modo que hay tres conceptos principales a tener en cuenta: la entrada, de una imagen la cual se representará como el kernel, se representa con la letra K y la salida, el mapa de características. La convolución de un kernel K de tamaño m × n colocado en el píxel (i, j) de una imagen I (Inc., 2004).

Por medio de las capas convolucionales de las redes normalmente aplican varias convoluciones en paralelo en diferentes puntos (i, j), esto puede entenderse mejor visualizando un kernel como una ventana deslizante que recorre toda la imagen y calcula el mapa de características resultante en cada salto. En la Figura 2 se muestra cómo se desliza un kernel (la ventana azul) por toda la matriz que constituye la imagen, se multiplican los valores del kernel con los valores de la imagen en cada posición, y se suman todos los valores de la ventana deslizante, consiguiendo como resultado un mapa de características (matriz de la derecha en la Figura 2), la cual tiene una dimensión menor que la imagen original. Esta reducción en la dimensión dependerá del tamaño de salto a la hora de deslizar

el kernel por la imagen, y del tamaño del kernel en sí (aprendemachinelearning, 2016).

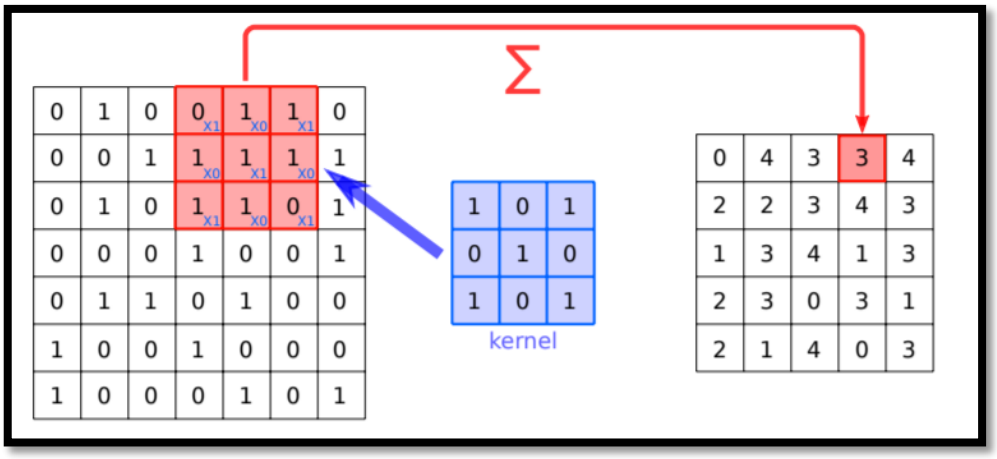


Figura 2. Convolución de un Kernel.

2.1.3. Herramientas para entrenar una red neuronal

Para el desarrollo de la Red Neuronal Convolucional se requiere de un modelo entrenado de lo contrario se llevaría un tiempo de aproximadamente dos semanas entrenando por medio de GPU con equipos muy potentes, que desafortunadamente el precio para adquirir estos equipos no está al alcance del público, costos de estos van desde los cuatrocientos mil pesos hasta cerca del millón de pesos , es por ello que la empresas como Imagnet y Open source, proporcionan repositorios de código abierto facilitando un recurso útil para investigadores, educadores, estudiantes. (IMAGNET, 2020)

Originalmente, la expresión open source (o código abierto) hacía referencia al software open source (OSS). El software open source es un código diseñado de manera que sea accesible al público: todos pueden ver, modificar y distribuir el código de la forma que consideren conveniente. El software open source se desarrolla de manera descentralizada y colaborativa, así que depende de la revisión entre compañeros y la producción de la comunidad. Además, suele ser más económico, flexible y duradero que sus alternativas propietarias, ya que las encargadas de su desarrollo son las comunidades y no un solo autor o una sola empresa. (OpenSource, 2020)

Se describen las herramientas de software y hadware utilizadas para consecución en la elaboración del proyecto, por tanto, una red neuronal convolutiva sabemos que utiliza múltiples filtros aplicados a un mismo valor de entrada mediante capas convolucionales y de pooling, lo cual permite sacar provecho de la extracción de patrones que brindan diferentes tamaños en los filtros, así, posteriormente, el resultado de estos filtros es concatenado y utilizado como el valor de salida del módulo. Este modelo aumenta el número de parámetros entrenables y la computación requerida, para mejorar considerablemente la precisión. Se incluye un directorio de imágenes para el entrenamiento y uso de la convolución para ser aplicado al mayor número de imágenes que serán obtenidas por medio de una cámara web. Las imágenes de entrenamiento serán almacenadas para su posterior uso, organizándose en un directorio de imágenes, de esta manera, las herramientas que se implementan en el proyecto se describen a continuación: (Crispi, 2020)

2.1.3.1. Python

Es un lenguaje de programación de código abierto, orientado a objetos. Tiene una sintaxis sencilla que cuenta con una vasta biblioteca de herramientas, que hacen de Python un lenguaje de programación único. (Python.org, Python.org, 2020)

2.1.3.2. Python en la Inteligencia Artificial (AI)

Python es un lenguaje de escritura rápido, escalable, robusta y de código abierto, ventajas que hacen de Python una gran herramienta perfecta para la Inteligencia Artificial. Permite plasmar ideas complejas con unas pocas líneas de código, lo que no es posible con otros lenguajes. Sus principales bibliotecas como Keras y TensorFlow, contienen información sobre las funcionalidades del aprendizaje automático. Además, existen bibliotecas proporcionadas por Python, que se usan mucho en los algoritmos AI como Scikitl, una biblioteca gratuita de aprendizaje automático que presenta varios algoritmos de regresión, clasificación y agrupamiento. Pero, sobre todo, Python es un lenguaje gratuito de código abierto con una gran comunidad en activo, que proporciona soporte a cualquier programador. Todas estas razones combinadas, hacen que aprender Python sea una opción fácil sobre otros lenguajes para aplicaciones de inteligencia artificial. (Enzyme, 2020)

2.1.3.3. Tensorflow y Keras

Como se describió, las principales librerías para el reconocimiento de imágenes son Tensorflow y Keras, TensorFlow es una biblioteca de código abierto creada para Python por el equipo de Google Brain. Tiene la capacidad de compilar muchos algoritmos y modelos diferentes juntos, lo que permite al usuario implementar redes neuronales profundas para usar en tareas como reconocimiento / clasificación de imágenes y procesamiento de lenguaje natural, funciona mediante la implementación de una serie de nodos de procesamiento, cada uno representa una operación matemática, y la serie completa de nodos se denomina “gráfico”. (JordiTorres.AI, 2020)

La librería de Keras, es una API de alto nivel (interfaz de programación de aplicaciones) que puede usar las funciones de TensorFlow (así como otras bibliotecas de ML como Theano). Keras fue diseñado con facilidad de ser implementada con diferentes funciones de Tensorflow y está configurada para trabajar con Python sin modificaciones o configuraciones importantes.

2.1.3.4. OpenCV-Python

Python es un lenguaje de programación de propósito general iniciado por Guido van Rossum, que se hizo muy popular en poco tiempo principalmente debido a su simplicidad y legibilidad de código. Permite al programador expresar sus ideas en menos líneas de código sin reducir la legibilidad. En comparación con otros lenguajes como C / C ++, Python es más lento. Pero otra característica importante de Python es que se puede ampliar fácilmente con C / C ++. Esta característica nos ayuda a escribir códigos computacionalmente intensivos en C / C ++ y crear un contenedor de Python para que podamos usar estos contenedores como módulos de Python. Tiene como ventajas: primero, el código es tan rápido como el código C / C ++ original (ya que es el código C ++ real que funciona en segundo plano) y segundo, es muy fácil de codificar en Python. Así es como funciona OpenCV-Python, es un contenedor de Python en torno a la implementación original de C ++. (Programarfacil, 2020)

2.1.3.5. Numpy-Python

El apoyo de Numpy hace la tarea más fácil, es una biblioteca altamente optimizada para operaciones numéricas. Da una sintaxis de estilo MATLAB. Todas las estructuras de matriz de OpenCV se convierten en matrices de Numpy. Entonces, independientemente de las operaciones que pueda hacer en Numpy, puede combinarlo con OpenCV, que aumenta el número de armas en su arsenal. Además de eso, varias otras bibliotecas como SciPy, Matplotlib que admite Numpy se pueden usar con esto. Entonces OpenCV-Python es una herramienta apropiada para la creación rápida de prototipos de problemas de visión por computadora. (Numpy, 2020)

2.1.3.6. Hardware

Las características principales del host son:

Dispositivo Laptop Hp Omen, característica, Intel i7 de 8va generación, tarjeta de video Nvida 1050, memoria Ram 16 Gb.

2.2. Metodología de la investigación

A continuación, se muestran algunos de los pasos para definir y sistematizar el conjunto de técnicas, métodos y procedimientos para el desarrollo del presente proyecto como lo son:

- Encuestas
- Platicas o conferencias
- Investigación Correlacional

Para definir y sistematizar el conjunto de técnicas, métodos y procedimientos que se deben seguir durante el desarrollo, el presente proyecto se utilizará el tipo de investigación descriptiva, ya que uno de los problemas está enfocado a la población por la falta de educación para clasificar los residuos inorgánicos, en este caso el manejo inadecuado de los residuos sólidos genera una problemática ambiental, en donde la sociedad rompe con el equilibrio ecológico, originándose por la escasa cultura de la clasificación de estos desechos, de esta manera se pretende incentivar a la comunidad por medio de una aplicación innovadora, para su clasificación y recolección, es por ello que se desarrollará una aplicación que utilice una red neuronal convolutiva para la clasificación de los residuos inorgánicos: papel, cartón y pet.

Por lo anterior se realizaron encuestas en el Instituto Tecnológico Superior de Teziutlán, para conocer la opinión sobre la clasificación de los residuos inorgánicos, así como su reutilización, sobre el conocimiento de tecnologías y el manejo de las mismas, que están siendo aplicadas para revertir el desecho de los residuos, se implementarán pláticas, sobre la importancia de clasificar los residuos como pet, papel y cartón, se mencionará la aplicación a desarrollar para que resulte innovadora para la sociedad.

2.2.1. Encuesta para los Alumnos del Instituto Tecnológico Superior de Teziutlán

Mediante la siguiente encuesta se obtienen los siguientes datos:

1. ¿Qué cantidad de PET generas a la semana?

1 kilo____ 2 kilos____ 3 kilos____

2. ¿Qué porcentaje de tus residuos son plásticos de un solo uso?
   10%____ 40%____ 60%____

3. ¿Cuántos kilos aproximadamente generas de cartón a la semana?
   1 kilo____ 2 kilos____ 3 kilos____

4. ¿Reutilizas las hojas blancas o papel?
   Sí____ No____

5. ¿Con base a tus residuos que porcentaje de papel generas a la semana?
   10%____ 40%____ 60%____

6. ¿Consideras factible la implementación de una maquina recolectora de PET en el TEC?
   Sí ____ No ____ Tal vez ____

7. ¿Qué gratificación te gustaría que ofreciera la máquina?
   Saldo electrónico ____ Un porcentaje para comedor ____
8. ¿Te sumarías a esta causa para el reciclaje?
   Sí ____ No ____ Tal vez ____

9. ¿Cuál crees, que es el residuo que predomina más en el Tecnológico?
   Cartón ____ Papel ____ PET ____

10. ¿Consideras que el TEC debe realizar más campañas para promover el reciclaje?
    Sí ____ No ____ Tal vez ____

11. Actualmente, ¿Qué residuos conoces que se les dé un proceso de reciclaje en el TEC?

    Ninguno ____ Plásticos ____ Cartón ____

12. ¿Sabe qué tiempo tardan en desaparecer los residuos?

    Sí ____ No ____

Los resultados que se obtuvieron fueron los siguientes:

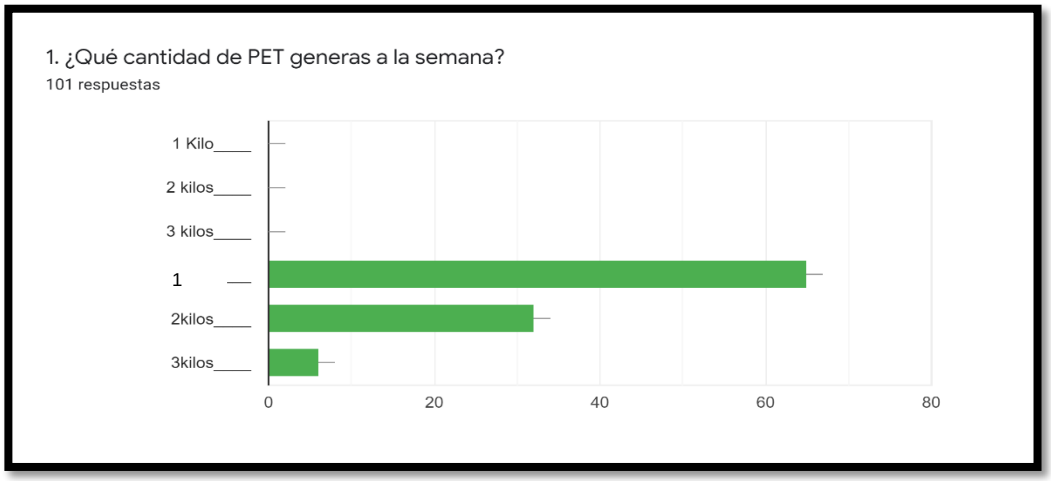


Figura 3. Porcentaje generado a la semana

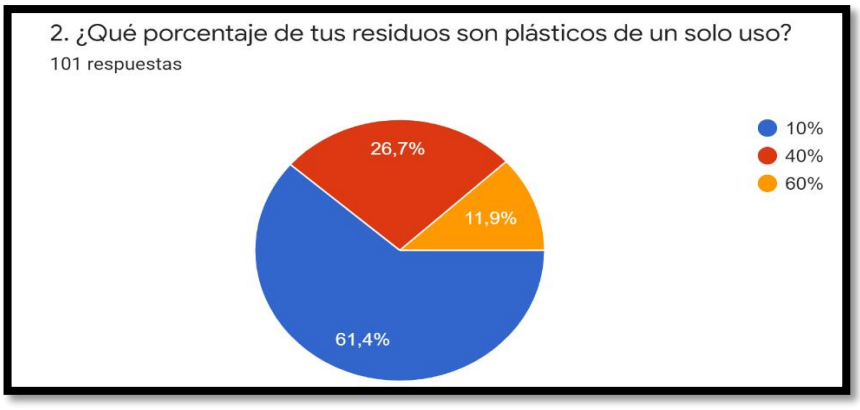


Figura 4. Porcentaje Residuos plásticos de un solo uso.

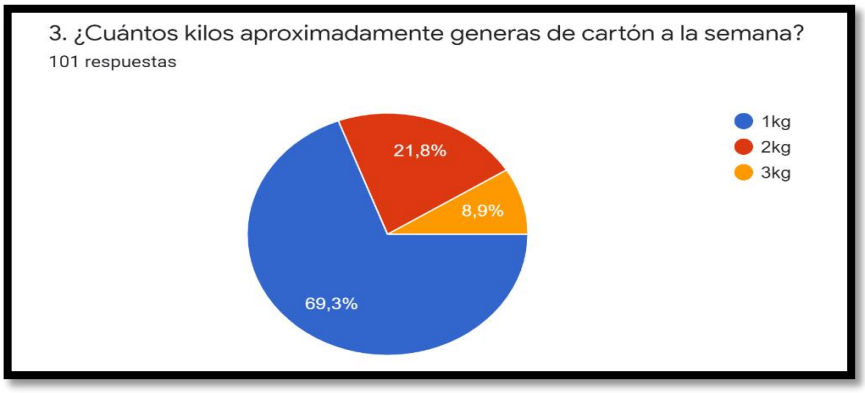


Figura5. Kilos de cartón generados a la semana.

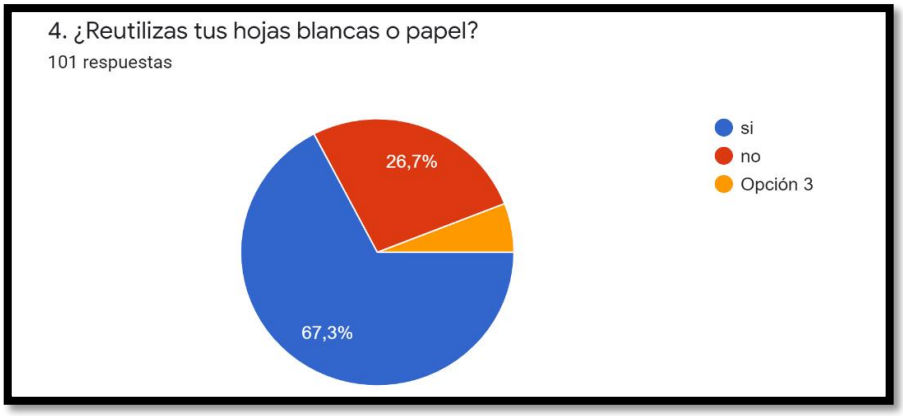


Figura 6. Porcentaje de hojas blancas o papel reutilizables

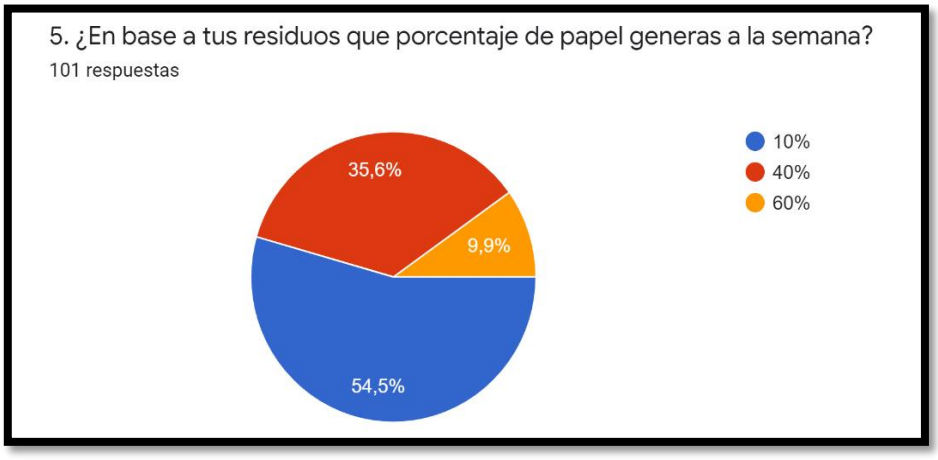


Figura 7. Porcentaje de papel generado a la semana.

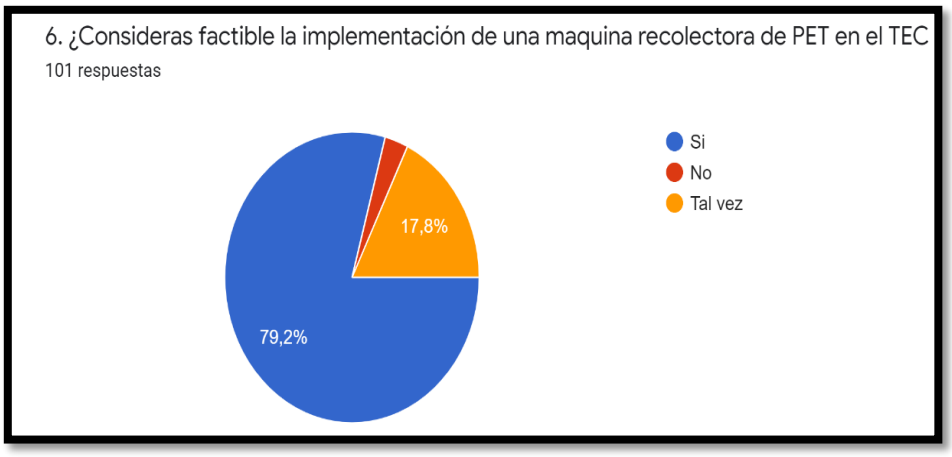


Figura 8. Factibilidad de la maquina recolectora.

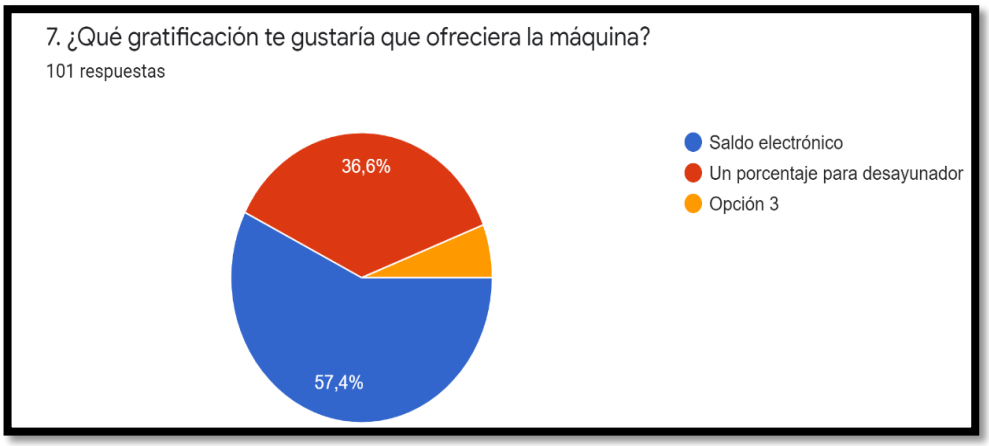


Figura 9. Incentivó de la máquina.

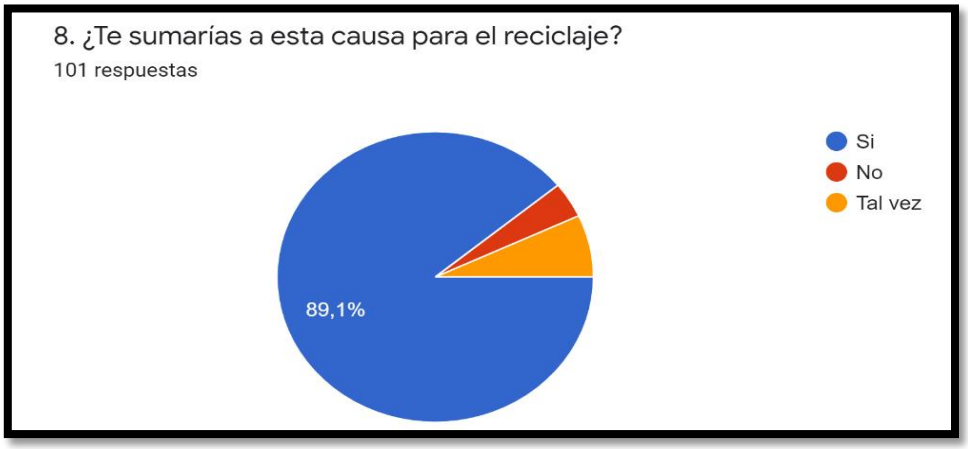


Figura 10. Respuesta favorable.

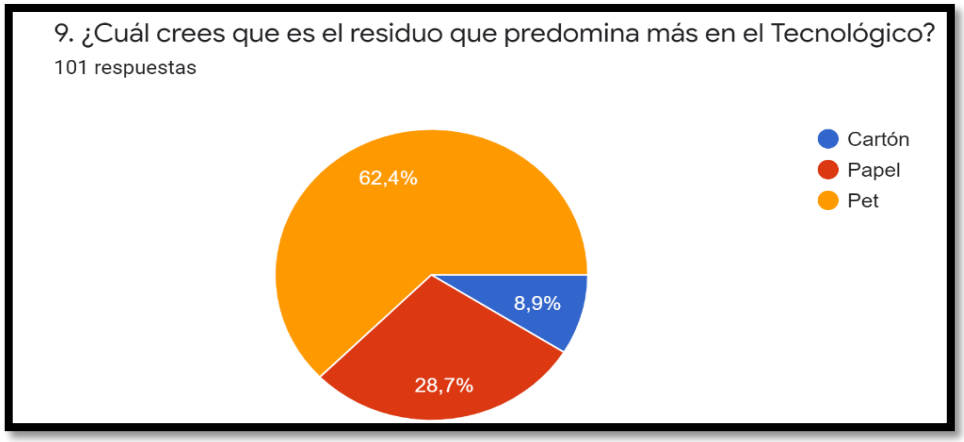


Figura 11. Residuo que predomina más en el Tec.

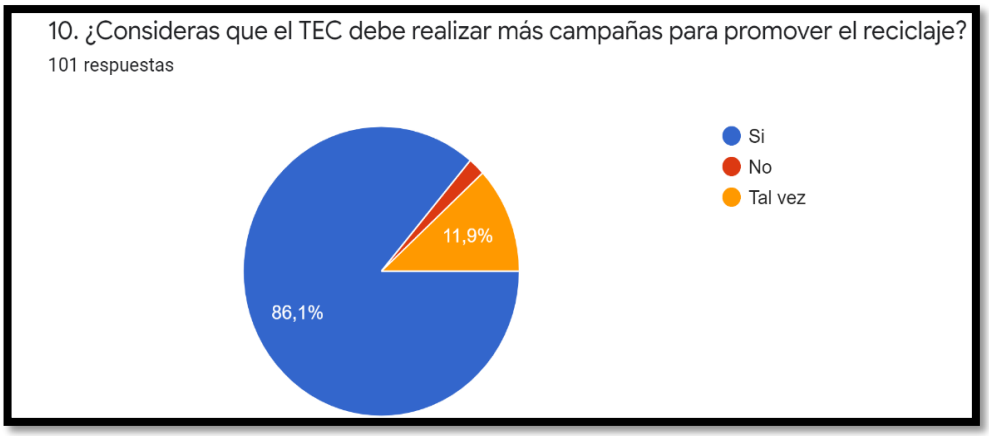


Figura 12. Promoción de reciclaje.

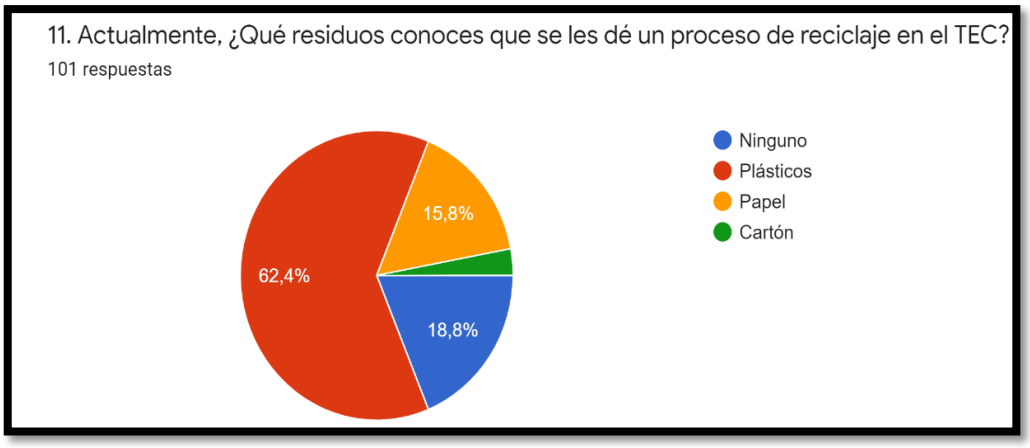


Figura 13. Porcentaje de residuos procesados en el Tec.

De acuerdo a los resultados por parte de los alumnos, se puede concluye que en su mayoría generan el 64% de pet a la semana, el 66% de los alumnos generan por semana 1 kg de cartón, el 64 % reutiliza el papel y el 50.6% genera el 10% a la semana de papel. De esta manera el material desechado y que predomina en el instituto por parte del alumnado se encuentra el plástico con un 59.6%, de modo que con un 79.8% la comunidad cree que es necesario una maquina recolectora de estos residuos, por lo que la comunidad del Tecnológico Superior de Teziutlán, están dispuestos a participar en la recolección de materiales, de igual forma concuerdan que es importante realizar campañas para concientizar el reciclado de los residuos, así como para una mejor recolección de los mismos y que crezca la cultura del reciclado.

2.2.2. Conferencia Día Internacional del Medio Ambiente

En el marco conmemorativo al día internacional del medio ambiente que se celebró el viernes 29 de mayo del presente año, el Instituto Tecnológico de Teziutlán hizo extensa la invitación, en la presentación del proyecto de grado cuyo enfoque está dirigido al cuidado del medio ambiente, por lo que se presentó el prototipo AbdRecycler y el impacto que puede generar en la región de Teziutlán, Puebla, como se muestra en la siguiente figura.

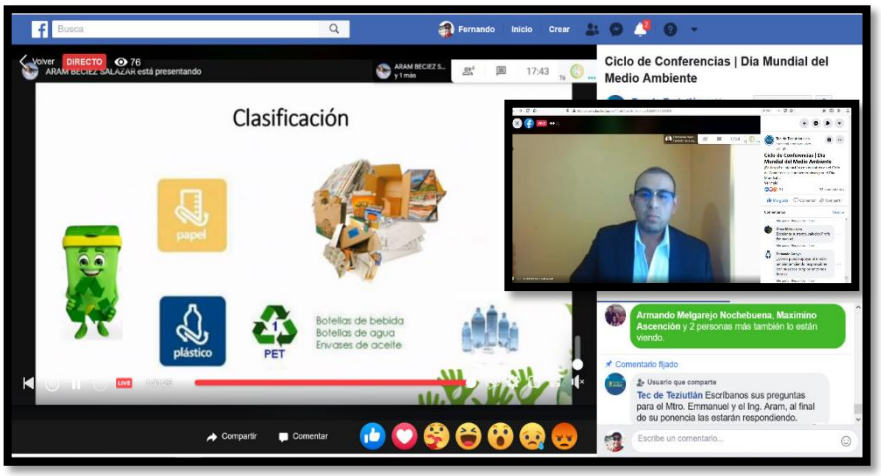


Figura 14. Conferencia Medio Ambiente

2.2.3. Investigación Correlacional

Por medio de la investigación correlacional se analizará el crecimiento de la población en los últimos 10 años, para conocer el grado de contaminación de los desechos como pet, papel y cartón que se han generado, así mismo el impacto de las tecnologías que se comenzaron aplicar dentro de esos años para conocer su crecimiento y su aplicación a este factor para la conservación del medio ambiente. Por lo anterior se pretende incentivar a la comunidad del Instituto Tecnológico de Teziutlán por la falta de educación que presentan, para clasificar los residuos inorgánicos por medio de una aplicación innovadora.

2.3. Metodología de desarrollo

El presente apartado describe el desarrollo del proyecto y los diferentes pasos llevados a cabo para la consecución de los objetivos marcados del presente:

- Requerimientos de Entrenamiento de la Red Neuronal
- Diagrama de operación y construcción del prototipo
- Metodología de desarrollo OOHDM
- Casos de uso
- Desarrollo del entrenamiento de CNN para la identificación y clasificación de materiales pet, papel y cartón

2.3.1. Requerimientos de Entrenamiento de la Red Neuronal

Para comenzar, se parte de unas imágenes que se capturan por medio de una cámara web, por lo que se requiere capturar de un N número de imágenes para realizar el análisis, y así extraer las características que permitirán entrenar la red neuronal convolutiva y esto nos permita identificar y clasificar las imágenes para conocer qué características son las más adecuadas para cada tipo de imagen, todo esto se realizará por medio del lenguaje de programación Python el cual contiene las librerías TensorFlow y keras para la clasificación he identificación de imágenes para el entrenamiento de la red neuronal, así como también la implementación de OpenCv. Una vez entrenada, los datos ofrecidos por la red que son: pesos de sus neuronas, vías, número de capas ocultas, neuronas de entrada y de salida) junto con información adicional de cada imagen, nombre, descripción, de los datos que estamos utilizando. Para crear la arquitectura del desarrollo de la aplicación se utiliza aprendizaje supervisado en cada una de las imágenes como resultado se obtiene una matriz de colores , aquí se pasará cada imagen por medio de las Apis de TensorFlow y keras, durante el proceso y análisis las imágenes serán una matriz de colores, para así transformar por las diferentes capas que la conforman, esto con la finalidad de entrenar a la red neuronal, la cual devolverá una etiqueta que va indicar el tipo de material. Se instalará el entorno de programación de Python, este software implementa anaconda para gestionar las librerías como sklearn mlp (multilayer perceptrón) para la red neuronal, este proceso enviará parámetros por medio de sklear mlp.

2.3.2. Diagrama de operación y construcción del prototipo

Una vez que sea logrado el entrenamiento de la red neuronal convolutiva, se implementara en una maquina como prototipo, el ciclo de su funcionamiento, comienza cuando las botellas, el papel y cartón son introducidos manualmente por el usuario, una vez hecho esto y mediante el entrenamiento de la red neuronal hará una validación sobre el tipo de material a clasificar o mandara un error si no es dicho residuo como se muestra en la siguiente figura.

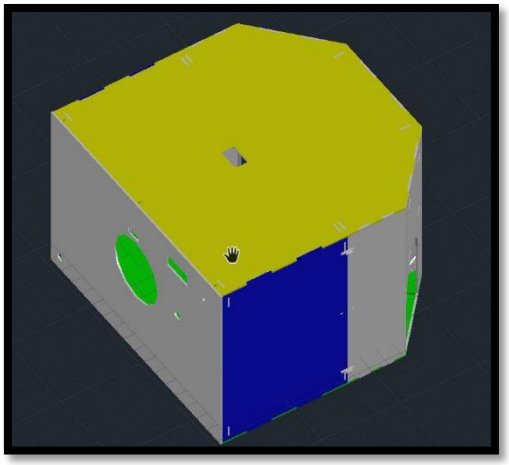


Figura 15. Prototipo AbsRecycler

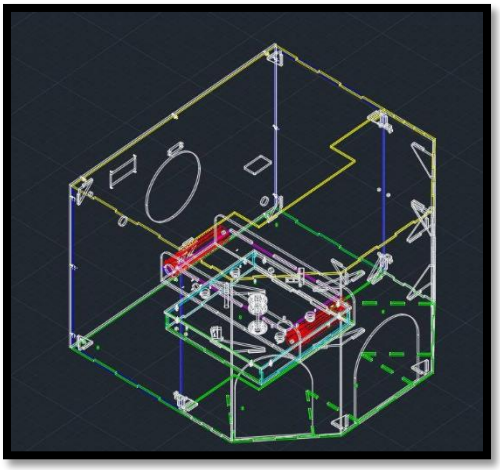


Figura 16. Prototipo AbsRecycler

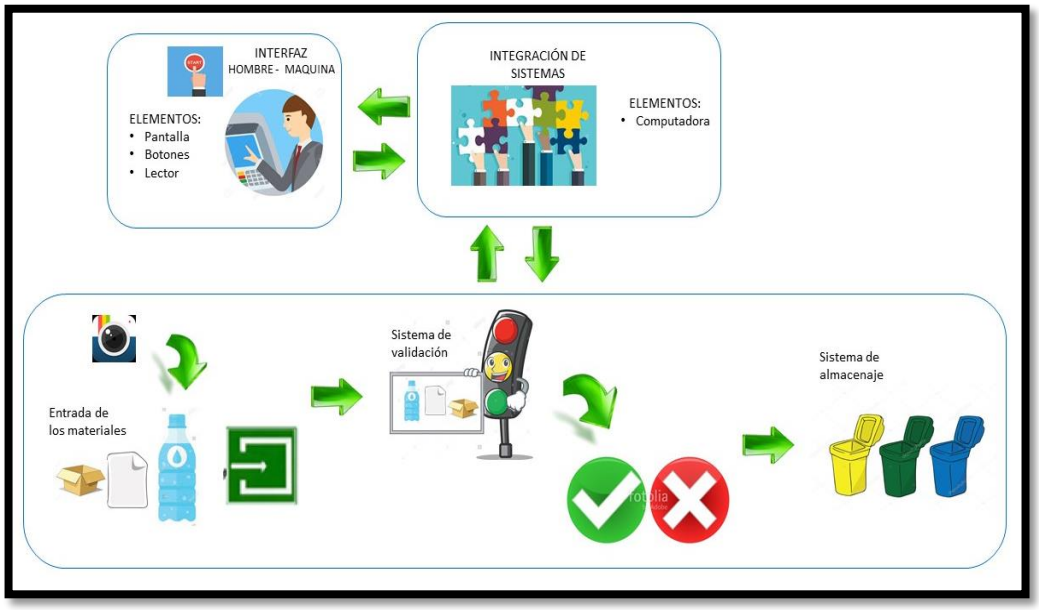


Figura 17. Diagrama de operación.

2.3.3. Metodología de desarrollo OOHDM

El modelo OOHDM u *Object Oriented Hypermedia Design Methodology,* para diseño de aplicaciones hipermedia y para la Web, fue diseñado por D. Schwabe, G. Rossi, and S. D. J. Barbosa y es una extensión de HDM con orientación a objetos, que se está convirtiendo en una de las metodologías más utilizadas. Ha sido usada para diseñar diferentes tipos de aplicaciones hipermedia como

galerías interactivas, presentaciones multimedia y, sobre todo, numerosos sitios web.

Al igual que RMM, este método se inspira en el modelo HDM, pero lo que le distingue claramente del primero es el proceso de concepción orientado a objetos. OOHDM propone el desarrollo de aplicaciones hipermedia mediante un proceso de 4 etapas:

- Diseño conceptual
- Diseño navegacional
- Diseño de interfaces abstractas
- Implementación

Cada etapa de la concepción define un esquema objeto específico en el que se introducen nuevos elementos (clases). En la primera etapa se construye un esquema conceptual representado por los objetos de dominio o clases y las relaciones entre dichos objetos. Se puede usar un modelo de datos semántico estructural (como el modelo de entidades y relaciones). La metodología OOHDM propone como esquema conceptual basado en clases, relaciones y subsistemas.

En la segunda etapa, el diseñador define clases navegacionales tales como: nodos, enlaces y estructuras de acceso (índices y visitas guiadas) inducidas del esquema conceptual.

La tercera etapa está dedicada a la especificación de la interfaz abstracta. Así, se define la forma en la cual deben aparecer los contextos navegacionales. También se incluye aquí el modo en que dichos objetos de interfaz activarán la navegación y el resto de funcionalidades de la aplicación, esto es, se describirán los objetos de interfaz y se los asociará con objetos de navegación. La separación entre el diseño navegacional y el diseño de interfaz abstracta permitirá construir diferentes interfaces para el mismo modelo navegacional.

Por fin, la cuarta etapa, dedicada a la puesta en práctica, es donde se hacen corresponder los objetos de interfaz con los objetos de implementación.

2.4. Desarrollo de la metodología OOHDM

Este documento es una especificación de requisitos para el desarrollo de lA aplicación que identifique pet, cartón y papel por medio de una red convolutiva. Se ha estructurado basándose en las directrices dadas por el estándar IEEE Práctica Recomendada para Especificaciones de Requisitos Software ANSI/IEEE 830.

2.4.1. Personal involucrado

Tabla 1. Personal involucrado.

Nombre	Aram Filemón Beciéz Salazar
Rol	Líder de proyecto
Categoría Profesional	Ing. Mecánico
Responsabilidad	Análisis de información, diseño y programación del SIS-I
Información de contacto	aramfbesa@hotmail.com

2.4.2. Requerimientos Funcionales

Tabla 2. Requerimientos funcionales

Identificación del requerimiento:	RF01
Nombre del Requerimiento:	Registro de materiales
Características:	Los materiales deberán registrarse para entrenar a la red neuronal.
Descripción del requerimiento:	El sistema podrá registrar y guardar los materiales como pet, cartón y papel para el entrenamiento de la red neuronal.
Requerimiento NO funcional:	• Seguridad: Garantiza la seguridad del sistema con respecto a la información y datos que se manejan.
Prioridad del requerimiento: Alta	

Tabla 3. Requerimientos funcionales

Identificación del requerimiento:	RF02
Nombre del Requerimiento:	Registrar Residuos Inorgánicos pet, papel y cartón.
Características:	El Administrador deberá registrar en el sistema la mayor cantidad de imágenes para que la identificación de los residuos se la óptima.
Descripción del requerimiento:	El sistema permitirá agregar imágenes para que sea eficiente el sistema en la identificación y clasificación como nombre, tipo de material, así como la cantidad.
Requerimiento NO funcional:	• RNF01 • RNF02 • RNF05 • RNF06
Prioridad del requerimiento: Alta	

Tabla 4. Requerimientos funcionales.

Identificación del requerimiento:	RF03
Nombre del Requerimiento:	Consultar Información.
Características:	El sistema ofrecerá la modificación de información registrada de acuerdo al entrenamiento de la red neuronal como estado en el que se encuentra los materiales, sucio o mojado.
Descripción del requerimiento:	**Consultar Instrucción:** Permite modificar la información registrada por parte del administrador
Requerimiento NO funcional:	• RNF01 • RNF02
Prioridad del requerimiento: Alta	

Tabla 5. Requerimientos funcionales.

Identificación del requerimiento:	RF04

Nombre del Requerimiento:	Modificar.
Características:	El sistema permitirá al administrador modificar la información registrada anteriormente
Descripción del requerimiento:	Permite al administrador modificar la información para el eficaz entrenamiento de la red neuronal
Requerimiento NO funcional:	• RNF01 • RNF02 • RNF05
Prioridad del requerimiento: Alta	

Tabla 6. Requerimientos funcionales.

Identificación del requerimiento:	RF05
Nombre del Requerimiento:	Eliminar Información.
Características:	El administrador podrá eliminar la información de los materiales inorgánicos
Descripción del requerimiento:	El Administrador podrá eliminar información de los residuos inorgánicos
Requerimiento NO funcional:	● RNF01 ● RNF02
Prioridad del requerimiento: Alta	

Tabla 7. Requerimientos funcionales.

Identificación del requerimiento:	RF06
Nombre del Requerimiento:	Consultar Información.
Características:	El sistema ofrecerá el reconocimiento y clasificación de los residuos según el tipo de material.
Descripción del requerimiento:	**Consultar información:** identificación y clasificación a los residuos inorgánicos como pet, papel y cartón.

Requerimiento NO funcional:	• RNF01 • RNF02
Prioridad del requerimiento: Alta	

Tabla 8. Requerimientos funcionales.

Identificación del requerimiento:	RF07
Nombre del Requerimiento:	Visualizar Residuos
Características:	Permite visualizar información referente a los materiales de mayor clasificación.
Descripción del requerimiento:	**Visualizar Residuos:** permite al administrador verificar la información de la red neuronal entrenada.
Requerimiento NO funcional:	• La información deberá ser mostrada en orden de las siguientes categorías; tipo de material, y cantidad.
Prioridad del requerimiento: Alta	

2.4.3. Requerimientos No Funcionales.

Tabla 9. Requerimientos no funcionales.

Identificación del requerimiento:	RNF01
Nombre del Requerimiento:	Interfaz del sistema.
Características:	El sistema presentara una interfaz de usuario sencilla para que sea de fácil manejo a los usuarios del sistema.
Descripción del requerimiento:	El sistema debe tener una interfaz de uso intuitiva y sencilla.
Prioridad del requerimiento: Alta	

Tabla 10. Requerimientos no funcionales.

Identificación del requerimiento:	RNF02
Nombre del Requerimiento:	Ayuda en el uso del sistema.
Características:	La interfaz del usuario deberá de presentar un sistema de ayuda para que los mismos usuarios del sistema se les faciliten el trabajo en cuanto al manejo del sistema.
Descripción del requerimiento:	La interfaz debe estar complementada con un buen sistema de ayuda (la administración puede recaer en personal con poca experiencia en el uso de aplicaciones informáticas).
Prioridad del requerimiento: Alta	

Tabla 11. Requerimientos no funcionales.

Identificación del requerimiento:	RNF03
Nombre del Requerimiento:	Mantenimiento.
Características:	El sistema deberá de tener un manual de instalación y manual de usuario para facilitar los mantenimientos que serán realizados por el administrador.
Descripción del requerimiento:	El sistema debe disponer de una documentación fácilmente actualizable que permita realizar operaciones de mantenimiento con el menor esfuerzo posible.
Prioridad del requerimiento: Alta	

Tabla 12. Requerimientos no funcionales.

Identificación del requerimiento:	RNF04
Nombre del Requerimiento:	Diseño de la interfaz a la característica de la web.
Características:	El sistema deberá de tener una interfaz de usuario, teniendo en cuenta las características de la web.

Descripción del requerimiento:	La interfaz de usuario debe ajustarse a las características de la web de la aplicación.
Prioridad del requerimiento: Alta	

Tabla 13. Requerimientos no funcionales.

Identificación del requerimiento:	RNF05
Nombre del Requerimiento:	Desempeño
Características:	El sistema garantizara a los usuarios un desempeño en cuanto a los datos almacenado en el sistema ofreciéndole una confiabilidad a esta misma.
Descripción del requerimiento:	Garantizar el desempeño del sistema informático a los diferentes usuarios. En este sentido la información almacenada o registros realizados podrán ser consultados y actualizados permanente y simultáneamente, sin que se afecte el tiempo de respuesta.
Prioridad del requerimiento: Alta	

Tabla 14. Requerimientos no funcionales.

Identificación del requerimiento:	RNF06
Nombre del Requerimiento:	Nivel de Usuario
Características:	Garantizara al usuario el acceso de información de acuerdo al nivel que posee.
Descripción del requerimiento:	Facilidades y controles para permitir el acceso a la información al personal autorizado a través de Internet, con la intención de consultar y subir información pertinente para cada una de ellas.
Prioridad del requerimiento: Alta	

Tabla 15. Requerimientos no funcionales.

Identificación del requerimiento:	RNF07
Nombre del Requerimiento:	Confiabilidad continúa del sistema.
Características:	El sistema tendrá que estar en funcionamiento las 24 horas los 7 días de la semana. Ya que es una página web diseñada para la carga de datos y comunicación entre usuarios.
Descripción del requerimiento:	La disponibilidad del sistema debe ser continua con un nivel de servicio para los usuarios de 7 días por 24 horas, garantizando un esquema adecuado que permita la posible falla en cualquiera de sus componentes, contar con una contingencia, generación de alarmas.
Prioridad del requerimiento: Alta	

Tabla 16. Requerimientos no funcionales.

Identificación del requerimiento:	RNF08
Nombre del Requerimiento:	Seguridad en información
Características:	El sistema garantizara a los usuarios una seguridad en cuanto a la información que se procede en el sistema.
Descripción del requerimiento:	Garantizar la seguridad del sistema con respecto a la información y datos.
Prioridad del requerimiento: Alta	

2.5. Diagrama de casos de uso

Casos de uso

El usuario:

-Usuario: introduce materiales.

-Recicladora: manda información al usuario.

-Recicladora: Valida material. Espera respuesta del sistema.

-Recicladora: informa al usuario que el material es valido

-Recicladora: clasifica material. *Ver imagen 18*

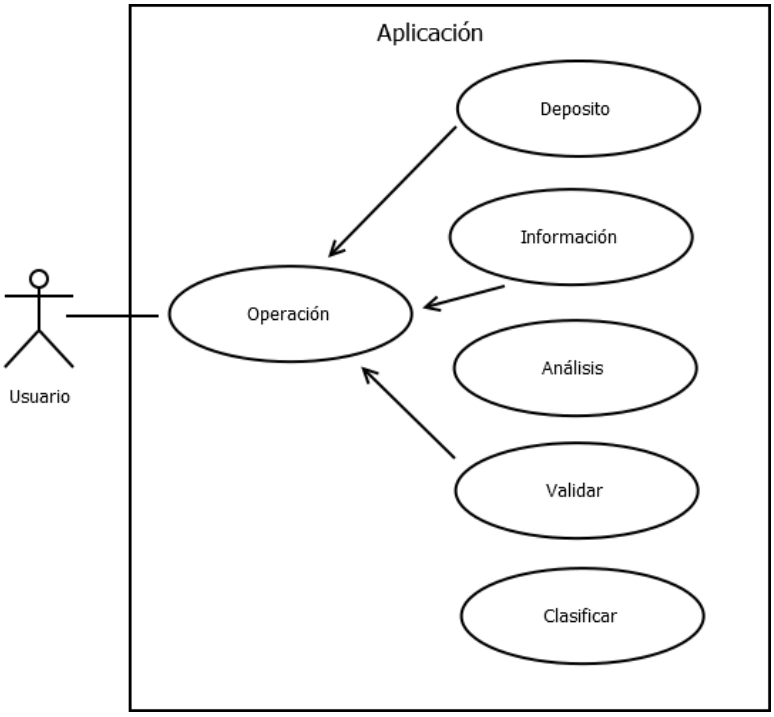


Figura 18. Diagrama casos de uso.

2.6. Desarrollo del entrenamiento de CNN para la identificación y clasificación de materiales pet, papel y cartón.

- Instalación plataforma Python y librerías
- Proceso de entrenamiento
- Conectando la CNN con la placa de Arduino

2.6.1. Instalación Plataforma Python y Librerías

Para el proceso de entrenamiento de la red neuronal, se procede a la instalación de la plataforma de Python, así como las principales librerías para Deep Learning, el cual es un tipo de machine learning que entrena a una computadora para que realice tareas como las que realizan los seres humanos, como el reconocimiento del habla, la identificación de imágenes o hacer predicciones. En lugar de organizar datos para que se ejecuten a través de ecuaciones predefinidas, el

deep learning configura parámetros básicos acerca de los datos y entrena a la computadora para que aprenda por cuenta propia reconociendo patrones mediante el uso de varias capas de procesamiento.

Así una vez instalado el intérprete de Python, se procede a instalar las librerías como matplotlib, numpy y pandas, estas son utilizadas como las principales librerías de machine learning y deep learning, estas proporcionan herramientas útiles para análisis de datos, posteriormente se instalan la principales librerías de deep learning para Python que es tensorflow y keras, todo esto al ser gran cantidad de software para instalar, de modo que se llevará a cabo en una distribución de Python denomida Anaconda, aquí ya viene instalado el paquete de librerías como matplotlib, numpy, pandas, jupyter, spyder, y para instalar el gestor de paquetes de tensorflow y keras se utiliza el entorno de Conda, y por último se instala OpenCv, todo esto se realiza en el cmd del sistema junto con todo el gestor de paquetes pip que se encuentra en el siguiente enlace (Python.org, Portal python-opencv, 2020). *Ver figura 19 y 20.*

Figura 20. Librerías de Python

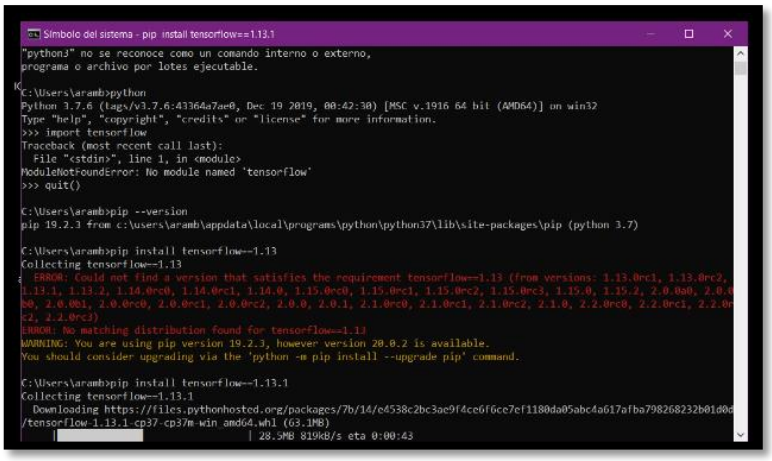


Figura 21. Instalación de Python y sus librerías

De esta manera, al describir el código en el entrenamiento de la red neuronal, Anaconda cuenta con una aplicación de escritorio que se llama Anaconda Navigator, la cual permite gestionar paquetes y ejecutar aplicaciones, por lo que instalamos VS code para empezar a desarrollar la aplicación. *Ver figura 22*

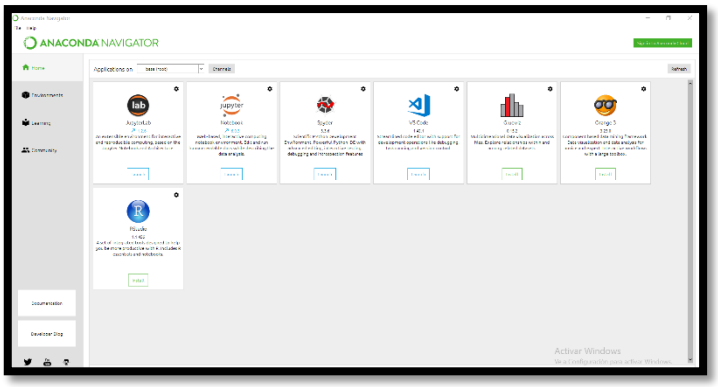


Figura 22. Gestor de paquetes Anaconda-Python

Al importar las librerías de tensorflow, keras y opencv, y comenzando con el entrenamiento describiré el proceso para entrenar una red convolutiva para obtener la predicción de los materiales, de esta manera tenemos el siguiente código: *Ver figura 23*

```
# ---------------------- SUPPRESS WARNINGS ------------------------------ #
import logging
from os import environ
import warnings
warnings.filterwarnings('ignore', category=FutureWarning)
environ['TF_CPP_MIN_LOG_LEVEL'] = '3'  # FATAL
logging.getLogger('tensorflow').setLevel(logging.FATAL)
# ----------------------------------------------------------------------- #
# IMPORT MODELS
from keras_squeezenet_model import SqueezeNet
from keras.models import Model, model_from_json
# IMPORT LAYERS
from keras.layers import *
# IMPORT OPTIMIZERS
from keras.optimizers import Adadelta
# IMPORT OTHER PACKAGES
from keras.preprocessing.image import ImageDataGenerator
from keras.callbacks import ModelCheckpoint, EarlyStopping
import numpy as np
from sklearn.metrics import classification_report
import json
import os
import sklearn.utils
```

Figura 23. Librerías de Python importadas

2.6.2. Proceso de entrenamiento

De los materiales a identificar por medio de la red neuronal, se toma fotos de cada uno de ellos por medio de una cámara web de la marca Logitech modelo c270, una vez armado el prototipo, la cámara se coloca en la parte superior a una distancia considerable para que la toma de imágenes se ha lo más completa al material, cada material debe ser rotado en diferentes ángulos y posiciones, pues será de gran importancia al momento de que la CNN pase cada imagen para distinguir formas y comience aprender los diferentes tipos de imágenes que serán almacenadas en una carpeta, para que el resultado sea muy eficiente se considera tomar el mayor número de fotos como se ilustra en las siguientes imágenes. *Ver figura 24, 24.1, 24.2, 24.3 y 24.4.*

Figura 24. Prototipo y cámara logitech

Figura 24.1 Toma de fotografías.

Figura 24.2 Posición de material (papel).

Figura 24.3 Posición de material (pet).

Figura 24.4 Posición de material (cartón).

Posteriormente, para comenzar se generan carpetas con los nombres deploy y training, dentro de la carpeta data se generan las carpetas entrenamiento y validación, dentro de la primera, se generan cuatro carpetas con los nombres nada, pet, papel y cartón, así dentro de cada una de ellas se tienen las imágenes para entrenar a la red neuronal, por lo que en la carpeta de validación se genera la misma estructura con tres carpetas con los mismos nombres y dentro de ellas las imágenes, que servirán para el entrenamiento, cuya finalidad será, que, al leer una imagen, el sistema una vez entrenado, mandará como respuesta si es un material de tipo pet, papel o cartón. *Ver figura 25 y 25.1.*

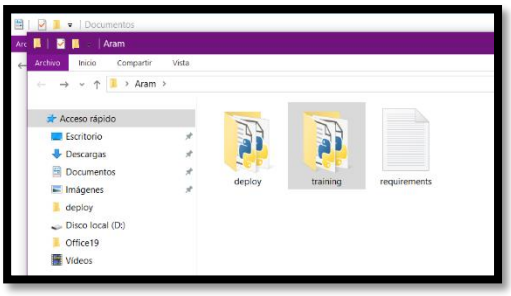


Figura 25. Carpetas deploy y training

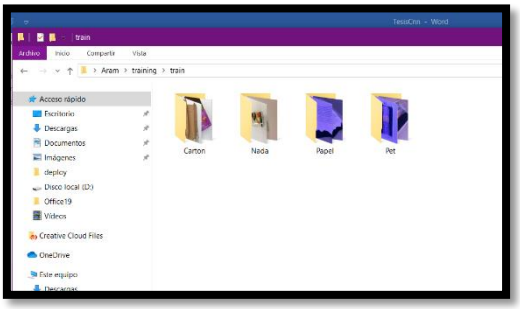


Figura 25.1 Data set de imágenes.

Ahora el nombre de la red llamada SqueezeNet, carga los pesos weights='imagnet', con la finalidad de entrenar la red con las imágenes una vez tomadas, tomando como base la red keras_squeeznet la cual es una red liviana que tiene pocos parámetros y es muy eficiente en el desarrollo del entrenamiento extrayendo características de las imágenes, ahora lo que se va ejecutar es que se toman los parámetros de esta red, le quita el final de esa red en este caso

neuronas y le pone neuronas entrenables que son para que aprenda nuestra clase, construyendo unas neuronas por encima y a partir de la layer 23 se comienza a ejecutar. *Ver figura 26.*

```
def fine_tune_sqnet(height, width, channels, n_classes):

    base = SqueezeNet(include_top=False, weights='imagenet', pooling='avg', input_shape=(height, width, channels))

    for layer in base.layers[:23]:
        layer.trainable = False

    out = base.output

    out = Dense(100, name="hidden")(out)
    out = Activation('relu', name='hidden_relu')(out)
    out = Dropout(0.5, name='dropout')(out)
    out = Dense(n_classes, name='classifier')(out)
    out = Activation('softmax', name='softmax_classifier')(out)

    model = Model(inputs=base.input, outputs=out, name='squeezenet')

    return model


def train(name, epochs, train_data_dir, validation_data_dir,
          batch_size, width, height, n_channels, weighted):

    seed = 911

    train_datagen = ImageDataGenerator(
        shear_range=0.2,
        brightness_range=[0.5, 1.5],
```

Figura 26. Red neuronal cargando pesos de las imágenes.

Ahora la función principal train, va realizar una serie de ejecuciones aleatorias en las imágenes para que el modelo salga menos propenso a errores, en donde tuerce la imagen, le cambia el brillo, le hace zoom, cambia la posición, el resultado es que el entramiento la ve distinta aprendiendo más el concepto y no tanto solo una imagen en específico, por consiguiente, carga los datos, correr los mismos por la red neuronal. *Ver figura 27*

Figura 27. Cargando datos de las imágenes.

En el siguiente script se denomina con el nombre aram_squeezenet_v2, posteriormente designamos las épocas para lo cual designamos 100, en este valor no tenemos que preocuparnos por que tenemos la función llamada EarlynStopping, el cual tiene una paciencia de 10, la cual podemos cambiar, funcionando de la siguiente manera verifica el entrenamiento en donde al ver que han pasado 10 epocas el entrenamiento y ve que no mejoro, entonces para el entrenamiento y recarga el mejor modelo entrenado, así que puede terminar en la 30, 50 dependiendo de la eficiencia en el proceso de mejora el performace. Así de esta manera se guardan los modelos con arquitectura .jason, así por ultimo ejecutamos el código para ver el resultado del entrenamiento, indicamos en las direcciones de las carpetas llamadas train y test, lo que va hacer el modelo es ver lo que está en la carpeta train para aprender y lo que está en test es va ver cómo va aprendiendo el modelo, es evaluar sobre lo que el modelo no vio antes para obtener resultados de performace, entramos en cmd del sistema para ejecutar el entrenamiento, obteniendo lo siguiente: *Ver figura 28*

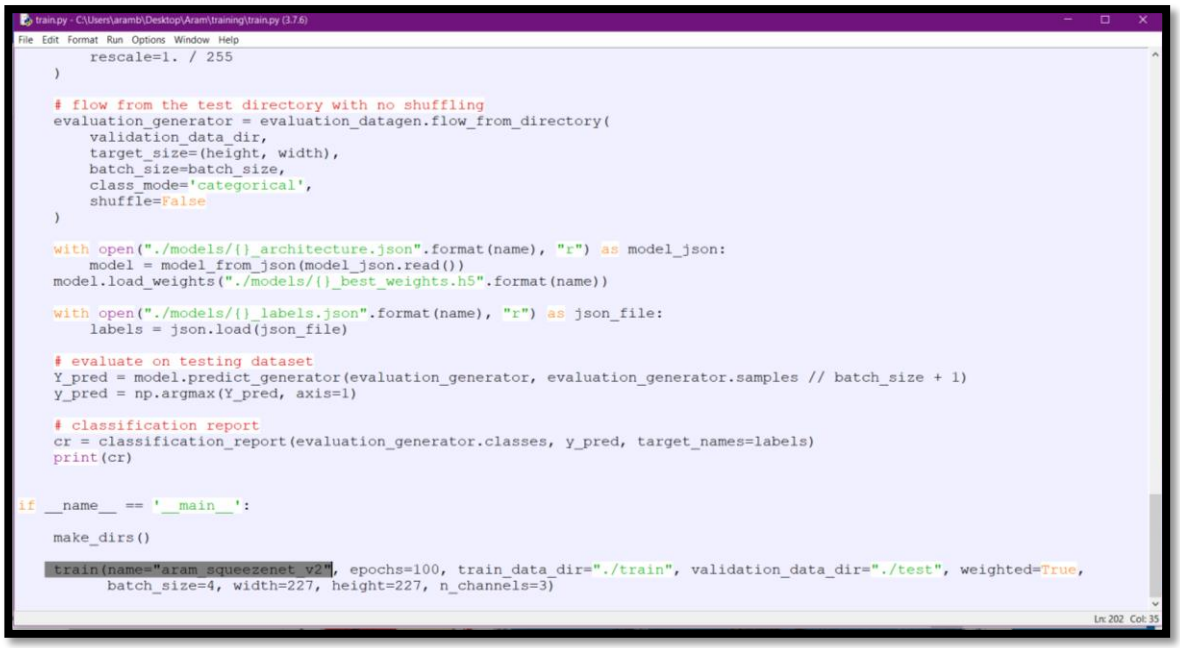


Figura 28 Asignado nombre y épocas para correr el entrenamiento.

Descarga los pesos de la red. *Ver figura 29*

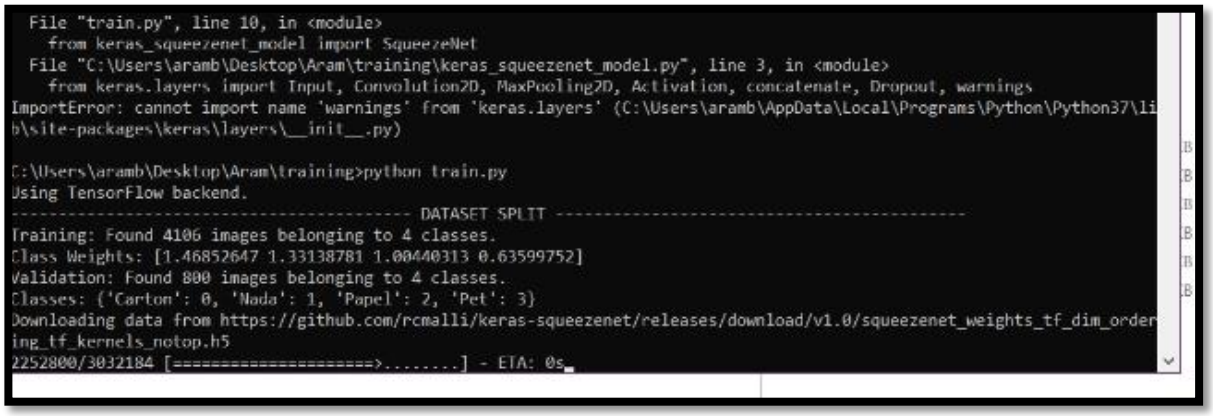


Figura 29. Descargando pesos de la red

Al principio del capítulo se comentó que se realizan dos directorios con los nombres deploy y training, en la primer carpeta se va generar mediante el entrenamiento 4 archivos en formato json, en el archivo con el nombre aram_squeezenet_label tenemos una lista con los 4 elementos (“Nada”, “Carton”, ”Papel”, ”Pet”) con los cuales fue entrenada la red neuronal en este caso tenemos 4 salidas 0,1, 2 y 3 correspondientes a cada elemento, posteriormente en el

archivo aram_squeezenet_arquitecture se obtiene los datos de cómo está construida la red neuronal en keras. *Ver figura 30*

Figura 30 Construcción de la Red Neuronal.

En el archivo aram_squeezenet_best_weights.h5, en el cual se generan todos los pesos durante el proceso de entrenamiento, y en el archivo aram_squeezenet_frozen.pb se tiene un modelo congelado que solamente sirve para predecir, la ventaja es que lo podemos abrir con opencv prediciendo más rápido en cpu, por tanto se puede predecir cargando la arquitectura y los pesos con keras o en su defecto puede cargar directamente el modelo .pb de tensorflow en opencv para predecir. *Ver figura 31.*

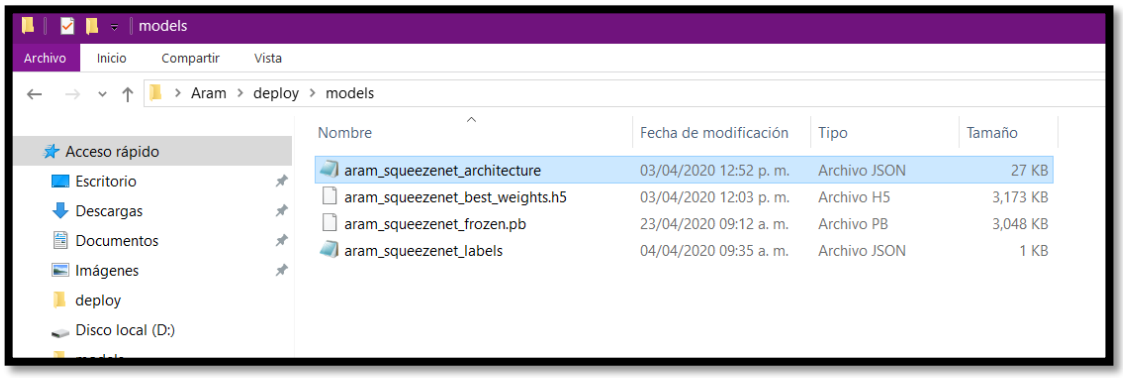


Figura 31. Obtención del modelo entrenado para predecir.

Por lo que se explica los dos modelos de la siguiente manera: *Ver figura 32*

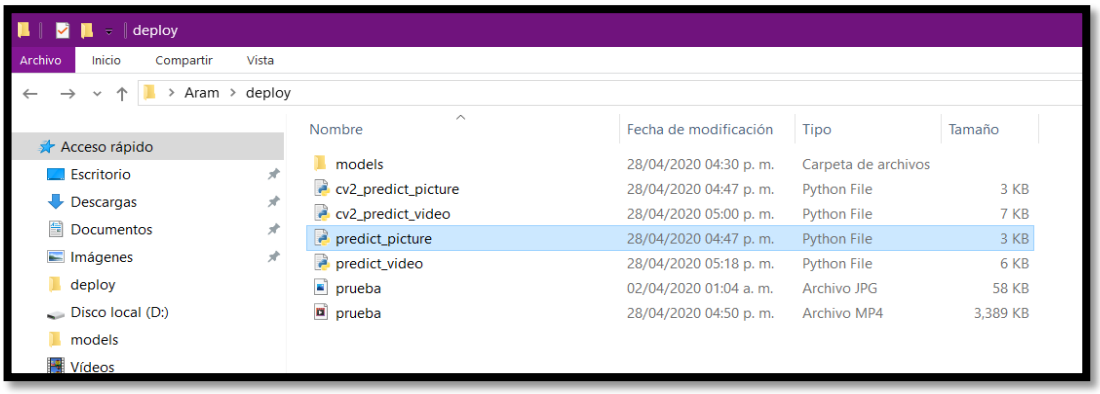


Figura 32. Modelos openc_cv y tensorflow

En este scrip se usa la predicción con el modelo de keras, aquí se le pasa la imagen 54ython predict_picture.py -i “./prueba.jpg”, en donde el modelo esta entrenado con imagen de (227,227), se tiene el nombre del modelo models/aram_squezeenet_architecture.json, carga los pesos best_weight.h5, así como la arquitectura del archivo aram_squeezenet_architecture.json, y las etiquetas de cada categoría, por ultimo ejecutamos en el cmd del sistema y tenemos lo siguiente: *Ver figura 33*

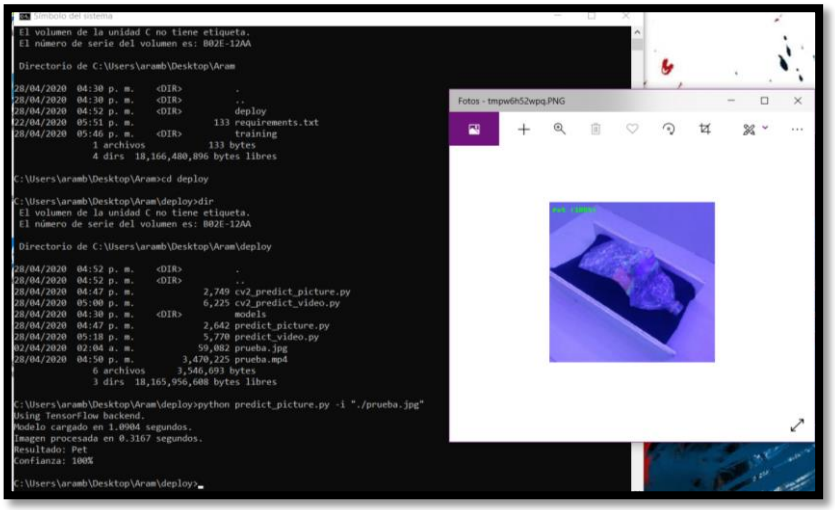


Figura 33 Ejecución del modelo entrenado con tensorflow.

Ahora se ejecuta el archivo cv2_predict_picture y video, el cual ocupa OpenCv y el archivo frozen.pb y mostraremos que resultados se obtienen en tiempos de predicción, en donde podemos ver que cv2_predict el cual ocupa OpenCv es más optimizado, el modelo es cargado con un tiempo de .0174s, imagen procesada en .0357s con respeto al modelo de keras con un tiempo de 1.0904s, imagen procesada en .3167, el resultado es mismo pero con tiempos diferentes. *Ver figura 34, 34.1, 34.2 y 34.3.*

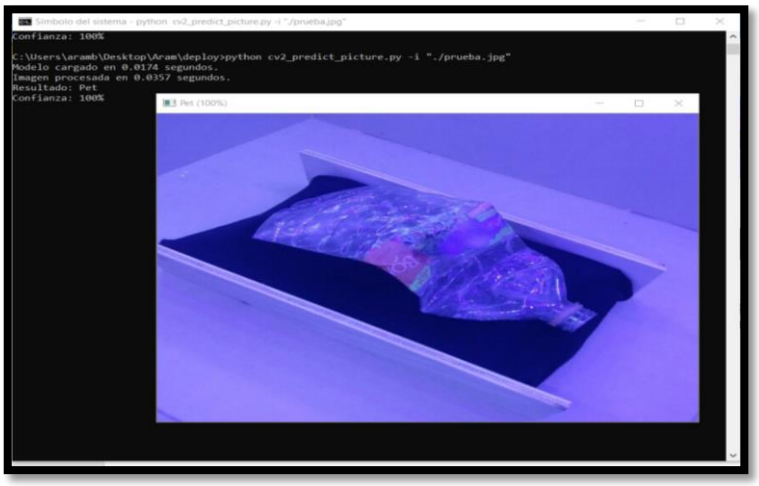


Figura 34 Ejecución modelo entrenado con OpenCV.

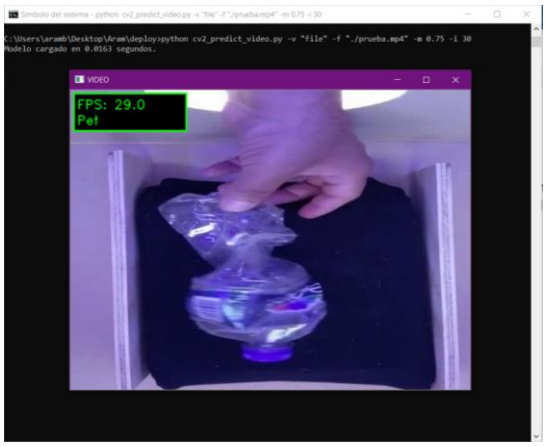


Figura 34.1 Ejecución modelo entrenado con OpenCV.

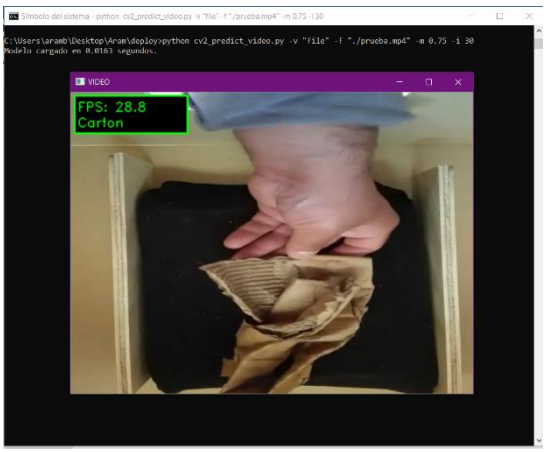


Figura 34.2 Ejecución modelo entrenado con OpenCV.

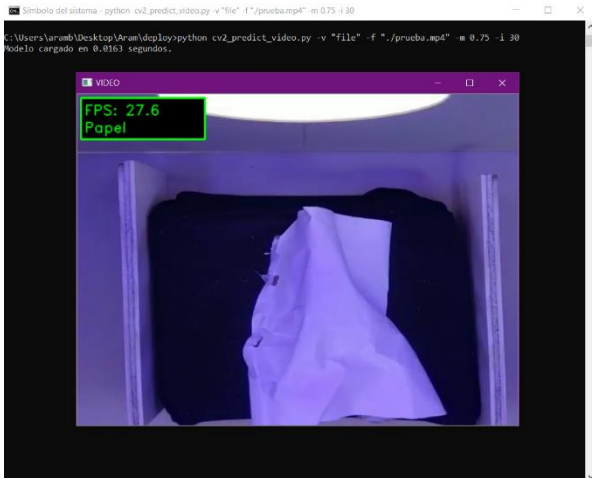


Figura 34.3 Ejecución modelo entrenado con OpenCV.

2.6.3. Conectando la CNN con la placa de Arduino

Una vez que se ha entrenado la red neuronal, que identifica los materiales, lo que a continuación se procede a realizar, es la conexión con la placa de Arduino con la librería using Python to control an Arduino, cuya principal función será la de mandar una señal, cuando al introducir un material de tipo pet, papel o cartón, serán enviados a un contenedor correspondiente, por lo que son relacionados como especie de diccionario identificados por una letra en específico donde a es cartón, b = papel y c = pet, por lo que al momento de predecir una imagen esta mandara una señal ya designada, así la banda transportadora se activara llevando a cada material a su lugar correspondiente. *Ver figura 35, 35.1 y 35.2.*

```
# USO: python cv2_predict_video.py -v "webcam" -m 0.75 -
import cv2
import numpy as np
from collections import deque
from time import time, sleep
import json
import argparse
from imutils.video import VideoStream
from serial import Serial
from time import time

class Commands:
    def __init__(self, port="COM3"):

        self.label2command = {
            "Carton": b"a",
            "Papel": b"b",
            "Pet": b"c"
            }

        self.wait = 8.0
```

Figura 35 Conexión con la placa Arduino

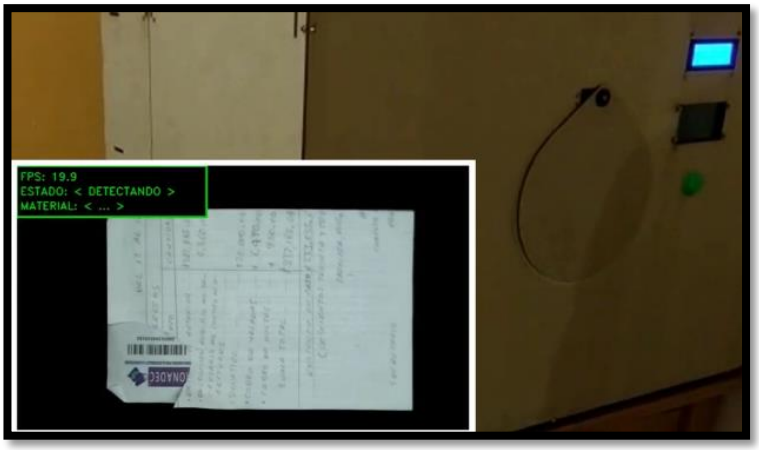


Figura 35.1. Activación banda transportadora

Figura 35.2. Activación de banda transportadora.

CAPÍTULO III

IMPLEMENTACIÓN Y PRUEBAS

3.1. Análisis de Datos

Una vez que se descargaron los pesos de la red, y concluyo el tiempo de entrenamiento se muestra el resultado en el siguiente análisis:

---------- DATASET SPLIT ----------

El resultado fue encontró 4106 imágenes de 4 clases

Training: Found 4106 images belonging to 4 classes.

Tenemos los pesos de cada clase, las que aparecen más, tienen un número menor. *Ver figura 36*

Figura 36 Pesos de las clases.

Para validación, encontró 800 imágenes, 200 en cada carpeta

Validation: Found 800 images belonging to 4 classes.

Classes: {'Ausencia': 0, 'Carton': 1, 'Papel': 2, 'Pet': 3}

Se muestra la estructura de la red neuronal con todas sus capas que la conforman.

---------- MODEL SUMMARY ----------

Layer (type)	Output Shape	Param #	Connected to
input_1 (InputLayer)	(None, 227, 227, 3)	0	
conv1 (Conv2D)	(None, 113, 113, 64)	1792	input_1[0][0]
relu_conv1 (Activation)	(None, 113, 113, 64)	0	conv1[0][0]
pool1 (MaxPooling2D)	(None, 56, 56, 64)	0	relu_conv1[0][0]

fire2/squeeze1x1 (Conv2D)	(None, 56, 56, 16)	1040	pool1[0][0]
fire2/relu_squeeze1x1 (Activation)	(None, 56, 56, 16)	0	fire2/squeeze1x1[0][0]
fire2/expand1x1 (Conv2D)	(None, 56, 56, 64)	1088	fire2/relu_squeeze1x1[0][0]
fire2/expand3x3 (Conv2D)	(None, 56, 56, 64)	9280	fire2/relu_squeeze1x1[0][0]
fire2/relu_expand1x1 (Activation)	(None, 56, 56, 64)	0	fire2/expand1x1[0][0]
fire2/relu_expand3x3 (Activation)	(None, 56, 56, 64)	0	fire2/expand3x3[0][0]
fire2/concat (Concatenate)	(None, 56, 56, 128)	0	fire2/relu_expand1x1[0][0] fire2/relu_expand3x3[0][0]
fire3/squeeze1x1 (Conv2D)	(None, 56, 56, 16)	2064	fire2/concat[0][0]
fire3/relu_squeeze1x1 (Activation)	(None, 56, 56, 16)	0	fire3/squeeze1x1[0][0]
fire3/expand1x1 (Conv2D)	(None, 56, 56, 64)	1088	fire3/relu_squeeze1x1[0][0]
fire3/expand3x3 (Conv2D)	(None, 56, 56, 64)	9280	fire3/relu_squeeze1x1[0][0]
fire3/relu_expand1x1 (Activation)	(None, 56, 56, 64)	0	fire3/expand1x1[0][0]
fire3/relu_expand3x3 (Activation)	(None, 56, 56, 64)	0	fire3/expand3x3[0][0]
fire3/concat (Concatenate)	(None, 56, 56, 128)	0	fire3/relu_expand1x1[0][0] fire3/relu_expand3x3[0][0]
pool3 (MaxPooling2D)	(None, 27, 27, 128)	0	fire3/concat[0][0]
fire4/squeeze1x1 (Conv2D)	(None, 27, 27, 32)	4128	pool3[0][0]
fire4/relu_squeeze1x1 (Activation)	(None, 27, 27, 32)	0	fire4/squeeze1x1[0][0]

fire4/expand1x1 (Conv2D)	(None, 27, 27, 128) 4224	fire4/relu_squeeze1x1[0][0]
fire4/expand3x3 (Conv2D)	(None, 27, 27, 128) 36992	fire4/relu_squeeze1x1[0][0]
fire4/relu_expand1x1 (Activation)	(None, 27, 27, 128) 0	fire4/expand1x1[0][0]
fire4/relu_expand3x3 (Activation)	(None, 27, 27, 128) 0	fire4/expand3x3[0][0]
fire4/concat (Concatenate)	(None, 27, 27, 256) 0	fire4/relu_expand1x1[0][0] fire4/relu_expand3x3[0][0]
fire5/squeeze1x1 (Conv2D)	(None, 27, 27, 32) 8224	fire4/concat[0][0]
fire5/relu_squeeze1x1 (Activation)	(None, 27, 27, 32) 0	fire5/squeeze1x1[0][0]
fire5/expand1x1 (Conv2D)	(None, 27, 27, 128) 4224	fire5/relu_squeeze1x1[0][0]
fire5/expand3x3 (Conv2D)	(None, 27, 27, 128) 36992	fire5/relu_squeeze1x1[0][0]
fire5/relu_expand1x1 (Activation)	(None, 27, 27, 128) 0	fire5/expand1x1[0][0]
fire5/relu_expand3x3 (Activation)	(None, 27, 27, 128) 0	fire5/expand3x3[0][0]
fire5/concat (Concatenate)	(None, 27, 27, 256) 0	fire5/relu_expand1x1[0][0] fire5/relu_expand3x3[0][0]
pool5 (MaxPooling2D)	(None, 13, 13, 256) 0	fire5/concat[0][0]
fire6/squeeze1x1 (Conv2D)	(None, 13, 13, 48) 12336	pool5[0][0]
fire6/relu_squeeze1x1 (Activation)	(None, 13, 13, 48) 0	fire6/squeeze1x1[0][0]
fire6/expand1x1 (Conv2D)	(None, 13, 13, 192) 9408	fire6/relu_squeeze1x1[0][0]
fire6/expand3x3 (Conv2D)	(None, 13, 13, 192) 83136	fire6/relu_squeeze1x1[0][0]
fire6/relu_expand1x1 (Activation)	(None, 13, 13, 192) 0	fire6/expand1x1[0][0]
fire6/relu_expand3x3 (Activation)	(None, 13, 13, 192) 0	fire6/expand3x3[0][0]
fire6/concat (Concatenate)	(None, 13, 13, 384) 0	fire6/relu_expand1x1[0][0]

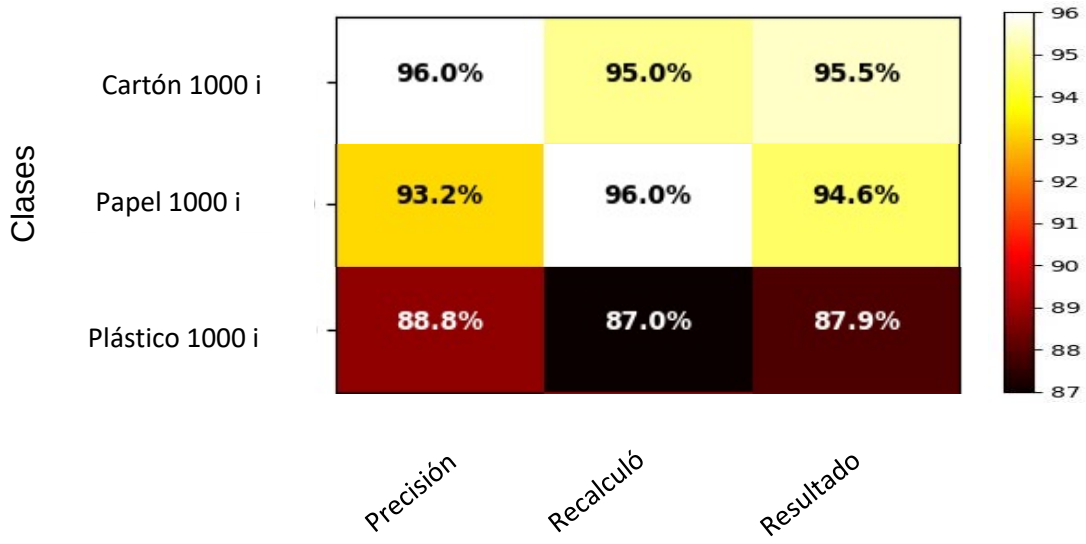
			fire6/relu_expand3x3[0][0]
fire7/squeeze1x1 (Conv2D)	(None, 13, 13, 48)	18480	fire6/concat[0][0]
fire7/relu_squeeze1x1 (Activation)	(None, 13, 13, 48)	0	fire7/squeeze1x1[0][0]
fire7/expand1x1 (Conv2D)	(None, 13, 13, 192)	9408	fire7/relu_squeeze1x1[0][0]
fire7/expand3x3 (Conv2D)	(None, 13, 13, 192)	83136	fire7/relu_squeeze1x1[0][0]
fire7/relu_expand1x1 (Activation)	(None, 13, 13, 192)	0	fire7/expand1x1[0][0]
fire7/relu_expand3x3 (Activation)	(None, 13, 13, 192)	0	fire7/expand3x3[0][0]
fire7/concat (Concatenate)	(None, 13, 13, 384)	0	fire7/relu_expand1x1[0][0] fire7/relu_expand3x3[0][0]
fire8/squeeze1x1 (Conv2D)	(None, 13, 13, 64)	24640	fire7/concat[0][0]
fire8/relu_squeeze1x1 (Activation)	(None, 13, 13, 64)	0	fire8/squeeze1x1[0][0]
fire8/expand1x1 (Conv2D)	(None, 13, 13, 256)	16640	fire8/relu_squeeze1x1[0][0]
fire8/expand3x3 (Conv2D)	(None, 13, 13, 256)	147712	fire8/relu_squeeze1x1[0][0]
fire8/relu_expand1x1 (Activation)	(None, 13, 13, 256)	0	fire8/expand1x1[0][0]
fire8/relu_expand3x3 (Activation)	(None, 13, 13, 256)	0	fire8/expand3x3[0][0]
fire8/concat (Concatenate)	(None, 13, 13, 512)	0	fire8/relu_expand1x1[0][0] fire8/relu_expand3x3[0][0]
fire9/squeeze1x1 (Conv2D)	(None, 13, 13, 64)	32832	fire8/concat[0][0]
fire9/relu_squeeze1x1 (Activation)	(None, 13, 13, 64)	0	fire9/squeeze1x1[0][0]
fire9/expand1x1 (Conv2D)	(None, 13, 13, 256)	16640	fire9/relu_squeeze1x1[0][0]
fire9/expand3x3 (Conv2D)	(None, 13, 13, 256)	147712	fire9/relu_squeeze1x1[0][0]
fire9/relu_expand1x1 (Activation)	(None, 13, 13, 256)	0	fire9/expand1x1[0][0]

```
__________________________________________________________________________________________
__
fire9/relu_expand3x3 (Activation)          (None, 13, 13, 256)  0                                    fire9/expand3x3[0][0]
__________________________________________________________________________________________
__
fire9/concat (Concatenate)                 (None, 13, 13, 512)  0                                    fire9/relu_expand1x1[0][0]
                                                                                                     fire9/relu_expand3x3[0][0]
__________________________________________________________________________________________
__
global_average_pooling2d_1                 (Glo (None, 512)       0                                   fire9/concat[0][0]
__________________________________________________________________________________________
__
hidden_1 (Dense)                           (None, 100)          51300                               global_average_pooling2d_1[0][0]
__________________________________________________________________________________________
__
hidden_1_relu (Activation)                 (None, 100)          0                                   hidden_1[0][0]
__________________________________________________________________________________________
__
dropout (Dropout)                          (None, 100)          0                                   hidden_1_relu[0][0]
__________________________________________________________________________________________
__
classifier (Dense)                         (None, 4)            404                                 dropout[0][0]
__________________________________________________________________________________________
__
softmax_classifier (Activation) (None, 4)          0           classifier[0][0]
==========================================================================================
=======
Aquí se muestra el total de parámetros que se entrena
Total params: 774,200
Trainable params: 703,224 parámetros entrenables
Non-trainable params: 70,976 de los cuales dejamos congelados
__________________________________________________________________________________________
__
```

3.2. Selección de pruebas estadísticas

Mediante el resultado del entrenamiento se obtiene la siguiente tabla en la cual describe la eficiencia para la predicción de los materiales, cuyos valores son bastante aceptables, en promedio se tiene el 90% de efectividad para cumplir el objetivo.

Tabla 17 Resultado del entrenamiento Red Neuronal



3.3. Realización del Análisis

Durante el entrenamiento va recorriendo todas las imágenes, en este caso encontró 4106 imágenes de 4 clases, arrojando los pesos en cada una de ellas, cuyo valor , es importante para obtener un mayor performace, así, en cada época va recorriendo las cuatro mil imágenes que identifico, por tanto, en la primer época el resultado se identifica por medio de: val_accurary: 0.6525, y en el val_loss:0.8532 este es el error que encuentra y tiene que ir bajando a medida que va recorriendo el número de épocas por consiguiente al disminuir la predicción será mucho más eficiente, mediante este proceso tiene que disminuir conforme corre el entramiento como ya se mencionó y cuando baja por un lapso de mayor tiempo es cuando para el entrenamiento y arroja los mejores resultados, así para cuando ha llegado a un punto más bajo y el val_accurary sube, alcanzando un valor de 0.9125 en su punto máximo de eficiencia y en val_loss un valor 0.0490, de esta manera se obtiene un modelo por arriba del 90 % de efectivas para detectar he identificar los materiales, un valor bastante aceptable en la predicción. *Ver figura 37 y 37.1*

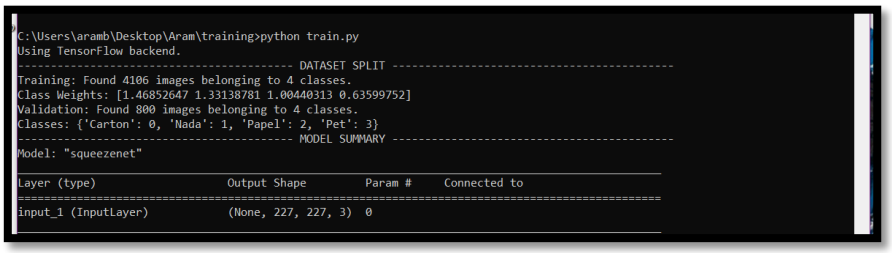


Figura 37 Eficiencia para la identificación de los materiales.

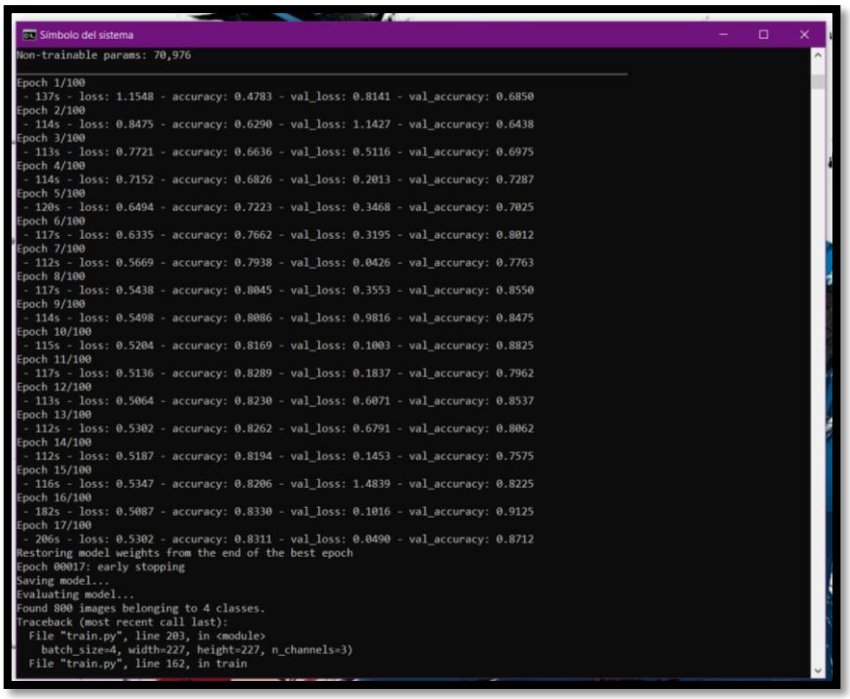


Figura 37.1. Resultados del entrenamiento

3.4. Comprobación de la Hipótesis

De acuerdo al planteamiento de la hipótesis, la cual menciona el desarrollo de una aplicación por medio de una red neuronal, para la identificación y clasificación por medio de un mecanismo y mediante haber hecho una exhaustiva investigación se comprobó la pertinencia con la puesta en práctica por medio del entrenamiento de una red neuronal convolucional, todo esto ya antes mencionado en la elaboración del documento explicado en los capítulos que lo conforman, este modelo tiene la función de identificar y clasificar por medio de imágenes previamente entrenado, que para su funcionamiento se construyó un mecanismo que se le introducen los materiales como pet, cartón y papel, una vez hecho esto, el modelo realiza la tarea bajo este mecanismo que fue construido por medio de placas de mdf 6mm, una banda transportadora, una cámara web de la marca Logitech c270, una placa de Arduino, todo esto satisface las condiciones del modelo entrenado, cuyo resultado está por arriba del 90% un resultado bastante aceptable (véase anexo1), esta aplicación, toma cada imagen por medio de la web cam, leyendo la predicción de fps recorridas en las imágenes, dando como resultado la predicción de las ultimas 30 fps y al superar el .75 de confiabilidad, entonces nos va mandar como resultado el nombre del material que se está introduciendo al prototipo así como su clasificación a un determinado contenedor.

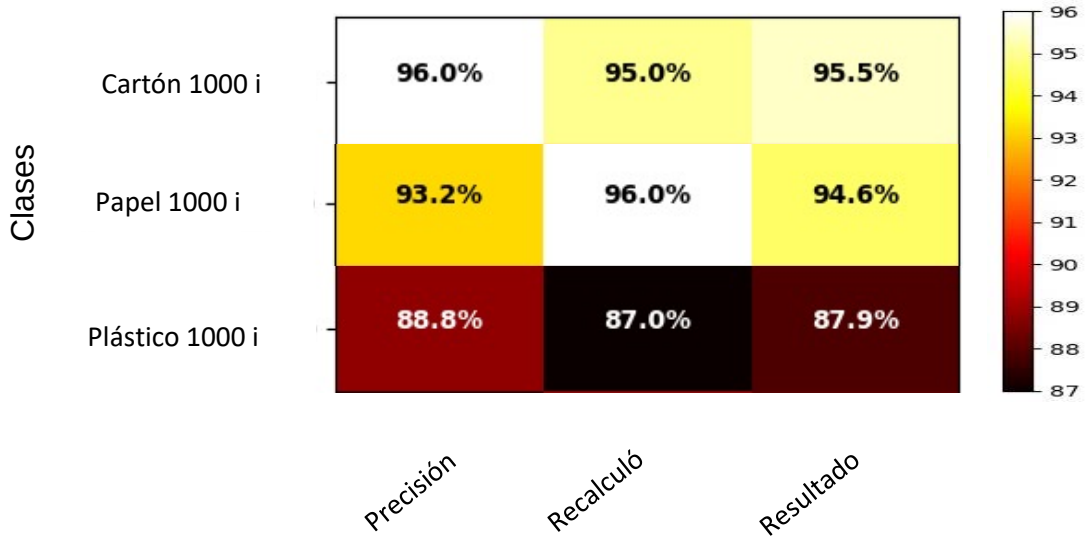
CAPÍTULO IV

RESULTADOS Y CONCLUSIONES

4.1. Resultados

Los resultados mostrados en este capítulo hacen referencia al modelo de red neuronal entrenado para la predicción y clasificación de los materiales reciclables Pet, Papel y Cartón, por tanto, al realizar el proceso de entrenamiento para predicción, se entrenó un modelo con un promedio de 1000 imágenes de cada material para el rentrenamiento, obteniendo como respuesta, una eficiencia mayor al 90%, por lo que de esta manera se obtiene, que entre mayor sea el número de imágenes que se vallan recolectando y procesando en la aplicación, mayor será la efectividad en la predicción de los materiales, por lo que para posteriores mejoras se pueden agregar algunos otros materiales reciclables.

Tabla 18 Entrenamiento tomando 1000 Imágenes de cada material



4.2. Conclusiones

De acuerdo con el planteamiento del problema, la cual menciona, por medio del conjunto con Inteligencia Artificial, desarrollar una aplicación para la identificación de pet, cartón y papel utilizando redes neuronales convolutivas, así y mediante el previo estudio e investigación, se concluye que tras finalizar este trabajo, se ha logrado realizar una aplicación por medio de una red neuronal convolutiva para la predicción y clasificación de los residuos, por medio de imágenes para su previo entrenamiento, por lo que todo esto ha sido posible mediante el estudio de Inteligencia Artificial, implementando diversas herramientas que proporciona la plataforma de Machine Learning, como lo son tensorflow, keras, numpi, OpenCV que es una herramienta diseñada por Google, de esta manera se mencionó en el capítulo 2, metodología y desarrollo una empresa dedicada al desarrollo de la tecnología basa en IA, en la predicción de objetos por medio de imágenes que es Open source proporcionando repositorios de código abierto que facilitan la capacidad de abstracción para la implementación de redes neuronales, dando ventaja a una gran reducción en los tiempos de desarrollo para obtener aplicaciones sobre sistemas embebidos. De modo que los objetivos específicos se fueron cumpliendo al paso del desarrollo del proyecto, todos estos llevados acabo en los diferentes capítulos que estructuran la elaboración del presente, todo esto nos brindo un gran resultado en la predicción de cada imagen a identificar y posteriormente se implementaron nuevos materiales para ver el comportamiento de este modelo, pues para posteriores entrenamientos y como propuesta en desarrollar y hacer más eficiente a la predicción de materiales se entrenó con las siguientes imágenes como aluminio, metal, vidrio imágenes tomando en un primer entrenamiento 500 imágenes y posteriormente 1000 imágenes, por lo que se puede mencionar que entre mayor sea el número de imágenes, será más eficiente la predicción del modelo para cada material que se valla agregando.

Tabla 19 Entrenamiento tomando 500 imágenes de cada material

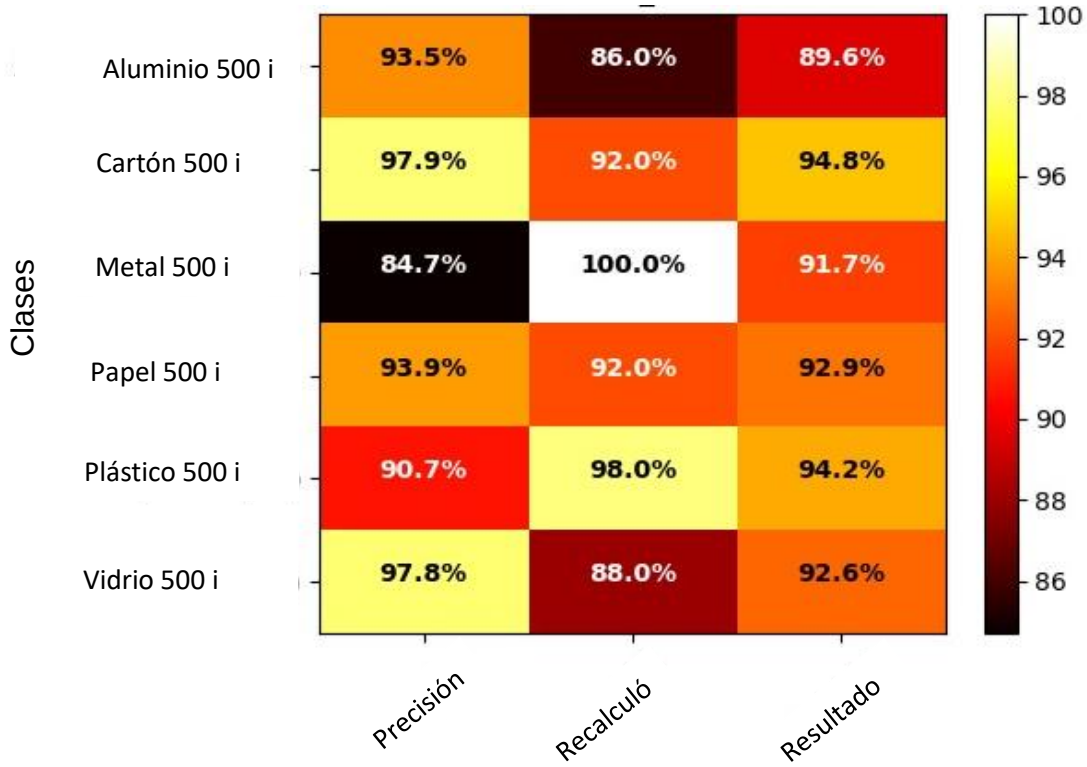
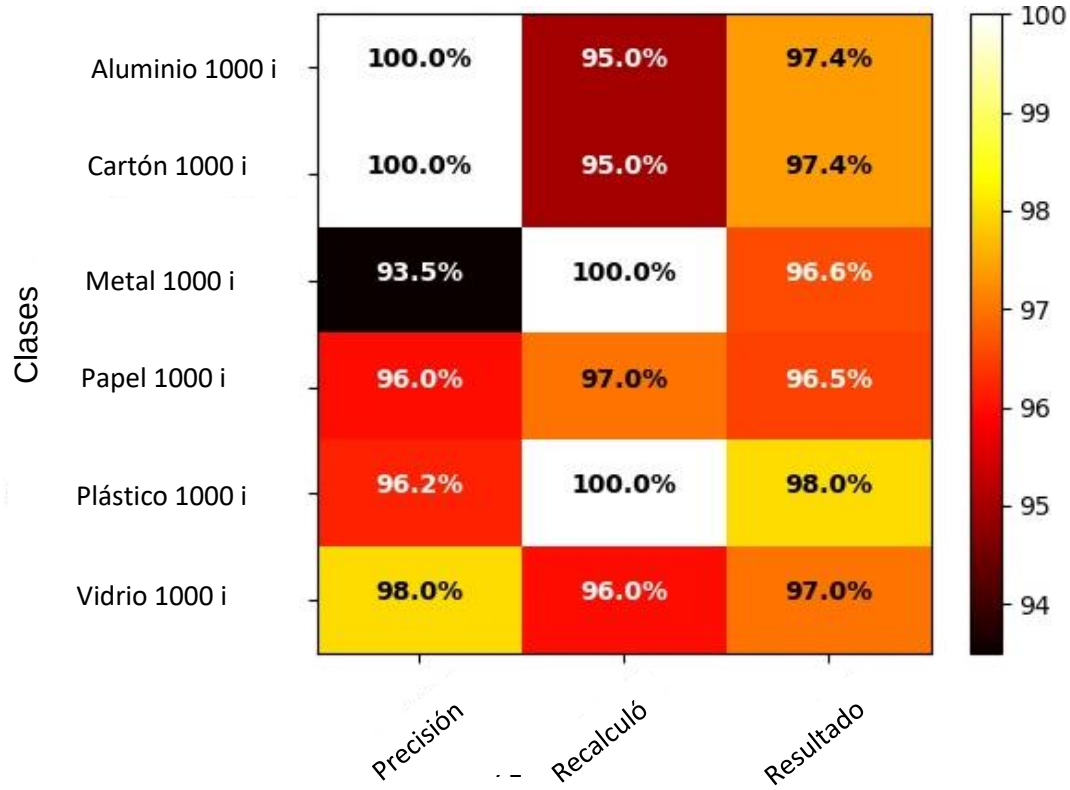


Tabla 20 Entrenamiento tomando 1000 imágenes de cada material



Referencias Bibliográficas

Wikipedia. Obtenido de https://en.wikipedia.org/wiki/Machine_learning M. l. (2020).

Antonio-Diego, I.-S.-S. (2019). *TodoRedesCNN*. Obtenido de http://www.diegocalvo.

Aprendemachinelearning. (2016). Obtenido de https://www.aprendemachinelearning.com/aprendizaje-profundo-una-guia-rapida/

Artificial., H. d. (2020). *InteligenciaArtificial.com*. Obtenido de http://ocw.uc3m.es/ingenieria-telematica/

Crispi, J. (2020). *Medium.* Obtenido de www.DeepLearningBásico-Python.com

ECOCE. (s.f.). *ecoce.mx*. Obtenido de https://www.ecoce.mx/

EIA, U. (2015). Memorias. *Memorias EIA*.

Enzyme. (01 de 2020). Obtenido de https://blog.enzymeadvisinggroup.com-inteligencia-artificial

GogglesAplication. (4 de 12 de 2015). *Google.* Obtenido de https://elandroidelibre.elespanol.com/2015/12/que-ha-pasado-con-google-goggles.html

Goodfellow. (2012). *GoogleAcademico*. Obtenido de https://scholar.google.com.mx/scholar?q=(Goodfellow+et+al.,+2012).&hl=es&as_sdt=0&as_vis=1&oi=scholart

IMAGNET. (02 de 2020). Obtenido de http://www.image-net.org/

IMCO. (2012). Obtenido de https://imco.org.mx/indice_de_competitividad_estatal_2012/resultados/

Inc., P. H. (2004). Architectures, Algorithms and Aplication. *Fundamental of Neural Networks*, 296.

JordiTorres.AI. (01 de 2020). *Torres.AI*. Obtenido de https://torres.ai/espanol-introduccion-a-tensorflow-para-inteligencia-artificial-y-deep-learning/

Loncomilla, P. (206). *RedesConvolucionales*. Obtenido de https://ccc.inaoep.mx/~pgomez/deep/presentations/2016Loncomilla.pdf

neuronales, C. d. (02 de 2020). *RedesCNN*. Obtenido de http://www.redesCNN.com

Numpy. (02 de 2020). *Thefundamentalpackage*. Obtenido de https://numpy.org/

OpenSource. (02 de 2020). Obtenido de https://www.redhat.com/es/topics/open-source/what-is-open-source

Packaging. (10 de 2018). Obtenido de http://www.packaging.enfasis.com/notas/73727-hablan-la-tecnologia-clasificacion-sensores-aumentar-la-cantidad-y-la-calidad-del-plastico-recuperado

Pardos, E. C. (2 de Septiembre de 2013). *Google*. Obtenido de http://oa.upm.es/215/1/10200404.pdf

Programarfacil. (02 de 2020). *Programasfacil*. Obtenido de https://programarfacil.com/podcast/81-vision-artificial-opencv-phyton/

Python.org. (Febrero de 2020). *Portal python-opencv*. Obtenido de https://pypi.org/project/opencv-python/

Python.org. (02 de 2020). *Python.org*. Obtenido de https://www.jaewoong.org/pubs/fpt16-accelerating-

Samuel., J. M. (28 de 11 de 2019). *nfolab.stanford.edu*. Obtenido de http://infolab.stanford.edu/pub/voy/museum/

SEMARNAT. (2015). Obtenido de https://apps1.semarnat.gob.mx:8443/dgeia/informe15/tema/cap7.html

SEMARNAT. (2019). *Google*. Obtenido de https://www.gob.mx/semarnat

Anexos

Anexo 1.

Comprobación de hipótesis mediante el entrenamiento y construcción del prototipo.

Bajo el estudio y una investigación profunda se desarrollo un mecanismo para satisfacer las necesidades de clasificar los tipos de materiales reciclables como son lo son pet, papel y cartón por medio de una red neuronal convolutiva, por lo que a continuación se ilustra el proceso del mismo.

Figura 38. Visita a Centros de Reciclado

Figura 39. Selección de Material Cartón

Figura 40. Selección de material Papel

Figura 41. Selección de Material Pet

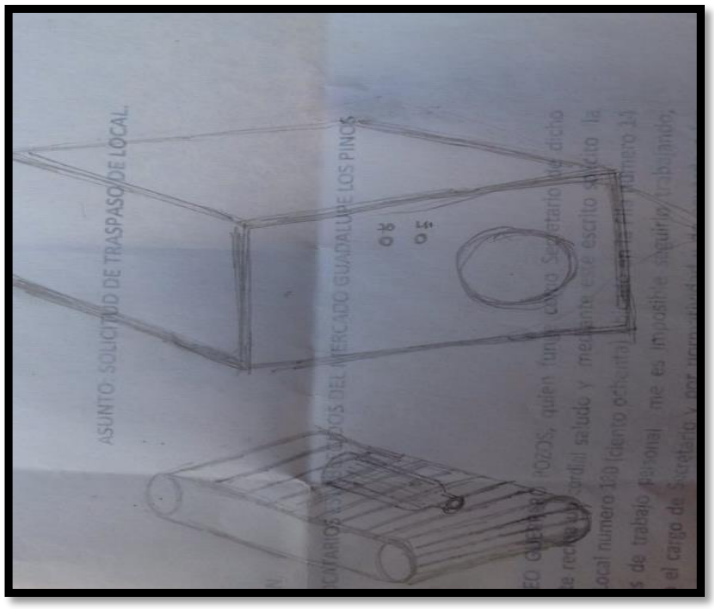


Figura 42. Propuesta construcción de prototipo

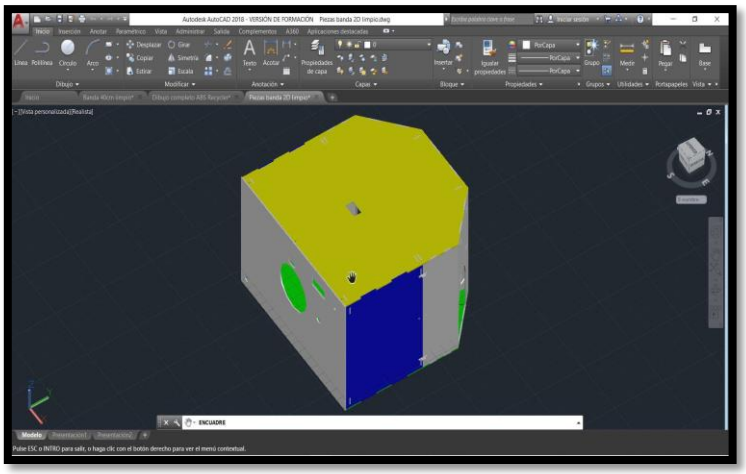


Figura 43. Diseño del prototipo

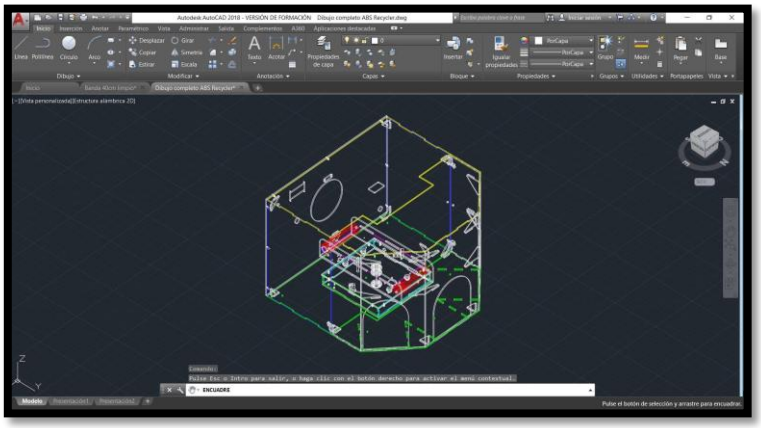


Figura 44. Diseño del prototipo

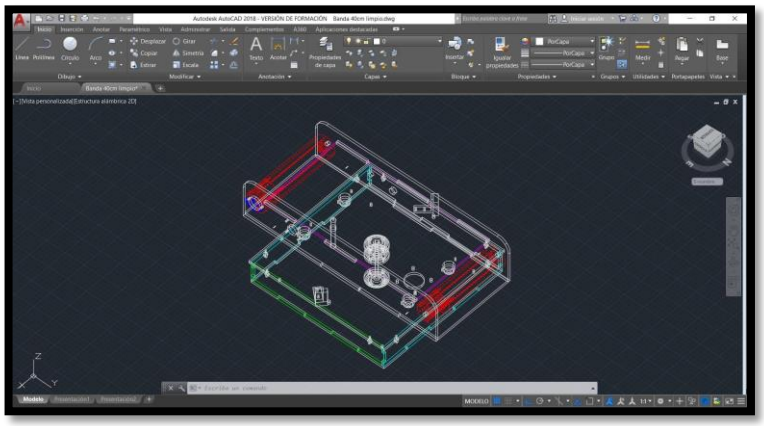


Figura 45. Diseño del prototipo

Figura 46. Construcción del prototipo

Figura 47 Construcción del prototipo

Figura 48. Construcción del prototipo

Figura 49. Construcción del prototipo

Figura 50. Prototipo final

Figura 51. Prototipo final

Figura 51. Pruebas prototipo

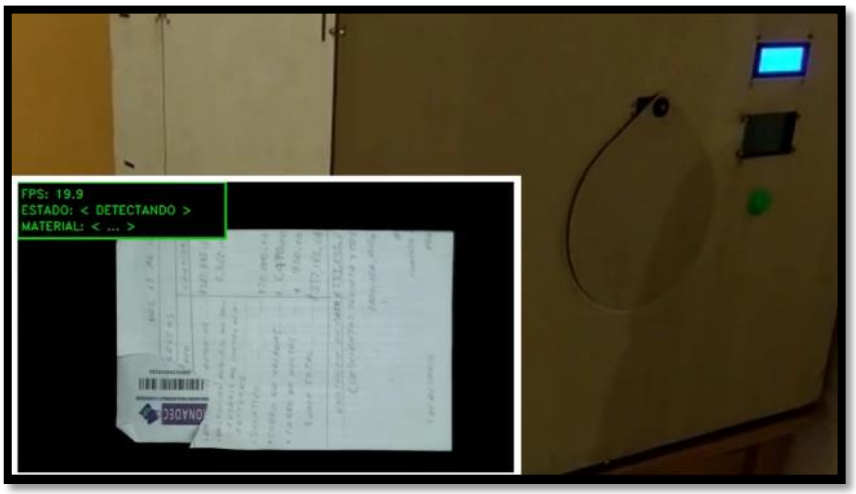


Figura 52. Pruebas prototipo