CLASIFICADORA DE FRUTAS USANDO IA (Redes Neuronales)

IVÁN ANDRÉS ATIS (1), CARLOS EDUARDO BAÉZ (2), JUAN ERNESTO CHAMORRO (3), LAURA CATALINA GONZÁLEZ (4)

1. ivan.atis@udea.edu.co 2. eduardo.baez@udea.edu.co 3. ernesto.chamorro@udea.edu.co

4. laurac.gonzalez@udea.edu.co

Universidad de Antioquia, Potencia Fluida (Electroneumática)

Medellín, Colombia

2022

RESUMEN: El presente artículo presenta un informe detallado del desarrollo del proyecto final de potencia fluida, el cual consiste en construir el prototipado de una máquina que integre soluciones tecnológicas multidisciplinares aplicables a algún proceso a nivel industrial, según las necesidades visibilizadas en la actualidad en torno a la cuarta revolución industrial. Más específicamente, se hace uso de la inteligencia artificial (Machine Learning) junto con un controlador lógico programable (PLC) para la transformación de un proceso industrial orientado a la clasificación de frutas.

PALABRAS CLAVE: Inteligencia Artificial, redes neuronales, automatización, control automático, procesos industriales, interfaces hombre-máquina

ABSTRACT: This article presents a detailed report of the development of the final project of the module of pneumatics, which consists of building the prototype of a machine that integrates multidisciplinary technological solutions that can be applied to some industrial processes, according to the needs currently identified in the fourth industrial revolution. More specifically, using artificial intelligence (Machine Learning) together with a programmable logic controller (PLC) for the transformation of an industrial process oriented to the classification of fruits.

KEYWORDS: Artificial Intelligence, neuronal networks, automation, automatic control, industrial processes, Human-machine interfaces, Deep learning.

1. INTRODUCCIÓN

En la actualidad el uso de herramientas de inteligencia artificial, más puntualmente el uso de redes neuronales artificiales (RNA) representan una de las herramientas más poderosas del aprendizaje de máquina en ámbitos industriales, como puede ser el control de procesos mediante el reconocimiento de patrones y predicción de comportamientos a partir de conjuntos de datos. Además, la inteligencia artificial en los últimos años ha impulsado la automatización industrial, puesto que aparte de los algoritmos inteligentes y macrodatos que lo conforman dentro sistemas informáticos, los procesos industriales incluyen dispositivos físicos, datos que deben obtenerse a través de sensores y comandos que deben ser enviados a través de los actuadores y sistemas de control. Es decir, la IA abre nuevas posibilidades a los controladores de máquina para el sector industrial al poder incorporar a los autómatas programables herramientas con más inteligencia y conectividad.

2. DISEÑO DE ALTO NIVEL

2.1 Inspiración

Actualmente nos movemos en un mundo a gran velocidad, donde se busca que todo sea rápido y eficiente, el uso de herramientas computacionales y diferentes tipos de maquinaria facilitan la vida en gran medida, desde cosas elementales como electrodomésticos hasta maquinaria de tecnología avanzada, hacen parte de las nuevas herramientas para el hombre, una tecnología relativamente nueva es la inteligencia artificial, esta se ha logrado establecer en soluciones industriales, mostrándose como una alternativa confiable para la automatización de procesos, aumentando productividad con gran precisión, minimizando errores humanos.

En este proyecto damos una opción viable para la selección de frutas automatizada con reconocimiento mediante inteligencia artificial.

2.2 Estado del arte

Desde siempre ha existido un paradigma sobre el funcionamiento del cerebro humano, desde ahí parte la evolución tecnológica conocida como inteligencia artificial, donde se trata de asemejar el comportamiento del cerebro humano por medio de algoritmos computacionales.

En 1943, se registraron los primeros intentos por elaborar una red neuronal, capaz de asemejar el funcionamiento del cerebro humano por medio de las RNA; A principios de los 80 el investigador Geoffrey Hinton, desarrollo un concepto que buscaba explorar las RNA por medio del uso de software y Hardware para crear una forma más pura de inteligencia artificial, que más adelante se conocería como Deep Learning.
En 1991, el investigador Jürgen Schmidhuber elabora una red de cientos de operadores lineales o capas de neuronas, el cual, mediante el uso de un "preentrenamiento no supervisado", entrena otra red neuronal recurrente (RNN).

En 2004, el Instituto Canadiense para la Investigación Avanzada con el apoyo de Yann LeCun y Yoshua Bengio, Hinton fundó el programa "Neural Computation and Adaptive Perception", un grupo selecto de científicos de varias especialidades, realizaron un profundo avance del concepto de Deep Learning, creando sistemas informáticos que imiten la inteligencia ecológica.

En el 2009, el grupo de Jürgen gano una competencia internacional de reconocimiento de patrones con tasas de error de hasta 0.37%, convirtiéndose en la primer RNN y sistemas Deep Learning en ganar una competencia de esta talla.

En 2010, se desarrolla el primer reconocimiento de patrones visuales sobrehumano, sobrepasando, como su nombre lo indica, el reconocimiento en este campo del ser humano. A esta técnica se le llamo "GPU-MPCNN".

Actualmente la inteligencia artificial es adoptada en diferentes entidades para el desarrollo de tecnología como Google, Amazon o Apple y continuamente se construyen modelos más sofisticados o robustos, con un nivel de acercamiento cada vez mayor al comportamiento del celebro.

2.3 Objetivo general

El objetivo del proyecto es lograr el reconocimiento de frutas con inteligencia artificial, el modelo recibe imágenes de las frutas que se mueven a través de una banda transportadora, donde se disponen sensores y actuadores para su clasificación dependiendo de una orden dada por un cliente

3. CONCEPTO

El concepto principal de la clasificadora se basa en la necesidad de suministrarle al cliente un producto de manera eficiente y precisa, en este caso se clasifican frutas las cuales el mismo cliente selecciona en una pantalla táctil y por medio de la aplicación de diversos sensores, actuadores neumáticos, programación de redes neuronales artificiales y demás se pueda realizar la automatización del proceso, logrando de esta manera la aplicación de nuevas tecnologías a un problema específico de producción o despacho de productos.

3.1 Descripción de los mecanismos y funcionamiento

Para la construcción de la clasificadora de frutas se utilizaron los siguientes elementos:

- Sistema de alimentación: Es el sistema encargado de almacenar las frutas para su posterior alimentación hacia la banda transportadora.
- Banda transportadora: Se encargada de transportar las frutas para su posterior selección.
- Motorreductor 12V: Se encarga de mover la banda transportadora.
- Pistones neumáticos: Se encargan alimentar la banda transportadora y la selección de cada fruta.

- Sensores fotoeléctricos: Se encargan detener la banda trasportadora, además de censar la posición de la fruta en determinadas posiciones.
- Sensores magnéticos: Se encargan de censar la posición de los vástagos de los pistones neumáticos.
- Relé: Encargado de encender y apagar el motorreductor
- PLC: Se encarga de realizar la automatización de todo el proceso recibiendo las señales de los sensores y activando los actuadores.
- Cámara: Realiza la obtención de imágenes para comparar dichas imágenes con la información dentro de la red neuronal almacenada en Python.

El funcionamiento general de la clasificadora de frutas es el siguiente:

i. El usuario selecciona a través de una pantalla táctil las frutas que desea comprar, posteriormente da inicio al proceso el cual según la selección del usuario se clasifican o descartan las frutas.
ii. Una vez iniciado el proceso se enciende la banda transportadora por medio del relé el cual está conectado al PLC.
iii. Se activa el pistón neumático moviendo la fruta desde el depósito hasta la banda transportadora, con la condición de que el sensor inductivo ubicado en el depósito cense una fruta y además si el sensor magnético censa al vástago del pistón en una posición retraída.
iv. Una vez la fruta se encuentra sobre la banda transportadora se mueve hasta un segundo sensor en este caso réflex el cual al detectar una fruta envía una señal para detener la banda y de esta manera posicionar la fruta debajo de la cámara la cual toma una foto para que la red neuronal detecte que tipo de fruta es y de esta manera poder clasificarla.
v. Una vez que la red neuronal detecta el tipo de fruta se enciende nuevamente la banda, trasportando la fruta hasta un tercer sensor fotoeléctrico que según el tipo de fruta acciona un segundo pistón neumático el cual se activa si la red neuronal detecta que la futa coincide con la que escogió el usuario o se mantiene en posición retraída si no coincide, descartando así la fruta.

3.2 Diseño CAD

Dado que el proyecto se planteó desde cero fue necesario realizar el modelado CAD de cada una de las piezas que compondría el ensamble (bases, sensores, actuadores, etc.), esto con el fin de evitar inconvenientes al momento de realizar el montaje, una vez realizado el diseño CAD en Inventor se procedió a realizar los planos para el corte de los soportes (MDF), obtener la geometría de las bases para su posterior impresión en 3D, el diseño CAD general se muestra a continuación.

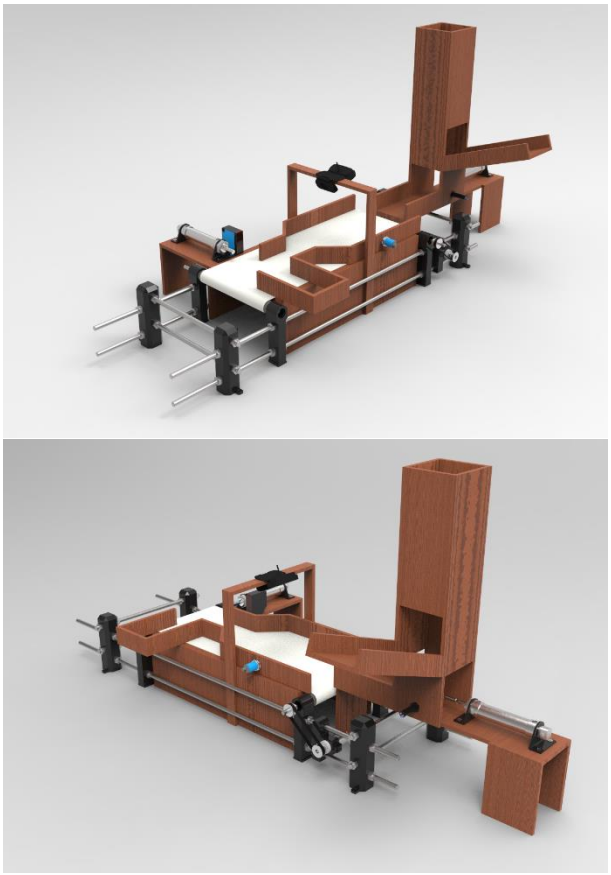

Figura 1: Diseño CAD.

4. DESARROLLO DEL PROYECTO

4.1 PROGRAMACIÓN PYTHON:

Entrenamiento red neuronal y selección de la arquitectura.

Con el objetivo de integrar inteligencia artificial en el proyecto, se consideró pertinente la aplicación de una red neuronal convolucional para la clasificación de las frutas.

En las partes tempranas de la red se utilizaron redes neuronales convolucionales, las cuales se encargan principalmente de la extracción de las características que se encuentran dentro de una imagen, tales como la textura, los bordes, transiciones de color, formas, etc. Cada capa de la red convolucional contiene unos filtros que serán aplicados a la imagen, tal como se había mencionado anteriormente cada filtro de encarga de extraer algunas características que serán entregadas a las capas siguientes. Algunas de las características que pueden extraer estos tipos de redes se encuentran en la siguiente imagen:

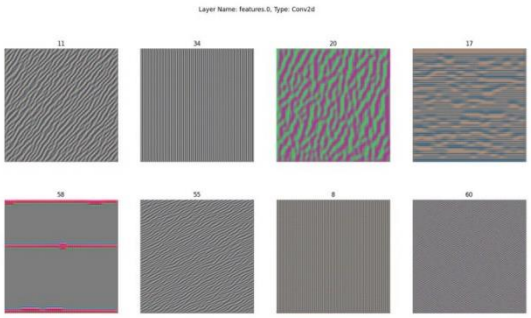

Figura 2 Características de la imagen aprendidas en capas tempranas.

En las primeras capas de la red se aprenden las características más generales, generalmente asociadas a las geometrías. Conforme la red se va volviendo más profunda, estas características se van volviendo cada vez más complejas y cada vez se especializan en las clases a las que se entrenan centrándose en sus características más específicas, como por ejemplo la nariz de una persona, los ojos, manos, pies, y demás características notables de una clase en específico. Entre más capas tenga la red, mayores características de la imagen podrá extraer, lo cual explica la tendencia en los algoritmos de aprendizaje profundo de ir aumentando el número de capas con el propósito de mejorar el rendimiento entregado por estos modelos. Las activaciones producidas por las imágenes en las capas más profundas de la red se pueden encontrar en la siguiente imagen, en donde es posible observar que estas activaciones corresponden a las características más notables de cada clase, como por ejemplo en el caso del pavorreal, se centran en los cambios de color del plumaje, y en el caso del zorro en la boca y las orejas.

Figura 3: Activaciones de la red neuronal generadas por la imagen de entrada.

Ahora bien, aunque la implementación de las redes neurales profundas puede ayudar al modelo a desempeñarse mejor en las tareas que realiza, debido a que pueden extraer características cada vez más complejas, presentan varios problemas durante su entrenamiento, en donde el más importante de ellos es la degradación que se presenta durante el entrenamiento de la red, dicha degradación afecta el entrenamiento de las capas más profundas de la red, haciendo más difícil la optimización de los parámetros esta.

En el modelo que se propone en este trabajo se utiliza una técnica que permite disminuir esta saturación y de esta manera logra conseguir una arquitectura mucho más profunda, la cual permita mejorar la capacidad de la red de extraer características más complejas de estas imágenes. La solución a este problema se conoce por us nombre en inglés como shortcut connection, o también como Deep Residual learning.

Las conexiones de salto se pueden definir como las activaciones de una capa que son sumadas a las activaciones de una capa posterior. En la siguiente imagen se puede observar de manera gráfica la configuración de una función de salto.

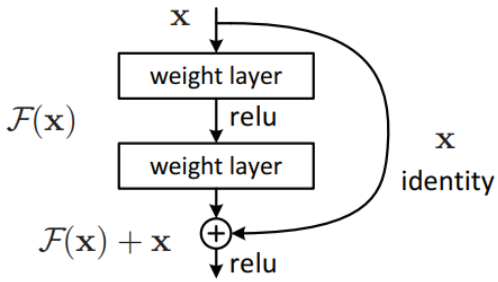

Figura 4: Esquema gráfico de las conexiones de salto.

Una vez definida la estructura general de la red, es necesario introducir el Dataset con el cual la red fue entrenada. Para el entrenamiento de esta red se probaron diversas fuentes, desde el dataset de OpenImages de Google, en el cuál se encuentran más de 16 millones de imágenes clasificadas con muy alta calidad, hasta datasets de Kaggle creados por los usuarios o compañías intersadas en hallar soluciones que involucraran la integración tecnológica para el reconocimiento de las frutas. Lastimosamente, durante el filtrado de datos y visualización de contenidos de los distintos datasets, solo uno cumplió con los requerimientos más próximos a los deseados. En el caso del dataset de Google, al realizar el filtrado de las imágenes para la tarea de clasificación, la cantidad de imágenes disponibles para el entrenamiento se redujo a solo 2000, lo cual es una cifra muy pobre teniendo en cuenta la cantidad de datos necesarios para poder entrenar una red con un rendimiento aceptable. Los otros datasets presentaban una calidad muy baja que puede generar rendimientos muy bajos del modelo durante la puesta en marcha, tales como la mayoría de las imágenes centradas, con buena iluminación, mismo tamaño y fondo de la imagen. Esta falta de diversidad de los datos de entrenamiento no favorece el fortalecimiento de la red para la extracción de características durante la etapa de entrenamiento, por lo que generalmente se presentaran errores durante la puesta en marcha del modelo dado que la distribución de los datos cuando el modelo está en producción va a ser distinta comparada con la cual la red fue optimizada. Estas diferencias en la distribución pueden ser por ejemplo imágenes borrosas, diferentes posiciones de las frutas, tamaños de imágenes pequeños, falta de iluminación, etc. Por tanto, en la medida de lo posible se recomienda que los datos de puesta en marcha y entrenamiento de la red tengan la misma distribución, de esta manera las métricas del modelo durante el entrenamiento estarán muy cercanas a las de producción. En la siguiente imagen se encuentran algunas imágenes de entrenamiento tomadas de un dataset de mala calidad, en las cuales se puede observar que todas tienen el mismo fondo, buena iluminación y la cámara tiene una calidad decente, aspectos que puede que no siempre se presenten durante la etapa de producción de la red neuronal.

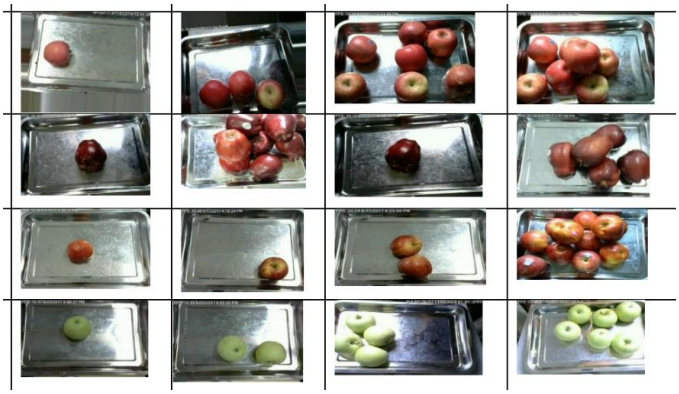

Figura 5 Dataset de baja calidad, en donde se puede apreciar el mismo fondo para todas las imágenes.

Afortunadamente se encontró un dataset que tenía unos datos un poco más diversos, aunque también tenía distintas limitaciones. Este conjunto de datos contiene aproximadamente 230000 imágenes correspondientes a 262 clases de frutas. A pesar de la gran diversidad de clases y la cantidad de imágenes que se encuentran en él, sigue presentando, aunque en menor medida comparado con los anteriores, una baja diversidad y complejidad de los datos. Estas imágenes de las frutas presentan condiciones casi ideales, como por ejemplo el fondo blanco y frutas limpias en gran parte de las imágenes, frutas frescas y también imágenes tomadas con buena calidad. Aunque se intentó mitigar el impacto que tiene la falta de diversidad de estos datos con el rendimiento del modelo haciendo uso de datos artificiales, se esperaban (y durante la puesta en marcha del modelo sucedieron) ciertos errores en el reconocimiento de las distintas frutas, como por ejemplo en el caso de la manzana o de la pera. En estas dos frutas, tal como se puede observar en la imagen de abajo, las capturas fueron tomadas en las posiciones más características de ellas, como por ejemplo vistas de frente o cortadas. Otro factor de gran importancia, que también tuvo efectos indeseables en la puesta en marcha, es que las frutas estaban frescas, entonces si las frutas que se iban a clasificar tenían algunos moretones o golpes, el modelo las asociaba más fácilmente a una chirimoya que a una manzana o a una pera.

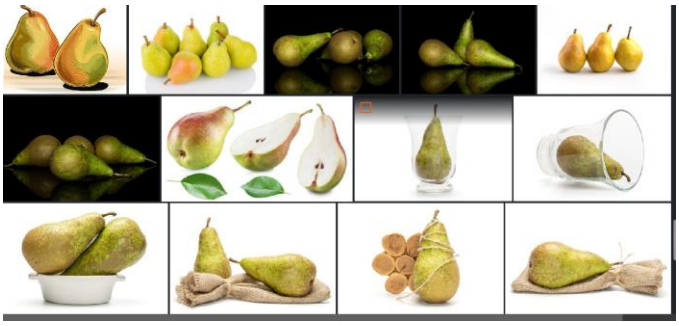

Figura 6: Ejemplo de las peras en el dataset utilizado, es posible apreciar que se encuentran en su posición característica, y están libres de golpes.

Retomando nuevamente la arquitectura de la red neuronal, dado que el modelo que se necesitaba no era tan complejo, se utilizó la API secuencial de Keras para la construcción del modelo. Teniendo en cuenta la falta de diversidad de las imágenes, y que podrían contribuir muy poco al entrenamiento de las capas encargadas de la extracción de características de la red, se utilizó una arquitectura de un modelo preentrenado conocido como ResNet-50, dicha arquitectura contiene conexiones de salto que permiten un entrenamiento y una extracción de características más complejas comparadas con otros modelos.

Con el objetivo de sacar el máximo provecho que puede ofrecer la implementación de modelos preentrenados, se utilizó una técnica ampliamente utilizada en el campo del entrenamiento profundo conocida como Transfer Learning, en la cual se toman los valores que contiene esta red como punto inicial de partida, evitando tener que entrenar todos los pesos con inicialización aleatoria, y optimizarlos con descenso del gradiente, lo cual puede requerir unos tiempos de entrenamiento muy largos, y teniendo en cuenta también la baja diversidad del dataset, se puede esperar, que si la red fuera entrenada desde cero, tendría un rendimiento muy inferior comparada con la que toma los datos ya entrenados como punto de partida. En la siguiente figura se encuentra parte del código con el cual se descargó la arquitectura de ResNet-50.

```
def feature_extractor(inputs):

  feature_extractor=keras.applications.resnet.ResNet50(
      input_shape=(224, 224, 3),
      include_top=False,
      weights='imagenet')(inputs) #This model is built with the sequential API

  return feature_extractor
```

Figura 7: Descarga del modelo preentrenado de Keras.

Tal como se había mencionado anteriormente, para contrarrestar en la medida de lo posible la baja calidad del dataset, se utilizaron datos artificiales, para simular imágenes ya sean borrosas, giradas, a blanco y negro, cortadas, etc. Esto se hizo mediante la aplicación de diversas capas de Keras, algunas transformaciones que se le aplicaron a las imágenes se encuentran en la siguiente figura.

```
x=keras.layers.RandomZoom(0.5,0.5)(x)
x=keras.layers.RandomContrast(1)(x)
x=keras.layers.RandomCrop(IMG_SIZE[0], IMG_SIZE[1])(x)
x=keras.layers.RandomFlip()(x)
x=keras.layers.RandomRotation(255)(x)
```

Figura 8: Capas de creación de datos artificiales para aumentar la cantidad de datos de entrenamiento.

Con la parte más importante de la red definida, tal como la parte de procesamiento, creación de imágenes artificiales y extracción de características, solo resta colocar las capas densas que son las encargadas de realizar la clasificación de las imágenes. Una vez fueron definidas, el modelo procedió al entrenamiento. En la siguiente imagen se encuentra un pantallazo de las métricas de la red durante la primera etapa de entrenamiento, lo cuál explica el bajo resultado en la precisión. Cabe destacar que en la segunda etapa del entrenamiento se logró una precisión mayor al 93%, dado que se entrenó la red completamente con una tasa muy pequeña de aprendizaje, lo cual le permite al modelo un mayor ajuste a los datos, generando unas pérdidas muy bajas, y por tanto buena precisión; esta segunda etapa de entrenamiento se conoce también como Fine-tuning.

```
Epoch 1/35
2821/2821 [==============================] - 3107s 1s/step - loss: 2.8723 - accuracy: 0.5352 - val_loss: 2.0268 - val_accuracy: 0.5658
Epoch 2/35
2821/2821 [==============================] - 3078s 1s/step - loss: 1.4726 - accuracy: 0.6762 - val_loss: 1.6853 - val_accuracy: 0.6144
Epoch 3/35
2821/2821 [==============================] - 3069s 1s/step - loss: 1.0769 - accuracy: 0.7519 - val_loss: 1.6118 - val_accuracy: 0.6318
Epoch 4/35
2821/2821 [==============================] - 3065s 1s/step - loss: 0.8229 - accuracy: 0.8063 - val_loss: 1.3925 - val_accuracy: 0.6790
Epoch 5/35
2821/2821 [==============================] - 3063s 1s/step - loss: 0.6444 - accuracy: 0.8474 - val_loss: 1.4569 - val_accuracy: 0.6812
```

Figura 9: Resultados de la primera etapa de entrenamiento del modelo, previa al fine-tuning

Para la comunicación de Python con el PLC se utilizó el protocolo de TCP mediante la librería de Snap7. Con el objetivo de poder leer mejor el código, y también facilitar la solución de errores, se elaboraron distintas funciones encargadas ya sea de la lectura, escritura y clasificación de la imagen. Un ejemplo de estas funciones se encuentra en la siguiente imagen.

```python
def escritura(clase, direccion=0):
    IP='192.168.0.1'
    RACK=0
    SLOT=1
    DB_NUMBER=100
    data=bytearray(2)
    PLC=snap7.client.Client()
    conexion=PLC.connect(IP, RACK, SLOT)
    estado=PLC.get_cpu_state
    print('la conexion al PLC es {}'.format(estado))
    snap7.util.set_int(data, 0, clase)
    PLC.db_write(100, direccion, data)
    PLC.destroy()
```

Figura 10: Bloque de escritura de datos al DB del PLC Siemens.

La fotografía se tomó con el celular con la ayuda de la librería de OpenCV. El algoritmo que se empleó para la toma de la foto se muestra a continuación.

```python
def encendido_camara():
    cap=cv2.VideoCapture(0) #ABRIMOS LA CAMARA DEL COMPUTADOR
    estado, foto=cap.read() #En el estado va un bool indicando
    foto=foto[:,:,::-1]
    foto=cv2.resize(foto, dsize=(220,220))
    foto=np.expand_dims(foto, axis=0)
    print('El estado de la foto es {}'.format(estado))
    plt.imshow(np.squeeze(foto, axis=0))
    cap.release()
    return foto
```

Figura 11: Función de captura de la imagen con la librería de OpenCV.

4.2 PROGRAMACIÓN PLC: SIEMENS S7-1200

La combinación de la inteligencia de los autómatas programables con los accionadores industriales, permiten automatizar variedad de procesos industriales, esto se logra a partir de los elementos que componen una instalación automatizada, los cuales son:

- Accionadores: Motor eléctrico y actuadores neumáticos.
- Reaccionadores: Comandan y activan los accionadores, en este caso el contactor que controla el switcheo ON/OFF del motor eléctrico y las electroválvulas asociadas a cada actuador neumático.
- Captadores: Elementos que informan al órgano de mando o PLC los eventos que sucedan en él, para así conocer el estado actual del proceso y decidir las transiciones futuras para completar la red de Petri. En este caso se utilizaron sensores captadores de posición (sensores fotoeléctricos y sensores finales de carrera magnéticos).
- Elementos de diálogo hombre-máquina: Pantalla HMI autómata programable.
- Elemento de mando: El autómata programable ejecuta las acciones consideradas, en base a las señales de los captadores y a las órdenes que provengan del operador. Es decir, el autómata dialoga con el operario, recibiendo órdenes y suministrando acciones.

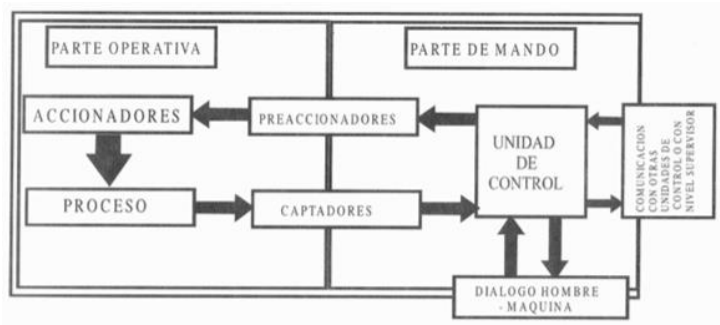

Figura 12: Proceso lógico PLC

El autómata programable PLC se programó para ejecutar la red de Petri visualizada en la *Figura 13.*

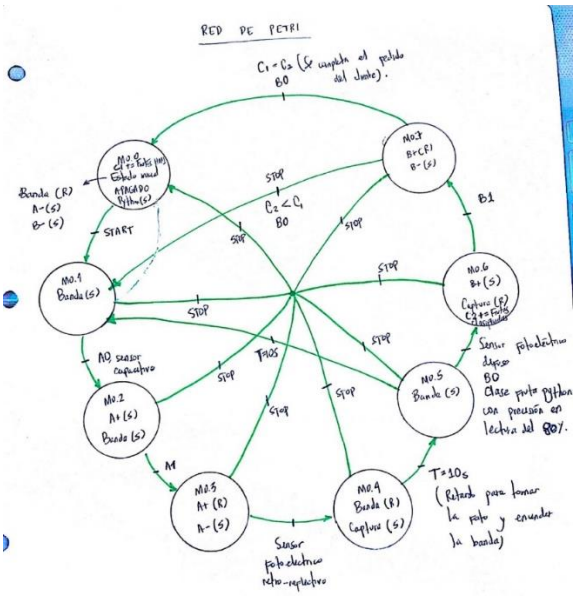

Figura 13: Red de Petri proyecto clasificador de frutas usando IA

5 RESULTADOS

Tal como se había mencionado durante el análisis del dataset, se esperaban errores durante la clasificación de las imágenes, dado que los datos con los cuales fue entrenada la red son difícilmente reproducibles durante la etapa de producción. La principal fuente de errores estaba relacionada con el mal estado de la

fruta, como por ejemplo golpes, o también distintos colores de algunas de ellas, como por ejemplo el mango. En el caso del mango, la mayoría de las imágenes de entrenamiento contiene mangos amarillos, por tanto, ante la presencia de un mango verde, la red va a tener problemas para la predicción inclinándose por otras clases, como por ejemplo el aguacate, o en el caso de la manzana, cuando esta se encuentra vista desde la parte superior, la puede predecir como una manzana, pero con probabilidades muy bajas, o como otra fruta.

Sin embargo, a pesar de estos problemas, se consiguió un rendimiento aceptable de la máquina, teniendo en cuenta el bajo presupuesto para su elaboración, y también la baja diversidad del dataset. En la imagen siguiente se encuentra una captura del modelo durante el funcionamiento de la máquina.

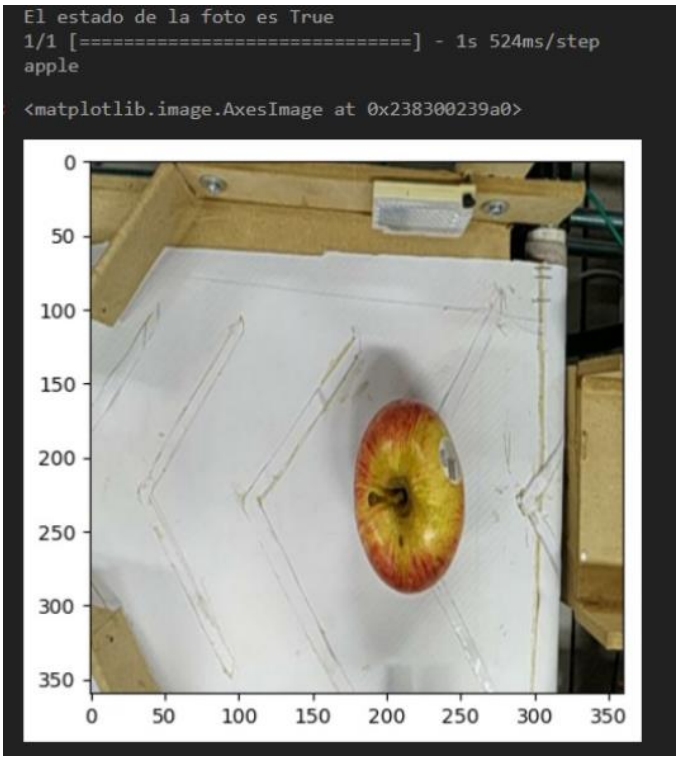

Figura 14: Ejemplo de la predicción de una manzana.

6 TRABAJO FUTURO

En el presente proyecto el inconveniente más relevante que se presento fue con el reconocimiento de frutas, esto debido a que la red neuronal fue entrenada con datasets de frutas en posiciones muy específicas, las cuales en la práctica fue muy difícil de conseguir, por otra parte se sabe que la apariencia de las frutas varían de acuerdo a su país de origen, por esta razón también se presentaron inconvenientes ya que muchas veces la red neuronal confundía las frutas, es por esto que para un trabajo futuro se recomienda generar datasets con frutas de la región en donde se vaya a implementar la clasificadora de frutas, lo cual permitiría una mejor precisión al momento del reconocimiento.

Con respecto al diseño de todo el proceso se puede implementar un subsistema que se encargue de regresar hacia el depósito las frutas descartadas para que el proceso sea totalmente autónomo, dicho sistema queda a criterio de quien desee realizar la implementación.

Finalmente es importante mencionar que este proyectó se puede llevar a cabo a escala industrial y no solo es aplicable industrias fruteras, si no que se puede aplicar en procesos en donde se requiera un reconocimiento y selección de productos de cualquier tipo.

7 CONCLUSIONES

Este proyecto integró teoría y práctica de distintas disciplinas aprendidas durante la carrera, aquí se implementó consideraciones de diseño, automatización, materiales, inteligencia artificial, instrumentación, electrónica, neumática entre otras, adicionalmente se profundizo en softwares como Autodesk Inventor, Tía Portal, Python, Linux.

En este proyecto se implementó inteligencia artificial con Deep Learning para el reconocimiento de frutas para su posterior clasificación, para lograr una precisión adecuada se entrenó un modelo de clasificación de frutas, para el cual se usó un Dataset de 230000 imágenes, Datasets de menor número de imágenes no alcanzaban un reconocimiento esperado.

En la ejecución del modelo se tuvo errores en la distribución de los datos con relación al entrenamiento con imágenes ideales, se logro mitigarlos con la generación de datos artificiales en condiciones reales, logrando mejores precisiones de reconocimiento.

Este proyecto presento una solución automatizada para la selección de frutas dadas por un usuario, el cual, debe seleccionarlas a través de una interfaz gráfica, y con la integración de sensores (fotoeléctricos y magnéticos), y actuadores neumáticos, el modelo es capaz de cumplir la orden generada.

7. REFERENCIAS

Kim, I.-J., & Xie, a. X. (s.f.). *Handwritten Hangul recognition using deep convolutional neural networks*. Obtenido de http://www.ics.uci.edu/~xhx/publications/HHR.pdf

Scherer, D., M¨uller, A., & Behnke, S. (2010). *Evaluation of Pooling Operations in Convolutional Architectures for Object Recognition*. Obtenido de https://www.ais.uni-bonn.de/papers/icann2010_maxpool.pdf

Wu, D. J. (Mayo de 2012). *END-TO-END TEXT RECOGNITION WITH CONVOLUTIONAL NEURAL NETWORKS*. Obtenido de https://www.cs.utexas.edu/~dwu4/papers/HonorThesis.pdf

HE, Kaiming, et al. (2016). *DEEP RESIDUAL LEARNING FOR IMAGE RECOGNITION*. *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016. p. 770-778.