Instituto Tecnológico de Costa Rica

Carrera de Ingeniería Mecatrónica

Diseño de sistema de análisis y reporte de fallas mecánicas en inspección visual de chips basado en inteligencia artificial

Informe de Proyecto de Graduación para optar por el título de Ingeniero en Mecatrónica con el grado académico de Licenciatura

Jeremy Alexander Fuentes Araya

Cartago, marzo de 2022

Declaratoria de autenticidad

Declaro que el presente Proyecto de Graduación ha sido realizado enteramente por mi persona, utilizando y aplicando literatura referente al tema, introduciendo conocimientos propios y asesoramiento técnico de miembros de Qorvo, Inc.

En los casos en que he utilizado bibliografía, he procedido a indicar las fuentes mediante las respectivas citas bibliográficas.

En consecuencia, asumo la responsabilidad total por el trabajo de graduación realizado y por el contenido del correspondiente informe final.

Jeremy Alexander Fuentes Araya

Céd: 3 0518 0550

Cartago, marzo de 2022

INSTITUTO TECNOLÓGICO DE COSTA RICA
PROGRAMA DE LICENCIATURA EN INGENIERÍA MECATRÓNICA
PROYECTO FINAL DE GRADUACIÓN
ACTA DE APROBACIÓN

El profesor asesor del presente trabajo final de graduación, indica que el documento presentado por el estudiante cumple con las normas establecidas por el programa de Licenciatura en Ingeniería Mecatrónica del Instituto Tecnológico de Costa Rica para ser defendido ante el jurado evaluador, como requisito final para aprobar el curso Proyecto Final de Graduación y optar así por el título de Ingeniero(a) en Mecatrónica, con el grado académico de Licenciatura.

Estudiante: Jeremy Alexander Fuentes Araya

Proyecto: Diseño de sistema de análisis y reporte de fallas mecánicas en inspección visual de chips basado en inteligencia artificial

FELIPE GERARDO MEZA OBANDO (FIRMA)

Digitally signed by FELIPE GERARDO MEZA OBANDO (FIRMA)
Date: 2022.03.08 20:37:33 -06'00'

MSc.-Ing. Felipe Meza Obando

Asesor

Cartago, 7 de marzo 2022

INSTITUTO TECNOLÓGICO DE COSTA RICA
PROGRAMA DE LICENCIATURA EN INGENIERÍA MECATRÓNICA
PROYECTO FINAL DE GRADUACIÓN
ACTA DE APROBACIÓN

Proyecto final de graduación defendido ante el presente jurado evaluador como requisito para optar por el título de Ingeniero(a) en Mecatrónica con el grado académico de Licenciatura, según lo establecido por el programa de Licenciatura en Ingeniería Mecatrónica, del Instituto Tecnológico de Costa Rica.

Estudiante: Jeremy Alexander Fuentes Araya

Proyecto: Diseño de sistema de análisis y reporte de fallas mecánicas en inspección visual de chips basado en inteligencia artificial

Miembros del jurado evaluador

JUAN LUIS CRESPO MARIÑO (FIRMA)
Firmado digitalmente por JUAN LUIS CRESPO MARIÑO (FIRMA)
Fecha: 2022.03.07 17:30:08 -06'00'

Dr. -Ing. Juan Luis Crespo Mariño

Jurado

JUAN CARLOS BRENES TORRES (FIRMA)
Firmado digitalmente por JUAN CARLOS BRENES TORRES (FIRMA)
Fecha: 2022.03.07 16:54:41 -06'00'

MSc. -Ing. Juan Carlos Brenes Torres

Jurado

ANA MARIA MURILLO MORGAN (FIRMA)
Firmado digitalmente por ANA MARIA MURILLO MORGAN (FIRMA)
Fecha: 2022.03.07 16:41:01 -06'00'

Ing. Ana María Murillo Morgan

Jurado

Los miembros de este jurado dan fe de que el presente proyecto final de graduación ha sido aprobado y cumple con las normas establecidas por el programa de Licenciatura en Ingeniería Mecatrónica.

Cartago, 7 de marzo 2022

Resumen

El proceso de recolección y empaquetado de chips dentro de la empresa Qorvo, Inc, genera un aproximado de 41 177 imágenes de fallos mecánicos por día que deben ser clasificadas para poder determinar si hay algún riesgo de calidad en el material procesado. Dicho proceso se lleva a cabo manualmente actualmente, por lo cual esta saturado y con la inminencia de un cuello de botella en la disposición en caso de que la producción aumente.

El presente documento detalla una solución que solventa este problema al diseñar un sistema de clasificación automático de fallo basado en inteligencia artificial, utilizando paradigmas de aprendizaje automático. Adicionalmente, dicho sistema cuenta con una arquitectura basada en internet de las cosas que permite consultar, almacenar y visualizar datos del material procesado, alertar a los departamentos respectivos en caso de un problema de calidad o problema mecánico en la máquina y mostrar la información del sistema de clasificación. Por último, se desarrolla un experimento para validar el sistema y realizar una comparativa con el sistema actual de clasificación.

Palabras clave: *Aprendizaje Automático, Clasificación, CNN, Inteligencia Artificial, PCA, Python.*

Abstract

The process of collecting and packaging chips within the company Qorvo, Inc, generates an approximate of 41,177 images per day of mechanical failures that must be classified in order to determine if there is any quality risk in the processed material. This process is currently carried out manually, which is why it is saturated and with the imminence of a bottleneck in the disposal in case production increases.

This document details a solution that solves this problem by designing an automatic classification system based on artificial intelligence, using machine learning paradigms. Additionally, the system has an architecture based on the internet of things that allows to consult, store and visualize data of the processed material, alert the corresponding departments in case of a quality problem or mechanical problem in the machine and display the information of the classification system. Finally, an experiment is developed to validate the system and make a comparison with the current classification system.

Keywords: *Machine Learning, Classification, CNN, Artificial Intelligence, PCA, Python.*

Dedicado a mi familia que me han apoyado incondicionalmente a lo largo de estos años

Agradecimientos

Primeramente, quisiera agradecer a mi madre Lorena por su apoyo incondicional durante todos mis estudios, a mis hermanos Jonathan y Jenifer por su gran apoyo y consejos a lo largo de todos estos años.

También quiero extenderle mis agradecimientos a todos mis compañeros, especialmente Steven y Lucía que me ayudaron a lo largo de la carrera.

De igual manera, quiero darle un especial agradecimiento al Ing. Elmer Arce Arce por darme la oportunidad de realizar este proyecto dentro de las instalaciones de Qorvo y por su apoyo en el proceso.

Finalmente, agradezco a todos los profesores de la carrera de ingeniería Mecatrónica que me guiaron y aportaron en mi formación durante todos estos años.

Lista de contenidos

Lista de figuras

Lista de tablas

Lista de abreviaciones

ANN	Redes neuronales artificiales
API	Interfaz de programación de aplicaciones
CNN	Redes neuronales convolucionales
DER	Desviación estándar relativa
DTR	Empaque de dispositivos
DTR Map	Inspección de fallos mecánicos por rollo
IA	Inteligencia artificial
NFS	Sistema de archivos de red
PCA	Análisis de componentes principales
RAM	Memoria de acceso aleatorio
ROI	Retorno sobre inversión
SGD	Optimizador de gradiente descendiente con momentum
SMB	Bloque de mensajes del servidor
SMTP	Protocolo simple de transferencia de correo
SQL	Lenguaje de consulta estructurado
TIR	Indicador total de retorno
VAN	Valor actual neto
WCF	Fundación de Comunicación de Windows

Capítulo 1: Introducción

1.1. Entorno del proyecto

Este proyecto se realiza en la empresa Qorvo, Inc., en el centro de manufactura y testeo de Costa Rica, ubicado en Barreal de Heredia. La empresa produce cientos de componentes SAW (Surface Acoustic Wave) y BAW (Bulk Acoustic Wave), que son filtros de radio frecuencia (RF) que se basan en ondas de acústicas, con frecuencias que van desde 10 MHz a 8 GHz que brinda soluciones para móviles, infraestructura y aplicaciones de la industria aeroespacial y defensa. Esta es una empresa de manufactura de tecnología de avanzada, como duplexers, filtros RF 2-en-1, etc. La compañía busca la mejora continua de sus operaciones, de los planes de mantenimiento, inspección de equipos, de las técnicas de solución de problemas recurrentes y los sistemas que conforman el proceso productivo [23].

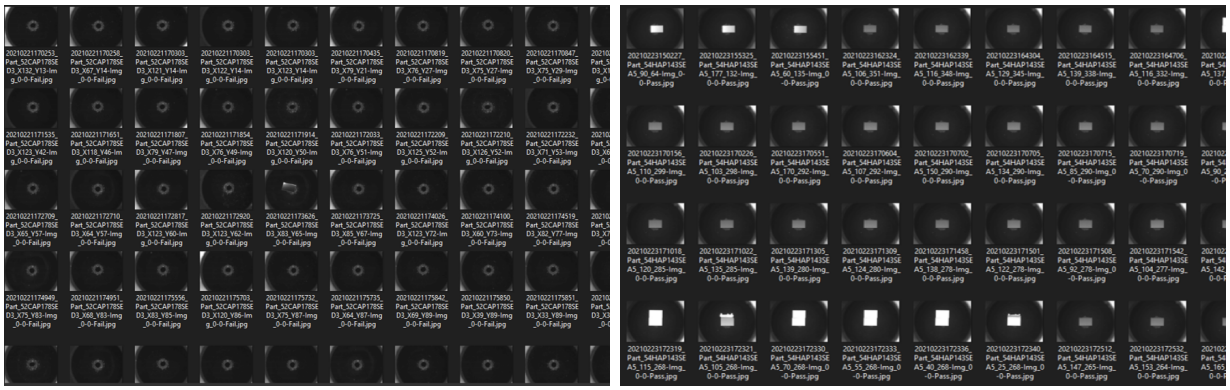
El proyecto se realiza en la inspección de calidad del proceso de extracción y empaque de dispositivos o Die To Reel (DTR), el cual consiste en extraer las piezas chips semiconductores o "Dies" de un dispositivo llamado wafer (contienen cientos o miles de estos chips) para posteriormente ponerlos individualmente en una cinta (Tape) para poder empacarse [6]. Este proceso se lleva a cabo en dos líneas de producción diferentes (dos tipos diferentes de tecnologías de chips), estas se llamarán **Línea A** y **Línea B** en el resto del documento; es importante mencionar que mecánicamente son muy similares, pero su proceso de manufactura y material es diferente.

Específicamente, se trabaja en el área de disposiciones de la empresa en el Departamento de Ingeniería de Producto (Ing. de Producto) que lleva a cabo la inspección de calidad del

proceso de recolección. Esta inspección visual de las fallas mecánicas encontradas en el proceso de DTR se llama *DTR map* y consiste en una inspección del 100 % de las imágenes que se generan en la estación 1 del proceso de recolección y empaquetado de chips (en cada una de sus estaciones se toma una foto del estado del chip). Es importante mencionar que en todo el documento se hace referencia únicamente a las líneas (Línea A y B) y no al tipo de tecnología por requerimientos de confidencialidad de la empresa.

1.2. Problema existente

El problema radica en el proceso de inspección visual de las imágenes de fallos mecánicos en la estación 1 del proceso de recolección y empaquetado de chips, el cual hace que los operarios y técnicos ocupen todo su día laboral solo en esta tarea por el alto volumen de datos por inspeccionar.



(a) Ejemplo de rollo con alta incidencia de **boquillas vacías**.

(b) Ejemplo de rollo con alta incidencia de **piezas descentradas**.

Figura 1.1: Ejemplo de alta incidencia de fallos falsos en sistema de visión del proceso de DTR. Elaboración propia.

Además, en estas inspecciones visuales hay una cantidad muy significativa de fallos falsos (que no representan una condición cuestionable de calidad) en la estación 1 y, como se guardan las imágenes con diferentes contrastes (la misma imagen varias veces), la cantidad de imágenes que son fallos marginales es más del **80 %** según un muestreo realizado en los rollos al estudiar las carpetas del servidor de imágenes. Un ejemplo de esto se muestra en la Figura 1.1 donde se aprecia que la mayoría son solo dos tipos de fallos específicos que no provienen de la pieza o Die,

sino de fallas de la máquina de DTR. Por lo tanto, son marginales para esta inspección de calidad y representan un problema que requiere acciones específicas para reponer el comportamiento normal de la máquina. A estos fallos se les denomina **boquilla vacía** y **pieza descentrada** respectivamente y se detallan en las siguientes secciones.

Según un estudio realizado en los meses de febrero y marzo del 2021 que se resume en la Tabla 1.1, se puede observar que se inspeccionaron en promedio un total de ***41 177 imágenes al día***, lo cual genera un gran consumo de recurso humano. Para estos meses, se dispuso de un técnico y un operario mediante 12 h diarias en esta inspección, lo cual se ha mantenido a lo largo del año.

Tabla 1.1: Resumen de incidencia de fallos en la estación 1 del proceso en los meses de febrero y marzo del 2021. Elaboración propia.

Mes	Línea B	Línea A	Total	Días	Promedio/día
feb	732074	328676	1060750	28	37884
mar	719381	649335	1368716	31	44152
Total general	1451455	978011	2429466	59	41177

Posteriormente, al analizar el crecimiento del proceso de DTR (producción) desde setiembre a la actualidad se puede observar un crecimiento significativo en la producción y se proyecta que esta continúe creciendo. Con esto se genera un cuello de botella en este proceso, porque con la tendencia en aumento de la producción llegará un momento en que los tiempos de ciclo de cada rollo es muy grande, lo cual puede generar problemas con los embargues y con los clientes.

1.2.1. Síntesis del problema

El problema se puede establecer como la inminencia de un cuello de botella en la disposición del producto en el proceso de DTR, esto por la necesidad de rediseñar el procedimiento de inspección visual para verificar la calidad este proceso, específicamente en la estación 1. Por esto, se debe buscar una solución para disminuir la carga de trabajo de las personas colaboradoras de la empresa.

1.3. Objetivos

1.3.1. Objetivo General

Diseño de un sistema de clasificación, evaluación y reporte de fallos mecánicos, que se basa en inteligencia artificial, en la inspección visual de chips de la estación 1 del proceso de recolección y empaquetado de chips (DTR) de las líneas de producción de Qorvo, Inc.

1.3.2. Objetivos específicos

1. Generar un estudio de los fallos mecánicos en el proceso de recolección y empaquetado de chips (DTR), específicamente en la estación 1.
2. Diseño de un sistema automatizado de clasificación de fallos mecánicos que se basa en inteligencia artificial en la estación 1 del proceso de recolección y empaquetado de chips.
3. Diseño de un sistema de evaluación automática del estado de los rollos y de la máquina en el proceso de recolección y empaquetado de chips.
4. Generación de un estudio de funcionalidad del sistema de clasificación de fallos en referencia al sistema de clasificación actual.

Capítulo 2: Marco Teórico

2.1. Generalidades del proceso de recolección, empaque e inspección visual de chips

El proyecto se llevará a cabo en el proceso de *Die To Reel* (DTR), el cual consiste en extraer las piezas llamadas *Die* o chips semiconductores de un dispositivo llamado *wafer* (contienen cientos o miles de estos chips), para posteriormente ponerlos individualmente en una cinta (*Tape*) para poder ser empacados [6], esto mediante una máquina que denominaremos 'Máquina DTR', la cual se detalla en secciones posteriores junto a su funcionamiento, esto se ejemplifica en la figura 2.1. Este proceso se realiza en dos líneas de producción diferentes (dos tipos diferentes de tecnologías de chips), estas se llamarán **Línea A** y **Línea B** en el resto del documento, es importante mencionar que mecánicamente son muy similares pero su proceso de manufactura y material son diferentes.

La empresa produce cientos de diferentes componentes SAW (*surface acoustic wave* o onda acústica superficial) y BAW (*bulk acoustic wave* o onda acústica de magnitud), que son filtros de radio frecuencia (RF) basados en ondas de acústicas, con frecuencias que van desde 10 MHz a 8 GHz que brinda soluciones para móviles, infraestructura y aplicaciones de la industria aeroespacial/defensa. Es una empresa de manufactura de tecnología de avanzada, tales como duplexers, filtros RF 2-en-1, etc. La empresa busca la mejora continua de sus operaciones, de los planes de mantenimiento, inspección de equipos, de las técnicas de solución de problemas recurrentes y los sistemas que conforman el proceso productivo [23].

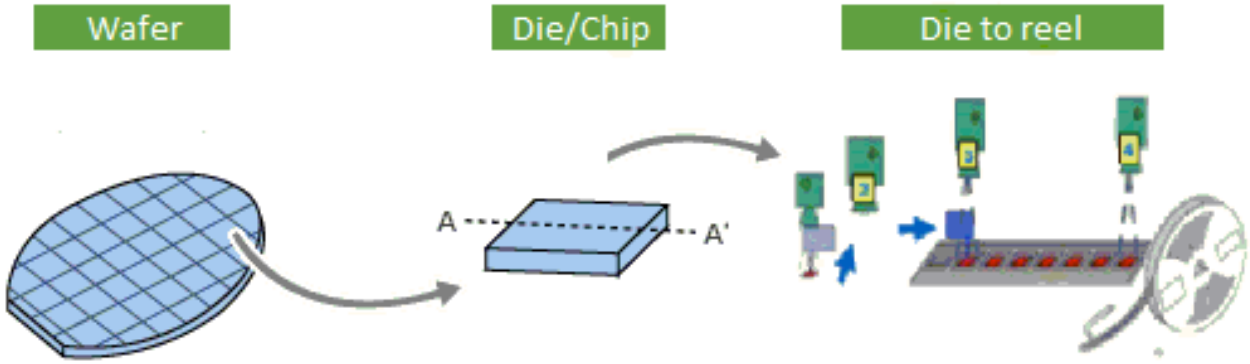


Figura 2.1: Esquematico general del proceso de recolección y empaque de chips. Elaboración propia.

En dicha máquina denominada 'Máquina DTR' ocurre una inspección visual de los chips para evaluar si existen fallas mecánicas, en dicho caso, las piezas se descartan del resto del proceso. Al detectar una falla mecánica (por medio de un sistema de visión integrado en la máquina) toma varias fotografías en varios contrastes de la pieza y la envía a un servidor.

Con esto el técnico u operario del área de disposiciones mediante una aplicación que le permite ver los lotes, cantidad de pieza, tipo de producto y otros aspecto, accede a la carpeta con estas imágenes, la cual se encuentra segmentada por la estación de la 'Máquina DTR' en donde se encontró el fallo; aquí el técnico u operario revisa individualmente el 100 % de las imágenes generadas en la estación 1 del proceso de recolección y empaquetado de chipsen en busca de fallos catastróficos, lo que deriva en que haya riesgo de que el rollo cuente con fallos que no haya detectado la máquina, por lo tanto, amerita enviar el lote a una inspección visual física de la cinta. Dicho proceso se denomina 'DTR Map'. Este proceso se muestra el diagrama de flujo del proceso en la figura 4.4.

Cabe recalcar que en todo el documento se hará referencia únicamente a las líneas de esta manera (Línea A y B) y no con qué tipo de tecnología, esto por requerimientos de confidencialidad de la empresa, esto se hará con diferentes datos a lo largo del documento.

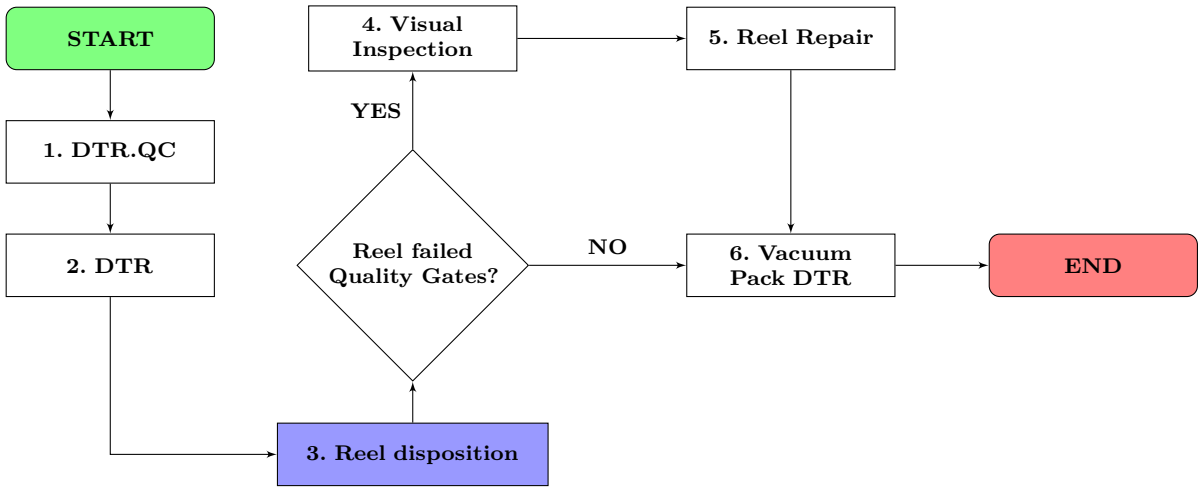


Figura 2.2: Diagrama de flujo del proceso de DTR para la **Línea A** y **Línea B**. Obtenido de [6].

2.2. Automatización industral

Actualmente, la automatización de los procesos industriales es uno de los objetivos más importantes de las empresas en la siempre incesante tarea de la búsqueda de la competitividad en un entorno cambiante y agresivo, tal y como lo menciona [19]. La automatización de un proceso industrial consiste en la incorporación de un conjunto de elementos y dispositivos tecnológicos al proceso para así aseguren su control y buen comportamiento.

En general, la automatización ha de ser capaz de reaccionar frente a las situaciones previstas de antemano y además frente a imponderables, además tener como objetivo volver más eficiente el proceso y reubicar los recursos humanos de manera que se óptimice no solo el proceso en cuestión sino también los adyacentes [19].

2.3. Inteligencia Artificial

Según [26], la inteligencia artificial o IA se puede describir como capacidad de las máquinas para usar algoritmos, aprender de los datos y utilizar lo aprendido en la toma de decisiones tal y como lo haría un ser humano. Sin embargo, los dispositivos basados en IA pueden trabajar sin descanso y analizan grandes volúmenes de información simultáneamente.

Asimismo, los sistemas basados en IA tienen significativamente menos errores al realizar las mismas tareas que sus contrapartes humanas.

Los sistemas basados en inteligencia artificial tiene una gama muy grande de posibles aplicaciones, pero según lo menciona [26], algunas de las aplicaciones técnicas que han crecido rápidamente en la actualidad son:

- Reconocimiento de imágenes estáticas, clasificación y etiquetado.
- Mejoras del desempeño de la estratégica algorítmica comercial.
- Procesamiento eficiente y escalable de datos de pacientes.
- Mantenimiento predictivo.
- Detección y clasificación de objetos.
- Distribución de contenido en las redes sociales.
- Protección contra amenazas de seguridad cibernética.

2.3.1. Aprendizaje automático

Uno de los principales paradigmas de la inteligencia artificial es el aprendizaje automático o *Machine Learning* y consiste en la utilización de algoritmos para aprender de los patrones de los datos y tomar decisiones con el conocimiento adquirido [26].

El aprendizaje automático a su vez se puede subdividir en 3 paradigmas o tipos de aprendizaje, estos son aprendizaje supervisado, no supervisado y aprendizaje de refuerzo. Esta subdivisión se ejemplifica en la Figura 2.3.

Figura 2.3: Tipos de aprendizaje automático. Obtenido de [26].

El **aprendizaje supervisado**, consiste en la utilización de datos que ya han sido previamente etiquetados u organizados para indicarle al algoritmo como debe ser categorizada la nueva información [26].

En lo que respecta a **aprendizaje no supervisado**, se puede definir como el uso de algoritmos que no usan ningún dato de etiquetado u organizado previo para indicar como se tendría que categorizar la nueva información, sino que estos algoritmos encuentran la forma de clasificar ellos mismos [26].

Por último, el **aprendizaje por refuerzo** es tipo de algoritmo que aprende por experiencia, en palabras simples, es un algoritmo al que se le da un reforzamiento positivo cada vez que acierta. La forma en que estos algoritmos aprenden es muy similar a la forma en que se entrenan a los perros actualmente, al darles premios cada vez que hacen una acción a la que consideramos correcta.

2.3.2. Redes neuronales convolucionales

Para este proyecto es necesario explicar lo que son las redes neuronales convolucionales o CNN, que es uno de los paradigmas más comunes de aprendizaje supervisado actualmente, el cual se utiliza normalmente para clasificación de imágenes. Las redes neuronales convolucionales son análogas a las tradicionales redes neuronales artificiales o ANN, ya que están compuestas por neuronas que se autooptimizan a través del aprendizaje. Cada neurona aún recibirá una entrada y realizará una operación (como un producto escalar seguido de una función no lineal), la base de las ANN [20].

Desde los vectores de imagen sin procesar de entrada hasta la salida final de la puntuación de la clase, toda la red sigue expresando una única función de puntuación perceptiva (el peso). La última capa contiene funciones de pérdida asociadas con las clases, y aún se aplican todos los consejos y trucos regulares desarrollados para las ANN tradicionales [20].

Sobreajutes o *Overfitting*

El sobreajuste es básicamente cuando una red no puede aprender correctamente, ya sea porque no encontró un patrón adecuado para clasificar o cuando se ajusto demasiado al conjunto de datos de entrenamiento, por lo que al tratar de interpretar nueva información no va a poder clasificar adecuadamente; una representación de esto se muestra en la Figura 2.4.

Este es un concepto importante en todos los algoritmos de aprendizaje automático y es importante que se tomen todas las consideraciones necesarias para reducir sus efectos. Si un modelos muestra signos de sobreajuste, tendrá una capacidad reducida para identificar características generalizadas no solo al conjunto de datos de entrenamiento, sino también para nuestros conjuntos de prueba y predicción [20]. Esta es la razón principal detrás de la reducción de la complejidad de las ANN. Cuantos menos parámetros se requieran para entrenar, menos probable es que la red se sobreajuste y, por supuesto, mejore el rendimiento predictivo del modelo [20].

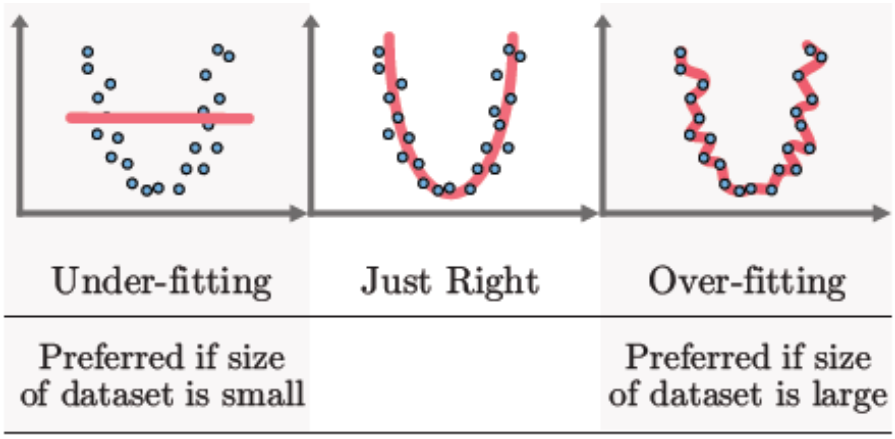


Figura 2.4: Representación del sobreajuste. Obtenido de [20].

Arquitectura de las CNN

Como se mencionó anteriormente, las CNN se centran principalmente en la base de que la entrada estará compuesta por imágenes. Esto enfoca la arquitectura que se configurará de la manera que mejor se adapte a la necesidad de tratar con el tipo específico de datos. Una de las diferencias clave es que las neuronas que forman las capas dentro de la CNN están compuestas de neuronas organizadas en tres dimensiones, la dimensionalidad espacial de la entrada (alto y ancho) y la profundidad [20].

Las CNN se componen de tres tipos de capas. Estas son capas de neuronas convolucionales, capas de agrupación y capas totalmente conectadas. Cuando estas capas se apilan, se ha formado una arquitectura CNN. Un ejemplo de una arquitectura básica se muestra en la Figura 2.5.

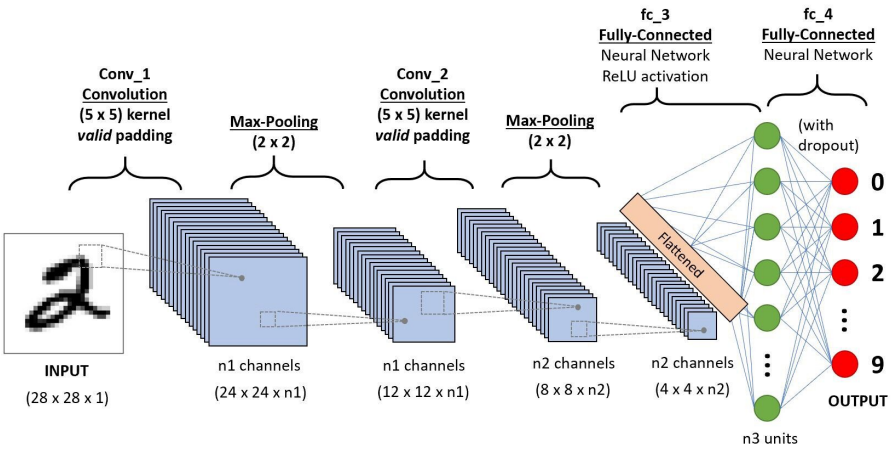


Figura 2.5: Una arquitectura de CNN simple. Obtenido de [20].

Las arquitecturas de CNN se componen de diferentes capas. A continuación se describen las más comúnmente usadas:

- **La capa de neuronas convolucional**: determina la salida de las neuronas que están conectadas a las regiones locales de la entrada a través del cálculo del producto escalar entre sus pesos y la región conectada al volumen de entrada. La unidad lineal rectificada (comúnmente abreviada como 'ReLu') tiene como objetivo aplicar una función de activación 'elemento' como sigmoide a la salida de la activación producida por la capa anterior [20].
- **La capa de agrupación**: realiza un muestreo descendente a lo largo de la dimensionalidad espacial de la entrada dada, reduciendo aún más la cantidad de parámetros dentro de esa activación [20].
- **Las capas completamente conectadas**: realizan las mismas tareas que se encuentran en las ANN estándar e intentarán producir puntajes de clase a partir de las activaciones, que se utilizarán para la clasificación [20]. Comúnmente, también se utilizan en el ultimo segmento de la arquitectura donde las neuronas de salida son el número de salidas del sistema, esto para obtener el resultado final.

2.3.3. Análisis de componentes principales

El análisis de componentes principales o PCA es un método de aprendizaje automático no supervisado, el cual, tal y como lo describe [9], consiste en reducir la dimensionalidad de un espacio original que se establece a partir de un conjunto de datos de las variables medidas de un proceso mediante su proyección de un espacio de menor dimensionalidad. El PCA busca combinaciones lineales de las variables que mejor describen la tendencia del proceso, el cual se basa en una descomposición de la matriz de covarianza de las variables del proceso a lo larfo de las direcciones que mejor explican las principales causas de variabilidad de la información analizada.

Según [9], la utilidad de PCA se puede describir en dos puntos:

1. Permite representar óptimamente en un espacio de dimensión pequeña observaciones de un espacio general p-dimensional. En este sentido, el análisis de componentes principales es el primer paso para identificar las posibles variables latentes, o no observadas que generan los datos.

2. Permite transformar las variables originales, en general correlacionadas, en nuevas variables incorrelacionadas, facilitando la interpretación de los datos.

El análisis de componentes principales parte de una matriz Y de dimensiones nxm (Ecuación 2.1, donde p corresponde al número de variables observadas y m al número de individuos o unidades de observación; en caso de aplicarlo a imágenes como en este proyecto n y m son el número de píxeles de la imagen. Es importante mencionar que la matriz Y debe estar estandarizada .

$$Y = \begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1m} \\ x_{21} & x_{22} & \cdots & x_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ x_{n1} & x_{n2} & \cdots & x_{nm} \end{bmatrix} \tag{2.1}$$

Con esta matriz se realiza la transformación mostrada en la Ecuación 2.2, obteneniendo así la matriz X, donde $\bar{y}_j$ y s_j son el promedio y la desviación estándar para cada una de las variables [36].

$$x_{ij} = \frac{y_{ij} - \bar{y}_j}{s_j \sqrt{n}} \tag{2.2}$$

Luego de esto, se determinan los valores propios y vectores propios mediante la matriz de coorelación o covarianzas R, con la cual se puede llegar a relación la cantidad de información obtenida en función de la cantidad de componentes de la reducción, esto porque los valores propios corresponde a la varianza de las observaciones de cada uno de los componentes (nuevas variables) y los elementos de los vectores propios corresponden a las coordenadas en el espacio inicial que dan la dirección de los componentes principales (Z) [9]. Esta matriz de R se calcula mediante la Ecuación 2.3.

$$R = \frac{1}{n-1} X^T X \tag{2.3}$$

Con estas matrices, finalmente se puede obtener la matriz de dimensiones reducida, la cual es el resultado de PCA. Esta se obtiene mediante el calculo de la Ecuación 2.4 [9].

$$Z = XR \tag{2.4}$$

A partir de esta mátrices R y Z tambien se puede volver al espacio de variables de partida, esto mediante la Ecuación 2.6 [9]. Siendo E la matriz de residuos que resulta de la diferencia entre X y $\bar{X}$, esto se representa en la Ecuación 2.5.

$$E = X - \bar{X} \tag{2.5}$$

$$Y = ZR^T + E \tag{2.6}$$

2.4. Internet de las cosas

Según [27], IoT (*Internet of things*/Internet de las cosas) se puede definir como una arquitectura emergente basada en la Internet global que facilita el intercambio de bienes y servicios entre redes de la cadena de suministro y que tiene un impacto importante en la seguridad y privacidad de los actores involucrados.

La IoT se refiere a la interconexión en red de todos los objetos cotidianos, que a menudo están equipados con algún tipo de inteligencia. En este contexto, Internet puede ser también una plataforma para dispositivos que se comunican electrónicamente y comparten información y datos específicos con el mundo que les rodea. Así, la IoT puede verse como una verdadera evolución de lo que conocemos como Internet añadiendo una interconectividad más extensa,

una mejor percepción de la información y servicios inteligentes más completos [27].

Una arquitectura de IoT esta basada o compuesta por 4 fases, según [30]:

1. **Fase 1 sensores y actuadores** : Es donde se supervisa o controla algún proceso físico.
2. **Fase 2 Sistema de adquisición de datos** : Es donde se convierten los datos a un valor digital.
3. **Fase 3 Preprocesamiento** : En esta etapa los datos se procesan para reducir su volumen antes de que vayan a la nube.
4. **Fase 4 Análisis en profundidad** : Es donde se analizan los datos para detectar patrones, tendencias o anomalías de los procesos. A este punto las empresas pueden tomar decisiones seguras respaldadas con la información recolectada.

2.4.1. Bases de datos

Una base de datos es una recopilación organizada de información o datos estructurados, que normalmente se almacena de forma electrónica en un sistema informático. Normalmente, una base de datos está controlada por un sistema de gestión de bases de datos (DBMS) [21].

El SQL es un lenguaje de programación que utilizan casi todas las bases de datos relacionales para consultar, manipular y definir los datos, además de para proporcionar control de acceso [21].

Según [21], existen diferentes tipos de bases de datos, los cuales se describen a continuación:

- **Bases de datos relacionales**: Los elementos de una base de datos relacional se organizan como un conjunto de tablas con columnas y filas.
- **Bases de datos orientadas a objetos**: La información de una base de datos orientada a objetos se representa en forma de objetos, como en la programación orientada a objetos.

- **Bases de datos distribuidas**: Una base de datos distribuida consta de dos o más archivos que se encuentran en sitios diferentes. La base de datos puede almacenarse en varios ordenadores, ubicarse en la misma ubicación física o repartirse en diferentes redes.
- **Almacenes de datos**: Un repositorio central de datos, un data *warehouse* es un tipo de base de datos diseñado específicamente para consultas y análisis rápidos.
- **Bases de datos NoSQL**: Una base de datos NoSQL, o base de datos no relacional, permite almacenar y manipular datos no estructurados y semiestructurados (a diferencia de una base de datos relacional, que define cómo se deben componer todos los datos insertados en la base de datos).
- **Bases de datos orientadas a grafos**: Una base de datos orientada a grafos almacena datos relacionados con entidades y las relaciones entre entidades.
- **Bases de datos OLTP**: Una base de datos OLTP es una base de datos rápida y analítica diseñada para que muchos usuarios realicen un gran número de transacciones.

2.5. Medidas de evaluación

Como se mencionó anteriormente, los sistema de aprendizaje automático, necesita una forma de poder visualizar si el entrenamiento se dio correctamente y si existe un sobreajuste. Para esto se tiene las curvas de aprendizaje, las cuales muestran la relación entre los valores de entrenamiento y validación en función del número de generaciones de entrenamiento [1]. Esto se ejemplifica en la Figura 2.6, donde se resalta el punto donde empezó un sobreajuste.

Figura 2.6: Curva de aprendizaje con presencia de sobreajuste. Obtenido de [1].

Según [1], los algoritmos de entrenamiento de clasificadores, en su implementación interna dividen el conjunto de datos de entrenamiento en dos partes, una destinada únicamente para entrenar al modelo, y otra para validarlo. De esta forma, el puntaje de entrenamiento muestra qué tan efectivo resulta el modelo al ser probado con el set de datos de entrenamiento; y el puntaje de validación que tan bueno resulta el modelo con el otro subconjunto de datos. Dependiendo si las curvas de validación y entrenamiento convergen, se puede determinar si el clasificador se encuentra sobre-ajustado o sobreentrenado.

Otra de las maneras más comunes para evaluar los sistemas de aprendizaje automático es mediante las matrices de cofusión, con la cual se puede observar el desempeño del algoritmo. Cada columna de la matriz representa el número de predicciones de cada clase, mientras que cada fila representa a las instancias en la clase real. Uno de los beneficios de las matrices de confusión es que facilitan ver si el sistema está confundiendo clases [1].Un ejemplo de esta matriz se muestra en la Figura 2.7.

Figura 2.7: Matriz de confusión para un sistema binario. Obtenido de [1].

Como se puede observar, la medición de la precisión de un sistema de esta naturaleza no esta influenciado únicamente por que tantas buenas predicciones se realizaron con respecto al número total, sino también con respecto a su clase original. Para explicar esto, se presentan a continuación las medidas de error más utilizadas, tomando en cuenta que la nomenclatura de $TP, TN, FPyFN$ que significan verdaderos positivos, verdaderos negativos, falsos positivos y falsos negativos respectivamente [1]:

- **Exactitud**: Número total de predicciones correctas sobre el número total de predicciones ejecutadas, tanto para una, como para otra clase.

$$Exactitud = \frac{TP + TN}{TP + TN + FP + FN} \tag{2.7}$$

- **Precisión**: Número total de predicciones correctas sobre el número total de predicciones ejecutadas, para una misma clase. Generalmente la precisión es evaluada sobre la clase positiva.

$$Exactitud = \frac{TP}{TP + FP} \tag{2.8}$$

Capítulo 3: Metodología

En este capítulo del documento se describe la metodología que se utiliza para solucionar el problema descrito en la sección 1.2, en este se describen las tareas definidas por cada objetivo específico mostrado en la sección 1.3.2 que conducirá el proceso del desarrollo del proyecto.

3.1. Enfoque metodológico

Para desarrollar este proyecto se utiliza la metodología de diseño de ingeniería que plantearon Ulrich y Eppinger [35]. A partir de esta se extraen los pasos que se seguirán en el desarrollo de este proyecto, estos se detallan a continuación y se representan en el diagrama de flujo mostrado en la Figura 3.1. Es importante mencionar que cada una de estas etapas implica un proceso de reflexión para verificar que el enfoque metodológico se utiliza correctamente.

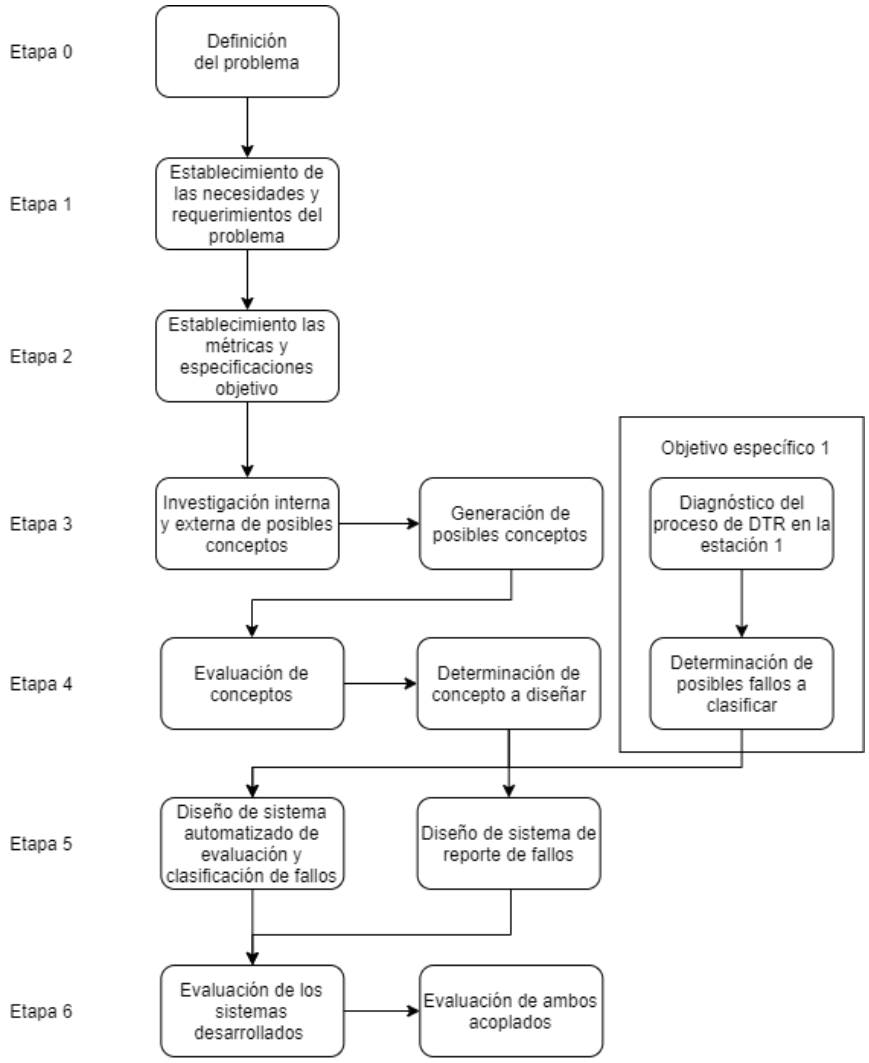


Figura 3.1: Diagrama de flujo de las fases que se plantearon para el desarrollo del proyecto. Elaboración propia.

3.1.1. Etapa 1: Identificación y determinación de necesidades

Primero, se determina de manera adecuada y completa el conjunto de necesidades del cliente con respecto al problema establecido. Para esto se debe realizar una investigación mediante una recopilación y análisis de declaraciones del cliente, fuentes externas, normativas, protocolos y regulaciones que sean aplicables al problema.

Una vez determinado las necesidades del cliente, se procede a determinar la importante de cada una de estas para, posteriormente, determinar métricas necesarias para evaluar la totalidad de las necesidades, esto mediante reuniones con los departamentos involucrados y el criterio propio.

3.1.2. Etapa 2: Establecimiento de especificaciones objetivo

Después de determinar las necesidades del problema se establecen las especificaciones que permiten validar la idoneidad de los conceptos.

Para establecer dichas especificaciones se debe realizar un experimento que evalué el comportamiento del sistema actual de clasificación. Esto para poder definir un conjunto de valores y rangos (objetivos y marginales) para cada una de las necesidades, las cuales deben abarcar una mejoría (se establece con los requerimientos del sistema) con respecto al sistema actual y cumplir con las expectativas del cliente respecto las características críticas del sistema (en caso de considerarse necesario estas pueden ser establecidas por este).

En base a esto se toman diferentes acciones para poder llevar a cabo esta últimas dos etapa, las actividades que se muestran a continuación:

1. Programar una reunión con los miembros del departamento de la empresa.
2. Estudiar los documentos referentes al proceso.
3. Definir las necesidades, especificaciones y criterios de selección con base en la investigación realizada.

3.1.3. Etapa 3: Generación de conceptos

Primeramente, para esta etapa se procede a realizar un análisis de la naturaleza del problema a resolver para posteriormente obtener una descomposición funcional del sistema, caracterizados correctamente, que separe el problema principal en subproblemas que se puedan abordar uno por uno para solventar el problema planteado.

Una vez establecido la descomposición funcional y sus respectivos subproblemas, se debe proponer posibles soluciones para cada uno de ellos, esto en base a investigaciones internas y externas, protocolos y documentaciones de la empresa y criterio propio y de los departamentos involucrados en el proyecto. Para llevar a cabo esto se debe desarrollar las siguientes acciones:

1. Investigación interna y externa de sistemas similares.

2. Desarrollar una descomposición funcional

3. Generar ideas de conceptos para el sistema a diseñar.

3.1.4. Etapa 4: Selección de conceptos

Con base en los conceptos que se plantearon en la etapa anterior se procede a evaluarlos mediante una exploración sistemática en dos diferentes niveles.

En el primer nivel (filtrado) se determinan los elementos que justifican la aceptación, mejora, combinación o rechazo del concepto y en el segundo nivel se determina el objeto de las pruebas de concepto. Para esto, se emplea una matriz de selección con base en los criterios y especificaciones que se determinan en las etapas anteriores.

Es importante mencionar que la selección del concepto se debe realizar para cada uno de los subsistemas de la descomposición funcional, que, como se mencionó, implica una exploración sistemática mediante dos niveles.

En caso de que sea necesario, para determinar aspectos de los conceptos, se realizan pruebas mediante experimentos para evaluar diferencias entre conceptos. En estos experimentos, al ser un sistema que cuenta con enfoques de inteligencia artificial y aprendizaje automático, se debe desarrollar un conjunto de datos limitado y preliminar para entrenar sistemas preliminares que sirvan para evaluar los conceptos y ayuden en la selección de los mismos.

3.1.5. Etapa 5: Prototipado

En esta quinta etapa se desarrolla el sistema del concepto seleccionado, esto corresponde tanto para el sistema de clasificación como para el sistema de reporte. Esto corresponde al desarrollo de los algoritmos necesarios y procesos de rediseño del concepto en caso de que sea necesario.

Para esto, primeramente, se debe generar un estudio mecánico de fallos en la estación 1, sus características, cómo se distingue, sus consecuencias y qué acciones se deben tomar para resolverlo para posteriormente diseñar un sistema de clasificación. Esto se realiza mediante una investigación que analiza las necesidades, especificaciones, criterio y procedimientos del proceso mediante investigaciones en la documentación de la empresa y visitas a la empresa.

Posterior a esto, se desarrolla cada uno de los conceptos seleccionados en la etapa anterior en cada subproblema de la descomposición funcional.

3.1.6. Etapa 6: Pruebas de concepto

Con el desarrollado el concepto seleccionado se procede a una etapa de evaluación del sistema donde se evalúa la funcionalidad del sistema desarrollado en referencia con el sistema de clasificación actual.

En esta etapa se necesita que se compruebe la funcionalidad de los sistemas desarrollados. Esto se hace mediante pruebas o experimentos junto con el Departamento de Calidad de la empresa para validar el funcionamiento correcto y cumplimiento de las especificaciones del sistema. Este objetivo se desglosa en las siguientes actividades:

1. Definir los factores de influencia del ambiente sobre el cual se probará el sistema.
2. Diseñar, mediante DOE, un conjunto de pruebas que validen el sistema al someterlo a las variaciones de los factores de influencia definidos.
3. Validar el funcionamiento del sistema al experimento diseñado.
4. Documentar los resultados de las pruebas de validación.

Finalmente, se debe determinar la exactitud, precisión, tiempo de clasificación y funcionalidad del sistema desarrollado.

3.1.7. Etapa 7: Documentación del proyecto

Esta es la última etapa del proyecto, se documenta el proceso realizado en las etapas anteriores, los resultados, la documentación de los algoritmos que se utilizan y la base teórica de los conceptos utilizados. Además, se detalla el estudio económico del proyecto, conclusiones, recomendaciones de los resultados finales y cualquier información que se considere relevante.

Capítulo 4: Estudio de fallos mecánicos

Para desarrollar la solución que se planteó se debe, como parte de los objetivos, llevar a cabo un estudio formal del proceso de DTR para ambas líneas. Este análisis es de los diferentes componentes, procedimientos, maquinaria y factores que afectan los *chips* en este proceso. Para esto, se apoya en la documentación dada por la empresa (procedimientos, planos, modelos, protocolos, entre otros), Ing. de Proceso, Producto y Producción, además de los diferentes operarios para tener una visión general de todo el proceso. Todo esto para determinar los diferentes tipos de en la estación 1 (donde se enfoca este proyecto) y su procedencia para establecer en que categoría se debe clasificar el sistema por desarrollar y qué hacer con la información que se recopiló.

4.1. Descripción del proceso

Primero, se debe explicar la máquina y el proceso de recolección para explicar posteriormente en qué consiste y de dónde proviene cada fallo que se analiza en este proyecto. El punto principal de estudio es la recolección de *chips*, que es donde suceden los fallos mencionados.

Esto sucede en máquinas que se denominan DTR y se muestra en la Figura 4.1, en esta existen varias estaciones donde se llevan a cabo diferentes operaciones con las piezas.

La máquina mostrada en la Figura 4.2(b), en su sector superior izquierdo tiene un

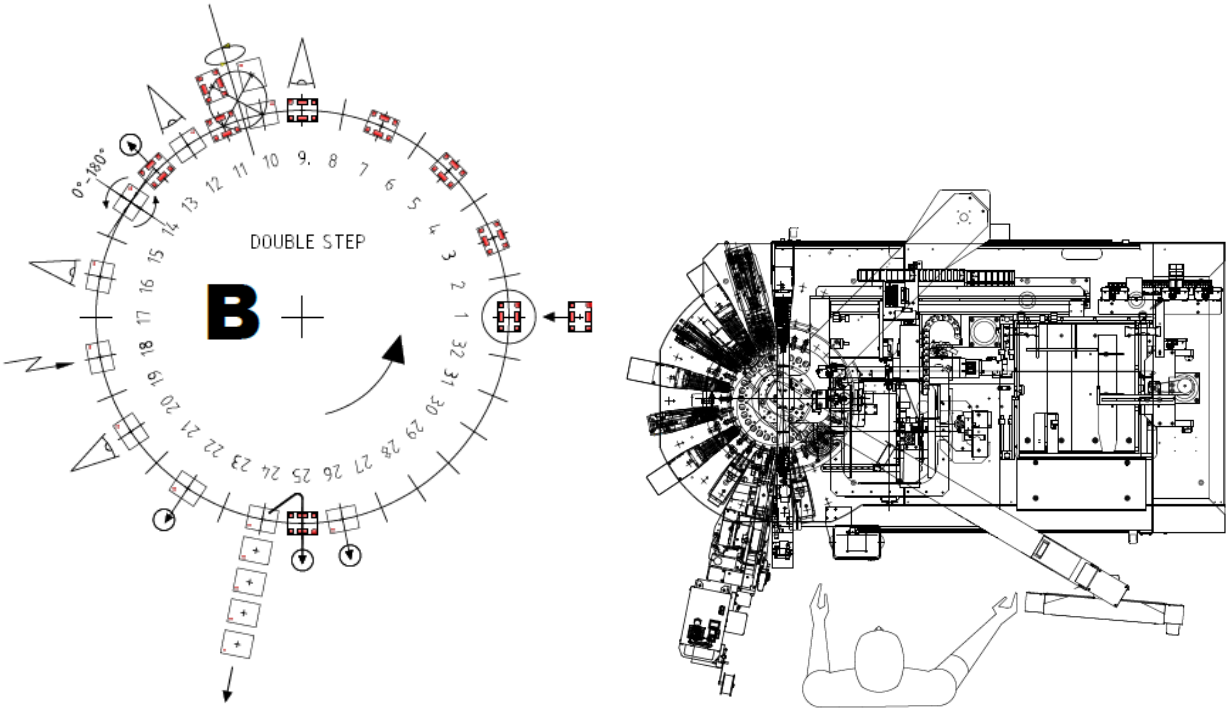
Figura 4.1: Máquina de DTR. Obtenido de [7].

subconjunto de mecanismos que consiste en varias estaciones donde se llevan a cabo varias operaciones como la recolección, inspección superior, volteado, inspección inferior y empaquetado. Este proyecto se concentra en la estación 1 (la cual se explica detalladamente a continuación) del proceso que corresponde a la recolección e inspección superior (espalda) del dispositivo.

Este subconjunto se muestra en la Figura 4.2(a), se puede observar el recorrido que hace la pieza, desde la recolección hasta el empaque. La estación 1, que es el enfoque de este estudio, va desde las posiciones 1 hasta la 9 que se muestran en el esquemático. En el esquemático se muestra que la recolección se presenta en la posición denominada 1 en la imagen, para después proseguir su recorrido en sentido antihorario por los demás pasos.

La recolección de las piezas del llamado *wafer*, que contiene cientos o miles de estas piezas, consiste en una secuencia de movimientos de componentes mecánicos, los cuales se muestra en el esquemático de la Figura 4.3. Este proceso consiste en un accionador, denominado en el esquemático como componente 114, que mediante un pistón levanta una pieza que sostiene la aguja que se encarga de la recolección (pieza 115 en el esquemático), esta pieza funciona como un Jig de la aguja (también tienen la posibilidad de poner más de una aguja en caso de que sea necesario). Además, funciona para levantar la cinta donde están posicionadas las piezas que se necesita recolectar (pieza 109 en el esquemático), para que posteriormente la aguja extraiga por completo la pieza del *wafer*. Este proceso se controla con un sistema de control mediado por un sistema lógico controlable o PLC a muy alta velocidad.

Después de extraer la pieza, la boquilla encargada de completar la recolección (pieza 107 del esquemático) y mover la pieza a las demás posiciones (Figura 4.2(a)) baja hasta una



(a) Figura 2.a. Esquemático de las estaciones que recorre el dispositivo en la DTR.

(b) Figura 2.b. Esquemático general de la DTR.

Figura 4.2: Esquemáticos de la DTR. Obtenido de [7].

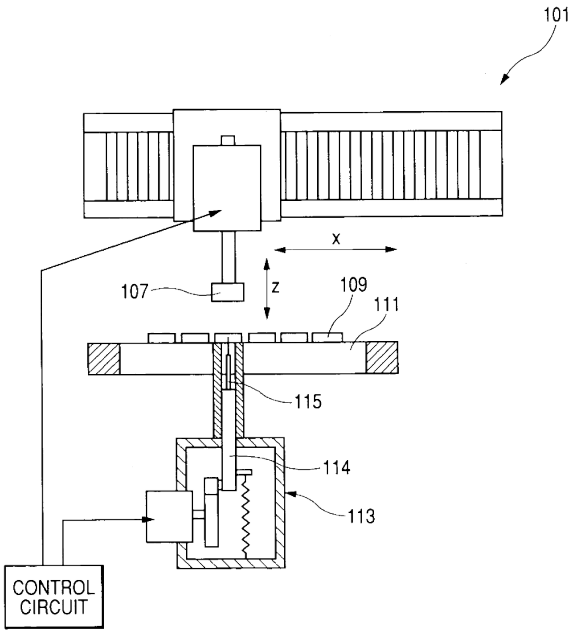


Figura 4.3: Esquemático general del sistema actual de recolección de chips en el proceso de DTR. Obtenido de [28].

distancia 'z' para recolectar la pieza y mediante un vacío crear la sujeción de la pieza. Esta distancia 'z' (mostrada en el esquemático) es un parámetro ajustable que puede variar según el tipo de dispositivo que se esté manejando.

Una vez recolectada la pieza, se traslada a la posición 9 donde se encuentra la inspección visual de la parte superior del dispositivo o espalda donde se descarta cualquier fallo mecánico que pueda afectar el funcionamiento del dispositivo (los mismos fallos que se describen en las secciones posteriores). Esta parte de la estación 1 consiste en un sistema de visión comprendido por una **óptica telecéntrica** y un **domo de iluminación** que puede variar las secciones que se iluminan para tener diferentes tipos de iluminación y diferentes ángulos de incidencia de la luz. Cuando la pieza llega a este sistema, un sistema de control acciona el sistema de visión que toma varias imágenes con diferentes tipos de iluminación para detectar cualquier defecto en la pieza mediante un *software* de visión por computadora. Con este *software* únicamente se conoce si la pieza presenta un fallo o no (binario), por lo que no genera información del tipo de fallo para su análisis posterior.

Es muy importante mencionar que este sistema y el *software* de visión están patentados por la empresa proveedora de la máquina de DTR (Cohu-Ismeca). Por lo tanto, el acceso a este *software* es limitado, esta es una de las razones por las que se necesita de un sistema externo para clasificar los tipos de fallos que se encontraron en esta estación (objetivo del proyecto), ya que acceder a modificar el *software* internamente no solo aumentaría el tiempo de inspección, sino que afectaría la producción por minuto de la máquina. Además, requeriría una solicitud a la empresa proveedora de una actualización y esto implica una importante suma monetaria.

Por último, se resume a continuación, *grosso modo*, los procesos previos que se llevan a cabo con los dispositivos antes de entrar, ya que los fallos detectados por la máquina pueden provenir de estos procesos; este conjunto de procesos se representa en el diagrama de flujo de la Figura 4.4. El primer proceso que se presenta es entrada del *wafer* al cuarto limpio y su inspección inicial.

En esta inspección inicial se detectan y remueven protuberancias en el *wafer* para evitar futuros fallos producidos por estas protuberancias. Después de esta inspección, se pasa a la operación de lijado del *wafer* mediante una maquinaria especializada, la cual necesita una limpieza e inspección posterior, con esto se pasa a la operación llamada de *taping* que consiste en poner un adhesivo en la parte posterior del *wafer* que se denomina cinta o tape, el cual se cura mediante luz UV para la adhesión correcta de este. Una vez terminado el *taping* se pasa a

la operación de singularización del *wafer* que consiste en que, mediante una máquina de corte mecánico o corte láser, según el material que se trabaje, se corten con precisión los dispositivos y con esto singularizar cada uno.

Una vez singularizado se pasa al último proceso antes de la operación de DTR y su máquina, la cual se detalló. Este proceso corresponde a la expansión del *wafer*, para separar un dispositivo de otro dentro del *wafer* y facilitar su recolectado. Este proceso de expansión consiste en extender la cinta o tape uniformemente para posteriormente sujetarlo mediante un aro metálico alrededor del *wafer* para que las piezas estén distanciadas entre sí y facilitar su recolección y, para finalizar, se lleva a cabo una inspección final del *wafer* en busca de cualquier defecto antes de pasarlo a la operación de DTR.

Después de explicar el proceso, a continuación se detallan los fallos mecánicos que puede detectar la máquina de DTR en la estación 1. Esto es el resultado del análisis mecánico que se llevó a cabo en el proceso mediante una inspección física de la máquina, análisis de los planos de esta y reuniones con los ingenieros a cargo de este proceso.

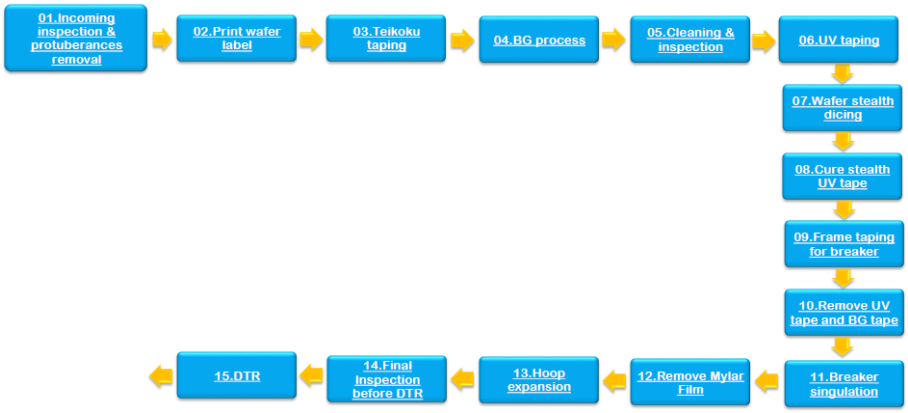
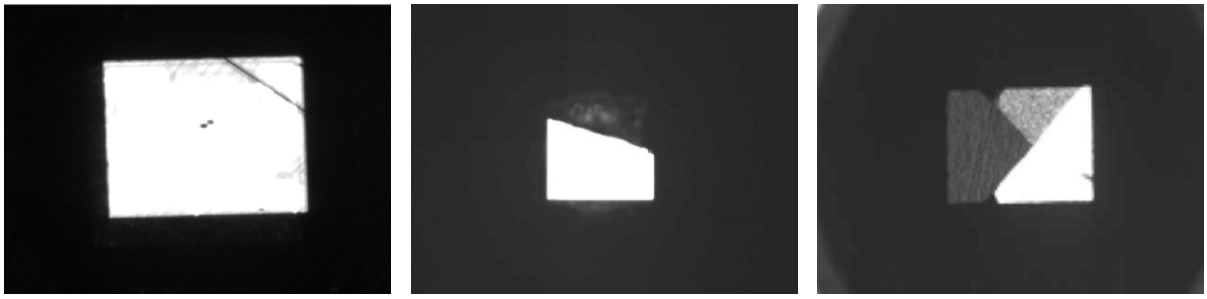


Figura 4.4: Flujo del proceso y tratamiento del *wafer* antes de procesarse en DTR. Obtenido de [6].



(a) Figura 5.a. Quebradura parcial. (b) Figura 5.b. Quebradura total. (c) Figura 5.c. Quebradura en varias direcciones.

Figura 4.5: Ejemplo de *Broken Die*. Elaboración propia.

4.2. Quebradura en dispositivo

Descripción y visualización

En el proceso el defecto más crítico son los dispositivos quebrados. Estos se denominan la fractura completa del dispositivo dividiéndolo en dos o más partes y es un fallo catastrófico que invalida completamente el funcionamiento del dispositivo.

Estos se observan mediante una línea oscura que se contrasta con el tono del dispositivo o un cambio muy definido en la intensidad de luz del dispositivo, en general, van de un lado al otro del cuadrado que este forma, como se muestra en la Figura 4.5(a). Además, se pueden observar como múltiples líneas de fractura en varias direcciones (Figura 4.5(c)) o como se ve en la Figura 4.5(b) donde se observa un faltante en la pieza, lo cual se considera como una fractura total.

Procedencia

Las fracturas de dispositivos provienen de múltiples procesos, el más común de ellos es directamente en la máquina de DTR durante el proceso de recolección, como se muestra en la Figura 4.3,donde se observa que la pieza se recoge con una boquilla mediante un vacío y una aguja que levanta el dispositivo. Este proceso puede llevar a que si está mal ajustada o el dispositivo es muy frágil la aguja perfore el dispositivo, lo que genera una quebradura.

Comúnmente, estos fallos se ven similar a la imagen de la Figura 4.5(c) donde se ve que se presentan múltiples fracturas, pero todas convergen aproximadamente en el centro.

Por otro lado, las fracturas parciales o totales pueden provenir de otros dos procesos específicos. El primero y más común de estos es de la inspección inicial del aro donde están los *wafers* al empezar a procesarlo, esto se muestra en la Figura 4.4 como *Incoming inspection y protuberances removal* o *Inspección inicial y remoción de protuberancias*, en donde se verifica que no haya protuberancias o imperfecciones en el *wafer*. Si en esta inspección no se detecta alguna imperfección al pasar los demás procesos se generarán quebraduras, desde pequeñas fisuras hasta fracturas extensas en el *wafer*, esto se daría en procesos donde se genera presión como el lavado o ligado.

El otro procedimiento que puede generar este tipo de fracturas es el proceso de lijado o *BG Process* que por un mal ajuste de la máquina puede generar fracturas o fisuras que posteriormente dañen las piezas o también puede existir una contaminación que genere rayones que pueden fracturar la pieza.

Consecuencias

Las quebraduras, ya sean parciales o totales del dispositivo, implican un importante defecto de calidad, ya que al ser dispositivos electrónicos pasivos muy pequeños (en la escala de los micrómetros), estos defectos llegan a afectar la respuesta en frecuencia de estos (su aplicación directa), por lo que se vuelve un defecto inaceptable que no debe llegar al cliente final. Este defecto implica que (en caso de gran incidencia) se deba asumir que el rollo presenta más fallos de este tipo, que se haya pasado por alto y actuar en consecuencia, ejecutando un procedimiento de inspección y reparación del rollo. Además, se debe considerar que estas quebraduras, según su tamaño, pueden obstruir la boquilla de la máquina, lo que ocasiona que la recolección de las piezas subsecuentes se vea afectada.

Acciones que tomar

Para abordar este fallo de calidad, según lo establecido en el procedimiento [31], se debe bloquear (no seguir utilizándolo en operaciones siguientes) el rollo que contiene estos defectos en caso de que sean mayor a 3 unidades, para posteriormente examinarlo detenidamente y extraer estos defectos. Además, se debe detener la máquina y seguir los procedimientos de intervención que se establecen [28]. Las acciones por tomar en la máquina que establece este procedimiento son quitar la guardia respectiva para poder acceder al sistema de recolección, limpiarlo detenidamente, y, mediante un microscopio determinar si las agujas y boquillas necesitan un reemplazo por desgaste.

Luego se debe verificar los parámetros de la máquina, siendo estos la presión del sistema neumático que debe estar entre 1.5 y 3.1Bar y el flujo de vacío del sistema neumático de la máquina en la recolección que debe estar en 7.5 +/- 0.05 l/s, la temperatura de operación (entre 150^o-230^oC) y el tiempo de retraso que tiene la máquina durante la recolección (tiempo entre que baja la aguja y sube la boquilla que debe estar entre 15 ms y 90 ms).

Una vez verificados todo esto se debe seguir procesando material, con seguimiento de cerca de la máquina y en caso de seguir detectando estos fallos se debe verificar el estado del *wafer* y notificar a los ingenieros de proceso en planta.

4.3. Muesca en dispositivo

Descripción y visualización

El segundo defecto más crítico en el proceso es el de muescas en el dispositivo. Estos fallos se definen como pequeñas concavidades o huecos que hay a los lados del dispositivo. Estas muescas pueden generarse como consecuencia del propio proceso de recolección o de alguno de los procesos anteriores, pero no todos estos defectos son catastróficos o requieren que se deseche el dispositivo, solo las que contienen ciertas características que se establecen por múltiples

experimentos por parte del Departamento de Calidad de la empresa. En la Figura 4.6, se puede observar la parametrización de este defecto [11],donde una muesca o *chip out* es catastrófico cuando el valor de la longitud (y) del mismo supera los 80 µm y la profundidad (z) supera el 75 % del grosor del dispositivo.

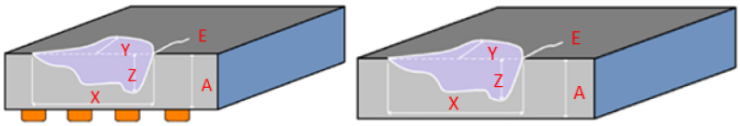
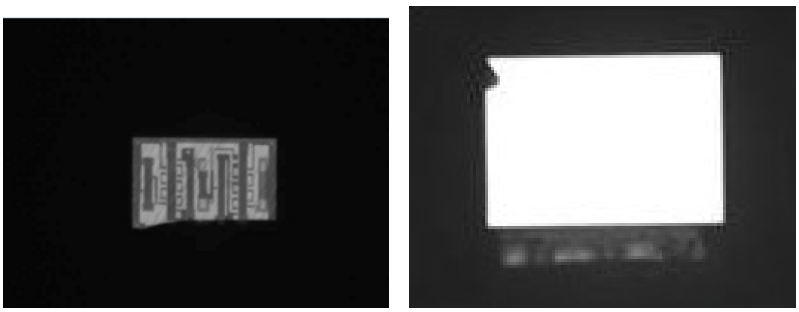


Figura 4.6: Parametrización del fallo por muesca o *Chip Out*. Obtenido [11].

Este defecto se puede identificar de dos maneras, cuando se observa como un segmento faltante en el dispositivo, cuyo tamaño puede variar, pero siempre se mantiene del mismo lado y nunca pasa de un lado a otro, como se observa en la Figura 4.7(b), cuando se traspasa a otro lado el fallo pasa a ser fractura. La otra forma de reconocerlo es cuando se ve un faltante que se extiende en todo un lado del dispositivo, como se muestra en la Figura 4.7(a).

Procedencia

Las muescas descritas provienen principalmente del proceso de *hoop expantion* o expansión del *wafer*, que se muestra en la Figura 4.4. Esto se debe a que cuando se hace el proceso de singularización, también mostrado en la Figura 4.4, puede haber algún mal ajuste o contaminación que causa un mal corte. Esto causa que en el momento del *hoop expantion* o expansión del *wafer* las piezas se separen mal y que una parte de la pieza o Die se quede en la que está al lado, lo que provoca daños a ambos dispositivos.



(a) Figura 7.a. *Chip Out* amplio. (b) Figura 7.b. Chip Out puntual.

Figura 4.7: Ejemplos de *Chip Out*. Elaboración propia.

La otra forma en que se puede dar este defecto es durante la recolección en la estación 1 del proceso de DTR, como se muestra en la Figura 4.3 donde se observa el mecanismo de recolección. Esto se presenta en el momento en que el dispositivo choca con el *nozzel* o boquilla (mostrada como pieza 107 en el esquema) y puede fracturar el dispositivo en los lados.

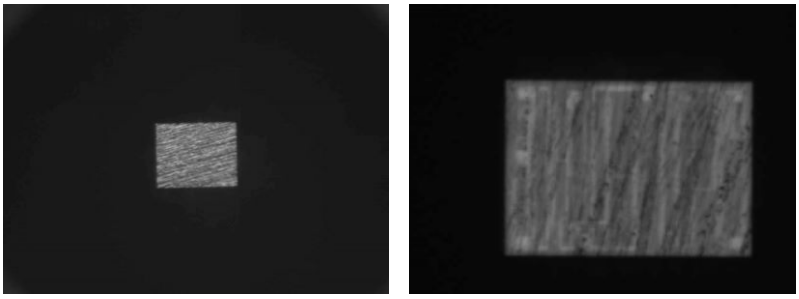
Consecuencias

Las muescas, al igual que las quebraduras, implican una importante falla de calidad, ya que al ser dispositivos tan pequeños, estas pequeñas muestras pueden afectar el desempeño o funcionalidad de estos. Como se mencionó, al ser dispositivos tan pequeños, cualquier inconsistencia afecta el comportamiento de este y al ser su aplicación directa el filtrado por medio de ondas acústicas, cualquiera de estas inconsistencias mecánicas afectará negativamente la respuesta del dispositivo. No obstante, a diferencia de las quebraduras, este tipo de fallo no afecta a la máquina directamente, pero es importante mencionar que por la naturaleza de este es muy probable que si se presenta este tipo de fallo los dispositivos circundantes en el *wafer* estén afectados de esta manera.

Acciones que tomar

Las acciones requeridas ante este tipo de fallo son muy similares a cuando se presentan quebraduras [31]. Para abordar este fallo, según lo establecido en el procedimiento [31], se debe intervenir la máquina en donde se proceso el rollo que contiene estos defectos en caso de que sean mayor a >0.1 % de las imágenes. Una vez detenida la máquina, se debe seguir los procedimientos de intervención que se establecen [28]. Las acciones por tomar en la máquina que establece este procedimiento son quitar la guardia respectiva para poder acceder al sistema de recolección, limpiarlo detenidamente, y, mediante un microscopio determinar si las agujas y boquillas necesitan un reemplazo por desgaste.

Luego se debe verificar los parámetros de la máquina, siendo estos la presión del sistema neumático que debe estar entre 1.5 y 3.1Bar y el flujo de vacío del sistema neumático de la máquina en la recolección que debe estar en 7.5 +/- 0.05 l/s, la temperatura de operación (entre



(a) Figura 8.a. Ejemplo 1 de líneas de pulido.

(b) Figura 8.b. Ejemplo 2 de líneas de pulido.

Figura 4.8: Ejemplos de líneas de pulido. Elaboración propia.

150^o-230^{o}C) y el tiempo de retraso que tiene la máquina durante la recolección (tiempo entre que baja la aguja y sube la boquilla que debe estar entre 15 ms y 90 ms).

Una vez verificados todo esto se debe seguir procesando material, con seguimiento de cerca de la máquina y en caso de seguir detectando estos fallos se debe verificar el estado del *wafer* y notificar a los ingenieros de proceso en planta para que inspecciones si el modo de fallo puede provenir de operaciones anteriores.

4.4. Líneas de pulido

Descripción y visualización

Este de defecto consiste en líneas provenientes del pulido de la pieza, estas son paralelas y poco profundas y se extienden por toda la superficie del dispositivo. El factor clave para reconocer este fallo es que son líneas paralelas a diferencia de las quebraduras que van en direcciones aleatorias o son una sola. Algunos ejemplos de este fenómeno se muestran en la Figura 4.8 Es importante mencionar que estas líneas siempre están en la pieza, sin importar si hay alguna falla en el proceso, pero estas suelen ser muy leves, solo en casos donde hay algún fallo se vuelven más profundas y, por consiguiente, detectables con el sistema de visión de la máquina.

Procedencia

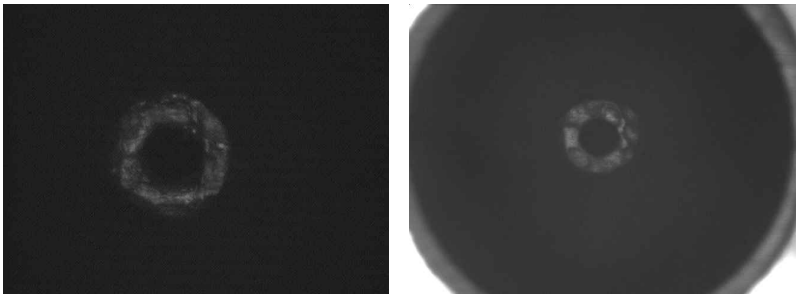
Como lo menciona la denominación del fallo, este proviene del proceso de lijado y se presenta cuando existe alguna contaminación o deformación en el equipo de lijado, lo que lleva a que cuando pase por el dispositivo deje marcas tan finas que la prueba de rugosidad mostrada en la Figura 4.4 no las detecta, pero el sistema de visión lo hace y rechaza la pieza.

Consecuencias

Las líneas de lijado no son fallos que afecten el comportamiento del dispositivo, pero son un fallo de calidad que puede generar otra falla en el futuro que puede afectar negativamente el comportamiento de la pieza. Esto se debe a que, al ser líneas de mayor profundidad que se generan en el dispositivo debido a una contaminación o mal ajuste, se convierte en un punto crítico de quiebre de esta, lo que vuelve más fácil su fractura. Además, es importante mencionar que estas líneas pueden verse superficiales en la imagen, pero pueden ser profundas y fracturarse a lo largo de los procesos posteriores a DTR. Por esto, en este tipo fallo, a pesar de que no implica ninguna consecuencia directa, se deben contemplar las mismas consecuencias que para una pieza con quebradura o muesca.

Acciones que tomar

Las acciones por tomar en este caso son circunstanciales. El procedimiento por seguir es exactamente el mismo que si hay presencia de quebraduras, pero este procedimiento únicamente se debe seguir en caso de que haya peligro de que estas líneas de lijado sean tan profundas como para generar quebraduras en los procesos subsecuentes. Para reconocer si estas líneas pueden ser un defecto de calidad (generar quebraduras) existen dos formas, la primera y más sencilla es que en caso de que en el mismo rollo aparezca cualquier indicación de una pieza quebrada, además de piezas con líneas de lijado, se asume que estas últimas pueden ser o generar quebraduras y, la segunda forma, es que haya una línea que sea notoriamente más pronunciada o gruesa que las demás, pero conservando el paralelismo con las demás. Una vez identificada la posibilidad de



(a) Figura 9.a. Ejemplo 1 de boquilla vacía.

(b) Figura 9.b. Ejemplo 2 de boquilla vacía.

Figura 4.9: Ejemplos de boquilla vacía. Elaboración propia.

que estas líneas de lijado sean fracturas se realiza el mismo procedimiento que para fallos por quebraduras o muescas [28].

4.5. Boquilla vacía

Descripción y visualización

Este es uno de los defectos más comunes, el cual se define únicamente por la ausencia del dispositivo después de la recolección y se observa cuando se ve en su totalidad la boquilla o *nozzel* de recolección, como se muestra en la Figura 4.9.

Procedencia

Este fallo puede darse por varias razones, la más común de ellas es que la aguja mostrada en la Figura 4.3, no logra despegar el dispositivo de la cinta y queda pegado, por lo que la boquilla no logra extraerla. La otra razón es que la boquilla esté desgatada, porque al ser un proceso repetitivo de miles de revoluciones por hora estas se desgastan fácilmente. Por lo tanto, si estas no se reemplazan oportunamente llegan a generar fallos en la recolección, ya que no recolecta la pieza de manera estable o dificulta su adhesión a la boquilla con la ayuda del vacío y como resultado da una boquilla vacía.

La tercera razón es por un mal ajuste en la recolección, ya sea porque la boquilla está

descentrada y no recolecta correctamente el dispositivo o recolecta donde no se encuentra ningún dispositivo, por lo que la boquilla queda vacía.

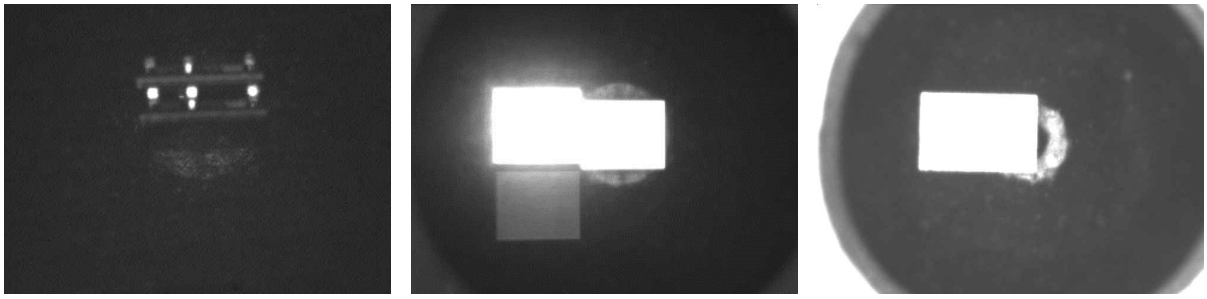
Consecuencias

Este defecto no genera ningún tipo de falla de calidad ni afecta el desempeño de la pieza, pero si aparece afecta el desempeño de la máquina o del proceso en lo que a producción se refiere. Al desperdiciar piezas con la mala recolección se ve un decrecimiento en la métrica de unidades producidas por hora o UPH de la máquina, además de que las piezas mal recolectadas representan un desperdicio en dólares para la empresa que depende del precio por unidad de cada dispositivo. Con esto se puede afirmar que, a pesar de no generar fallos de calidad, este defecto se refleja en un gasto económico importante para el proceso.

Acciones que tomar

Las acciones requeridas ante este tipo de fallo son muy similares a cuando se presentan quebraduras [31]. Para abordar este fallo, según lo establecido en el procedimiento [31], se debe intervenir la máquina en donde se proceso el rollo que contiene estos defectos en caso de que sean mayor a $>0.15\,\%$ de las imágenes. Una vez detenida la máquina, se debe seguir los procedimientos de intervención que se establecen [28]. Las acciones por tomar en la máquina que establece este procedimiento son quitar la guardia respectiva para poder acceder al sistema de recolección, limpiarlo detenidamente, y, mediante un microscopio determinar si las agujas y boquillas necesitan un reemplazo por desgaste. Adicionalmente se debe ajustar las alturas de las boquillas mediante una galga para medir dichas alturas, siguiendo el procedimiento de [28].

Luego se debe verificar los parámetros de la máquina, siendo estos la presión del sistema neumático que debe estar entre 1.5 y 3.1Bar y el flujo de vacío del sistema neumático de la máquina en la recolección que debe estar en 7.5 +/- 0.05 l/s, la temperatura de operación (entre 150^o-230^oC) y el tiempo de retraso que tiene la máquina durante la recolección (tiempo entre que baja la aguja y sube la boquilla que debe estar entre 15 ms y 90 ms).



(a) Figura 10.a. Recolección de dispositivo volteado

(b) Figura 10.b. Recolección de múltiples dispositivos.

(c) Figura 10.c. Recolección de dispositivo descentrado.

Figura 4.10: Ejemplos de dispositivos mal recolectados. Elaboración propia.

Una vez verificados todo esto se debe seguir procesando material, con seguimiento de cerca de la máquina y en caso de seguir detectando estos fallos se debe verificar el estado del *wafer* y notificar a los ingenieros de proceso en planta para que inspecciones si el modo de fallo puede provenir de operaciones anteriores.

Para solventar esta falla, se necesita que se detenga la máquina al seguir el procedimiento establecido en [28]. Esto para verificar los parámetros concernientes de las máquinas y el funcionamiento correcto de esta.

4.6. Problemas de recolección

Descripción y visualización

Este defecto se define como un problema en la recolección del dispositivo, en donde se recolecta el dispositivo, pero con algún problema de orientación o una múltiple recolección (se recolecta más de una pieza). Este fallo se puede observar cuando se logra diferenciar un dispositivo volteado o descentrado como en la Figura 4.10(a) y 4.10(c) o cuando se observa una recolección de más de un dispositivo como en la Figura 4.10(b).

Procedencia

Este defecto proviene enteramente de un mal funcionamiento de la máquina de DTR. Este mal funcionamiento corresponde un mal ajuste o un sobreajuste en los parámetros de la aguja, mostrada en la Figura 4.3,como su altura o compresión, los cuales debe estar puede ocasionar que el dispositivo se voltee. Estos parámetros son característicos de la máquina y entre los diferentes modelos (tamaños de piezas) que se procesa, están regulados y deben estar en valores entre -100um y 100um según la orientación que se muestra en la Figura 4.11.

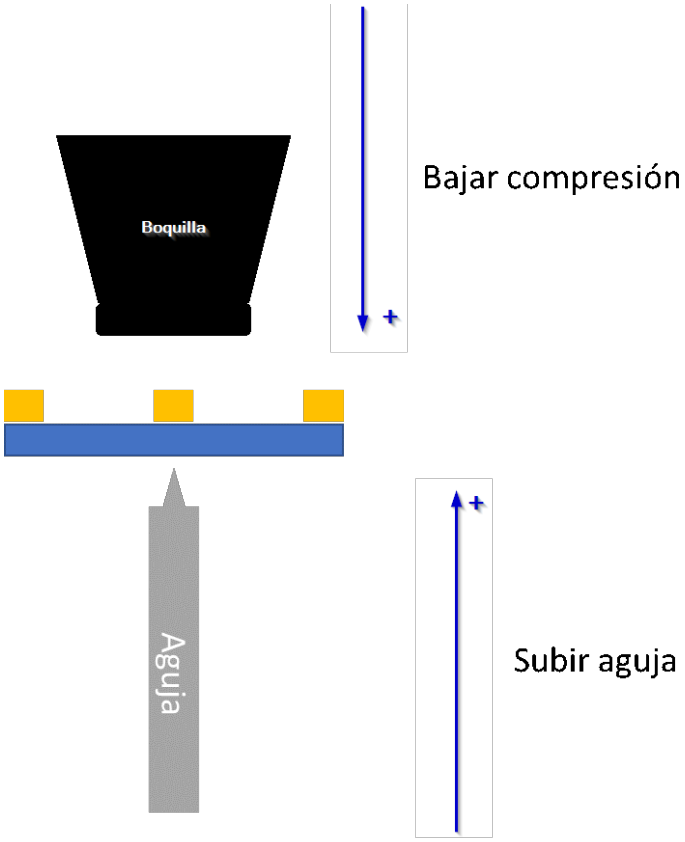


Figura 4.11: Esquemático de la compresión de la boquilla y la altura de la aguja en el proceso. Elaboración propia.

Estos valores pueden manipularse regularmente, suele pasar que se desajusta o sobreajusta el parámetro por parte del operario, lo que genera que se descentre, tuerza o gire la pieza como en la Figura 4.10(a) y la Figura 4.10(c) o incluso recolectar los dispositivos circundantes como en la Figura 4.10(b).

Consecuencias

Al ser un fallo directamente de la máquina, este no presenta un riesgo de calidad o riesgo de dañar los dispositivos, pero al igual que el fallo por boquilla vacía implica un gran costo a la producción. Este costo se puede definir como el decrecimiento en la métrica de unidades producidas por hora o UPH de la máquina, sumándole a esto que las piezas mal recolectadas representan un desperdicio en dólares para la empresa que depende del precio por unidad de cada dispositivo. Además, este fallo tiene una peculiaridad, ya que al no tratarse las piezas pueden llegar a fracturarse y quedar incrustadas en la boquilla, lo que imposibilita la correcta recolección de piezas subsecuentes.

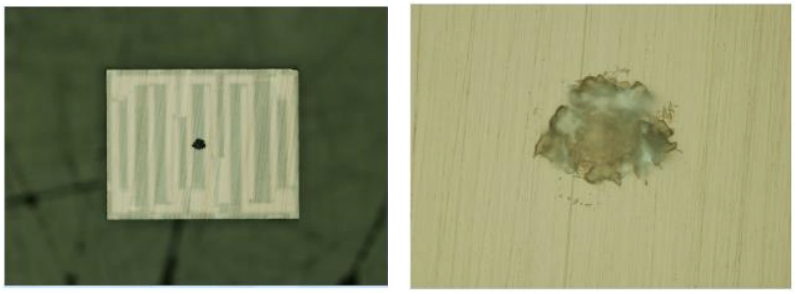
Acciones que tomar

Las acciones requeridas ante este tipo de fallo son muy similares a cuando se presentan quebraduras [31]. Para abordar este fallo, según lo establecido en el procedimiento [31], se debe intervenir la máquina en donde se proceso el rollo que contiene estos defectos en caso de que sean mayor a >0.10 % de las imágenes. Una vez detenida la máquina, se debe seguir los procedimientos de intervención que se establecen [28].

Las acciones por tomar en la máquina que establece este procedimiento son quitar la guardia respectiva para poder acceder al sistema de recolección, limpiarlo detenidamente, y reemplazar las agujas y boquillas indiferentemente si ya estan desgastadas o no. Adicionalmente se debe ajustar las alturas de las boquillas y de la aguja mediante una galga para medir dichas alturas, siguiendo el procedimiento de [28].

Luego se debe verificar los parámetros de la máquina, siendo estos la presión del sistema neumático que debe estar entre 1.5 y 3.1Bar y el flujo de vacío del sistema neumático de la máquina en la recolección que debe estar en 7.5 +/- 0.05 l/s, la temperatura de operación (entre 150^o-230^oC) y el tiempo de retraso que tiene la máquina durante la recolección (tiempo entre que baja la aguja y sube la boquilla que debe estar entre 15 ms y 90 ms).

Una vez verificados todo esto se debe seguir procesando material, con seguimiento de



(a) Figura 12.a Marca de aguja en la pieza

(b) Figura 12.b. Acercamiento a la marca de aguja.

Figura 4.12: Ejemplo de marca de aguja en el dispositivo. Elaboración propia.

cerca de la máquina y en caso de seguir detectando estos fallos se debe verificar el estado del *wafer* y notificar a los ingenieros de proceso en planta para que inspecciones si el modo de fallo puede provenir de operaciones anteriores.

Para solventar esta falla, se necesita que se detenga la máquina al seguir el procedimiento establecido en [28]. Esto para verificar los parámetros concernientes de las máquinas y el funcionamiento correcto de esta.

4.7. Fallo por aguja

Descripción y visualización

Este tipo de fallo es enteramente un desperfecto mecánico producido en la máquina y su denominación es la marca que deja al ejercer más presión de la que debe al recolectar la pieza. Esta marca puede ser de diferentes profundidades, pero en todo caso representa un defecto de calidad que llega a afectar el desempeño del dispositivo al ser componentes pasivos tan pequeños.

Este defecto es muy fácil de reconocer, debido que se identifica con un cambio bastante marcado en la intensidad de los píxeles en una ubicación muy cercana al centro de la pieza, como se muestra en la Figura 4.12(a), y este no tiene un diámetro definido o estrictamente una forma circular como se muestra en el acercamiento de la Figura 4.12(b).

Procedencia

Este fallo, como se mencionó, procede de un desperfecto mecánico producido en la máquina y corresponde a un mal ajuste de los parámetros de la aguja y de la boquilla o al mal estado (desgaste) de la propia aguja. El mal ajuste en los parámetros, como se mencionó en la sección 4.6, corresponde a un mal ajuste o un sobreajuste en los parámetros de la aguja, su altura o compresión, en referencia con los mencionados.

Consecuencias

Las marcas de aguja, al igual que las muescas, implican una importante falla de calidad. Como se mencionó, estas pueden afectar el desempeño o funcionalidad de estos, por lo tanto, cualquiera de estas inconsistencias mecánicas afectará negativamente la respuesta en frecuencia del dispositivo.

Acciones que tomar

Las acciones por tomar en la máquina que establece este procedimiento [28] son quitar la guardia respectiva para poder acceder al sistema de recolección, limpiarlo detenidamente, y reemplazar las agujas indiferentemente si ya están desgastadas o no. Adicionalmente se debe ajustar las alturas de las boquillas y de la aguja mediante una galga para medir dichas alturas, siguiendo el procedimiento de [28].

Luego se debe verificar los parámetros de la máquina, siendo estos la presión del sistema neumático que debe estar entre 1.5 y 3.1Bar y el flujo de vacío del sistema neumático de la máquina en la recolección que debe estar en 7.5 +/- 0.05 l/s, la temperatura de operación (entre 150^o-230^oC) y el tiempo de retraso que tiene la máquina durante la recolección (tiempo entre que baja la aguja y sube la boquilla que debe estar entre 15 ms y 90 ms).

Una vez verificados todo esto se debe seguir procesando material, con seguimiento de cerca de la máquina y en caso de seguir detectando estos fallos se debe verificar el estado del

wafer y notificar a los ingenieros de proceso en planta para que inspecciones si el modo de fallo puede provenir de operaciones anteriores.

Para solventar esta falla, se necesita que se detenga la máquina al seguir el procedimiento establecido en [28]. Esto para verificar los parámetros concernientes de las máquinas y el funcionamiento correcto de esta.

4.8. Contaminación

Descripción y visualización

Este de defecto consiste en una partícula o partículas (que pueden ser de diferentes materiales o aspectos) que está sobre el dispositivo. Este defecto es muy similar a una marca de aguja, pero el factor clave para reconocer este fallo es que recurrentemente se observa con bordes erosionados, se encuentra en posiciones aleatorias (a diferencia de la marca por aguja que es aproximadamente en el centro) y puede tener cualquier forma. Un ejemplo de este fenómeno se muestra en la Figura 4.13 donde se muestra una pequeña contaminación semicircular (similar a una marca de aguja) ubicada cerca del borde del dispositivo.

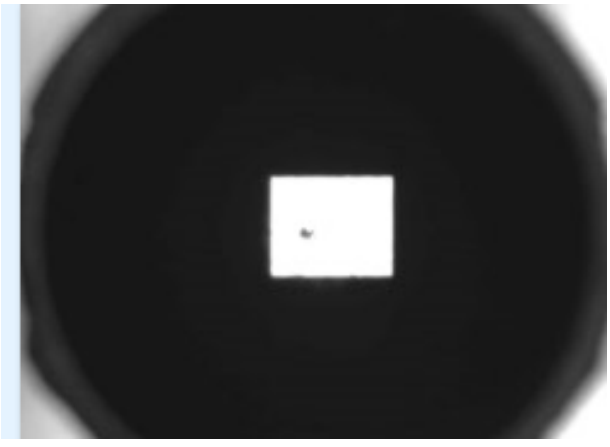


Figura 4.13: Ejemplo de contaminación sobre el dispositivo. Elaboración propia.

Procedencia

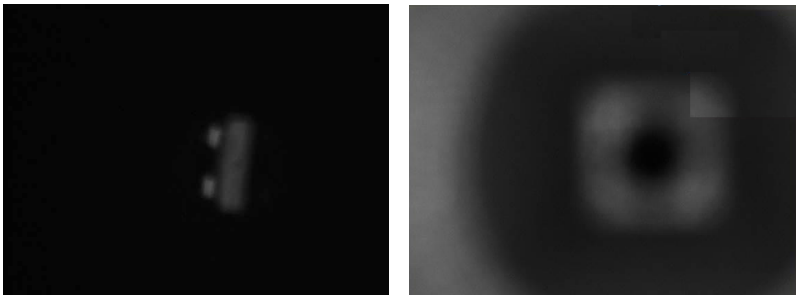
Este fallo, como se mencionó, es una contaminación o partícula situada sobre el dispositivo y puede provenir de varias situaciones. La primera y más común de las situaciones es una mala manipulación del wafer, la cual puede darse en cualquier parte del proceso donde se manipule manualmente. Las otras situaciones son poco comunes, pero pueden darse, como un mal funcionamiento de los filtros de la máquina (observados en la Figura 4.2(b)) o mal funcionamiento de los filtros o controles de los parámetros ambientales del área donde se encuentre la máquina (un cuarto limpio).

Consecuencias

Las contaminaciones, al igual que las marcas por aguja, implican una importante falla de calidad. Como se mencionó, estas pueden afectar el desempeño o funcionalidad de estos, por lo tanto, cualquiera de estas inconsistencias mecánicas afectará negativamente la respuesta en frecuencia del dispositivo.

Acciones que tomar

Al ser un fallo que no proviene de la máquina, no se necesita detener la producción de esa máquina o intervenirla, en su defecto se debe indagar de que proceso de los involucrados (Figura 4.4) proviene la contaminación y tomar las acciones para corregir la mala manipulación o inconsistencia en el proceso. En caso de que el fallo provenga de un mal funcionamiento de los filtros de la máquina o controles de los parámetros ambientales se debe informar al Departamento de Mantenimiento de la empresa para que tomen acciones lo antes posible.



(a) Figura 14.a Ejemplo 1 de problemas de visión.

(b) Figura 14.b. Ejemplo 2 de problemas de visión.

Figura 4.14: Ejemplos de fallos por problemas de visión. Elaboración propia.

4.9. Problemas de visión

Descripción y visualización

Los problemas de visión en la máquina no son un defecto muy frecuente, estos se definen únicamente como la mala o nula visualización de los componentes normales en la imagen (dispositivo y boquilla). Esto se observa como imágenes mal iluminadas o contrastadas o simplemente no se aprecian los componentes de la imagen por difuminación o ruido. Algunos ejemplos de estos casos se muestran en las imágenes de la Figura 4.14.

Procedencia

Como es de esperarse, este fallo proviene enteramente de un mal funcionamiento o ajuste de la cámara. La cámara tiene ajustes predefinidos según los protocolos de la empresa [28], donde se establece el lente por utilizar, la distancia y la longitud focales. De no ajustar esto de manera adecuad se presenta una mala visualización del dispositivo, como se muestra en la Figura 4.14. Además, como se mencionó, este fallo puede provenir de un mal funcionamiento del sensor (cámara).

Consecuencias

Este defecto no genera ningún tipo de falla de calidad ni afecta el desempeño de la pieza, pero puede afectar el desempeño de la máquina o del proceso en lo que a producción se refiere. Al desperdiciar piezas por una mala interpretación de la imagen por parte del software de visión, se ve un decrecimiento en la métrica de unidades producidas por hora o UPH de la máquina, además de que las piezas desechadas representan un desperdicio en dólares para la empresa que depende del precio por unidad de cada dispositivo. Con esto se puede afirmar que, a pesar de no generar fallos de calidad, este defecto se refleja en un gasto importante para el proceso.

Acciones que tomar

Para abordar este fallo, según lo establecido en el procedimiento [31], se debe intervenir la máquina en donde se proceso el rollo que contiene estos defectos en caso de que sean mayor a $>0.10\,\%$ de las imágenes. Una vez detenida la máquina, se debe seguir los procedimientos de intervención que se establecen [28].

Las acciones por tomar en la máquina que establece este procedimiento son quitar la guardia respectiva para poder acceder al sistema de recolección, limpiarlo detenidamente, y reajustuar los parámetros del sistema de visión, que son las máscaras que detecta cada uno de las especificaciónes [28].

Una vez verificados todo esto se debe seguir procesando material, con seguimiento de cerca de la máquina y en caso de seguir detectando estos fallos se debe verificar el estado del *wafer* y notificar a los ingenieros de proceso en planta para que inspecciones si el modo de fallo puede provenir de operaciones anteriores.

Para solventar esta falla, se necesita que se detenga la máquina al seguir el procedimiento establecido en [28]. Esto para verificar los parámetros concernientes de las máquinas y el funcionamiento correcto de esta.

Capítulo 5: Desarrollo

En este capítulo se detalla el diseño del sistema, lo que incluye la justificación de las decisiones tomadas durante su elaboración. Se comienza por la descripción de las necesidades que se identificaron del cliente, después se definen las métricas para evaluar el correcto desempeño de los conceptos que se plantearon y se definen valores ideales y marginales para cada métrica. Posteriormente, se hace un análisis del problema para plantear un esquema de solución y los conceptos para llevar a cabo la solución propuesta.

5.1. Establecimiento de las necesidades y especificaciones del sistema

Primero, se debe detallar el desarrollo de las primeras dos etapas de la metodología que se planteó, las cuales corresponden a las etapas de identificación y determinación de necesidades donde se identificarán las necesidades del cliente y las especificaciones necesarias del sistema por desarrollar.

Inicialmente, para establecer las necesidades que debe satisfacer el sistema por diseñar se hizo una reunión con los operarios, técnicos e ingenieros del Departamento de Disposiciones que actualmente realizan la inspección de imágenes de rechazos manualmente (DTR Map) y junto con la observación del proceso se determinan las características necesarias del sistema que se desarrolla. Las necesidades que se identificaron se presentan en la Tabla 5.1, estas se revisaron y aprobaron con el asesor de la empresa y el Departamento de Disposiciones.

Estas necesidades se categorizaron en una escala del 1 al 3, 1 eran las necesidades de importancia crítica, 2 las altamente deseables y 3 las deseables, pero no necesarias. Esta categorización se determinó mediante una jerarquización de estas necesidades y una reunión con el asesor asignado por la empresa y los miembros del Departamento de Disposiciones.

Estas necesidades se agruparon según el departamento de la empresa que se ve afectado por la necesidad, ya que este proyecto afecta directamente a varios departamentos. Las necesidades de la 1 a la 6 son de vital importancia para el Departamento de Calidad porque se concentran en la detección correcta de las fallas críticas y en evitar los errores humanos durante la inspección. Por otro lado, las necesidades de la 7 a la 12 se concentran en asegurar un aumento de la eficiencia del proceso de inspección, su estandarización (eliminar diferencias entre criterios de cada persona), su eficacia y su productividad. Por último, las necesidades de la 13 a la 16 se aseguran de que la información de la clasificación y su consecuente interpretación (si existen fallos por parte de la máquina) llegue a las personas colaboradoras pertinentes.

Una vez establecidas estas necesidades, se creó una lista de métricas que corresponda de manera directa y completa a cada una de las necesidades, esto mediante el análisis minucioso de cada necesidad. La relación entre estas métricas con cada necesidad se representa en la matriz de la Figura 5.1.

Junto con esta lista de necesidades es necesario establecer sus unidades y valores, tanto ideales como marginales. Para esto, se analizó cada métrica para establecer un valor que cumpla con los requerimientos del proyecto. Para el caso de las exactitudes de detección y la velocidad de clasificación por imagen se llevó a cabo un experimento donde se evaluó la exactitud del proceso actual. Este experimento consistió en seleccionar dos lotes procesados que tuvieran fallos variados y pedirles la clasificación a dos operarios que actualmente realizan el proceso de disposición, este experimento se hizo al final del turno de ambos operarios, ya que es el punto donde hay más cansancio acumulado y donde hay más fuentes de errores, esto se llevó a cabo 3 veces durante una semana (lunes, miércoles y viernes). Los dos lotes seleccionados contienen un total de 2350 imágenes (1835 y 515 respectivamente) y provienen de distintas máquinas, los valores promedio de los resultados de este experimento se observan en la Tabla 5.2.

Tabla 5.1: Lista de necesidades y su importancia del diseño a desarrollar. Elaboración propia.

Departamento	Núm.	Necesidad	Imp.
Calidad	1	El sistema detecta fallas críticas.	1
	2	El sistema clasifica autónomamente.	1
	3	El sistema no pierde información.	1
	4	El sistema tiene una alta precisión en la clasificación.	1
	5	El sistema puede detectar los diferentes fallos en el material.	2
	6	El sistema no se ve influenciado por ajustes rutinarios en las cámaras.	2
Producción	7	El sistema estandariza el proceso de clasificación de fallos.	1
	8	El sistema puede clasificar el volumen de imágenes sin afectar el tiempo de ciclo de las disposiciones.	2
	9	El sistema puede documentar los rollos clasificados y sus fallos.	2
	10	El sistema optimiza el proceso de inspección visual actual.	2
	11	El sistema optimiza el aseguramiento de calidad del material.	2
	12	El sistema tiene bajos consumos computacionales.	3
Producto y Proceso	13	El sistema detecta sobre posibles fallos causados por máquina en el material.	1
	14	El sistema permite acceder visualmente a la información de los rollos clasificados.	1
	15	El sistema notifica sobre posibles fallos causados por máquina y posibles acciones a tomar.	2
	16	El sistema solo debe uso de softwares de la empresa y de acceso libre.	2

Con base en los resultados del experimento, mostrados en la Tabla 5.2, y con una reunión el asesor de empresa y el grupo de disposiciones de la empresa se determinó que el sistema por desarrollar debía, como mínimo, tener una exactitud igual o superior a la actual en la clasificación de fallos críticos (90 % aproximadamente), lo que marca el valor marginal en 90 % y un valor ideal de 100 %. Para la clasificación de las demás categorías que no son críticas se desea una precisión alta, pero no tan estricta como con los fallos que impliquen un problema de calidad, por lo que se establece el valor ideal en el valor de exactitud del sistema actual (90 % aproximadamente) y se da un margen de marginalidad de 10 %. Por último, para el tiempo de clasificación se desea tener un valor como mínimo igual al actual, pero idealmente se debe

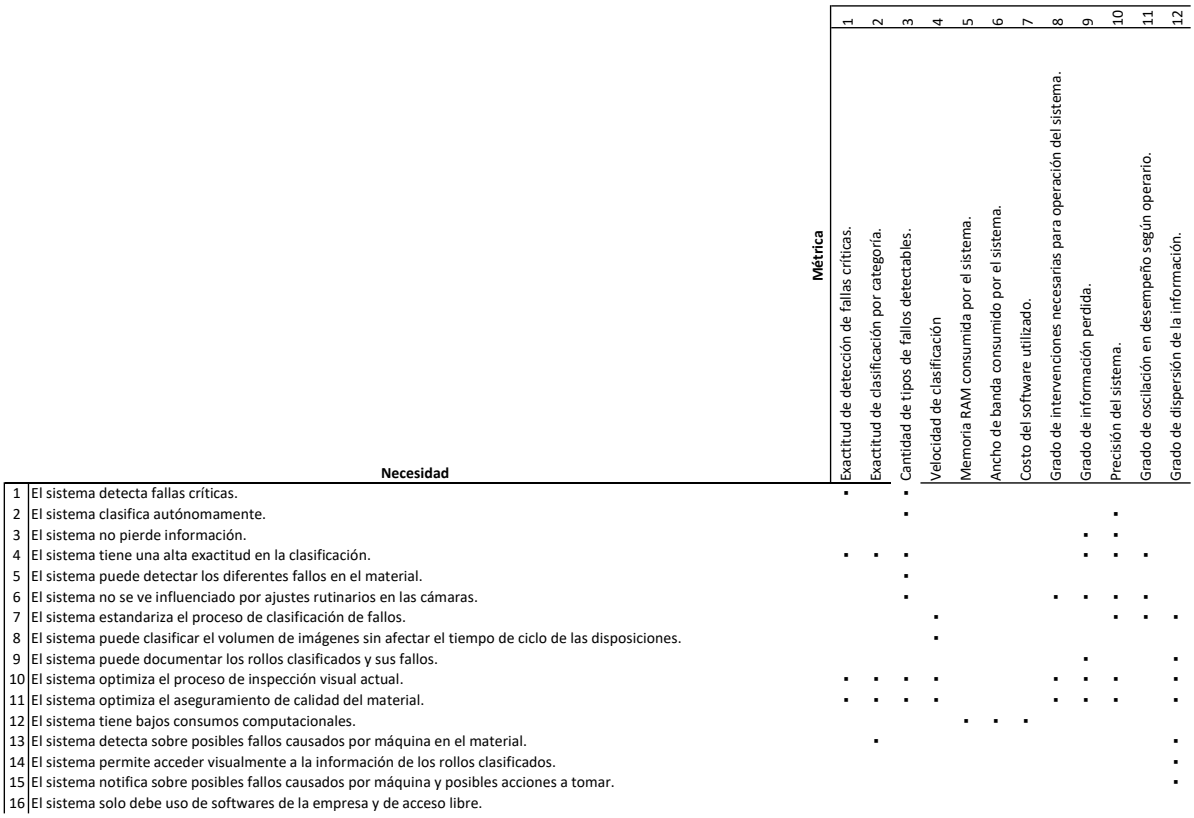


Figura 5.1: Matriz de necesidades-métricas. Elaboración propia.

tener una mejora en al menos un 40 % según el criterio del Departamento de Disposición, esto considerando la creciente tendencia del volumen de producción.

Por otro lado, con respecto a las métricas de la cantidad de tipos de fallos detectables por el sistema, se definió por medio del estándar que tiene la empresa para la clasificación de este tipo de fallos [31], y con el criterio del Departamento de Disposiciones y el de proceso y tiene como mínimo los 5 fallos que actualmente se clasifican por medio del proceso de *DTR Map* y como

Tabla 5.2: Resultados promedio del experimento para observar el sistema de clasificación actual Elaboración propia

Medición	**Operario 1**		**Operario 2**		**Promedio**
	Lote 1	**Lote 2**	**Lote 1**	**Lote 2**	**Total**
Exactitud de la clasificación	96.19 %	86.11 %	87.62 %	91.67 %	90.40 %
Exactitud de fallos críticos	87.19 %	98.05 %	90.24 %	98.47 %	93.49 %
Tiempo por imagen (s/img)	0.57	0.77	0.55	0.73	0.66

máximo 9, este es el número de categorías que contempla todos fallos de la máquina según el estándar. Estos valores iniciales se redefinieron durante el proyecto a raíz de una reestructuración del proceso de inspección y con esto el estándar de la empresa, donde se unieron dos tipos de fallos (recolección doble y mala recolección) que tenían características muy similares, pero con las mismas consecuencias y acciones por tomar, esto fue aprobado por ambos departamentos para mayor facilidad de las personas colaboradoras. Con esto, el valor marginal e ideal terminó en 4 y 8 respectivamente. Es importante mencionar que las clasificaciones mencionadas se explican con detalle en el estudio mecánico realizado a la máquina que se presenta en secciones posteriores.

Para las métricas 5, 6, 7, que consisten en la medición de los recursos computacionales del sistema una vez completado las fases de desarrollo, son valores medidos directamente en el recurso destinado para este fin, el cual especifica la empresa y se establece con base en los recursos del Departamento de IT de la compañía. Para establecerlos se realizó una reunión con y se destinó un espacio en un servidor local con estos recursos, los cuales fueron aprobados tanto por los involucrados en el proyecto como por el asesor de empresa, quienes los consideran suficientes. Es importante mencionar que esto se dispuso de esta manera, ya que durante el estudio inicial del problema surgió la limitante de que no se puede usar el *software* o *hardware* de la propia máquina por normas establecidas con su proveedor, por lo que esta infraestructura externa es estrictamente necesaria para llevar a cabo el sistema.

Con respecto a las métricas de la 8, 9, 11 y 12, por su naturaleza se necesita establecer un sistema de grados para evaluar el comportamiento de esta métrica, estos se detallan en la Tabla 5.3.

Por último, la métrica 10 contempla la precisión del sistema para evaluar si no se ve afectado por diferentes calibraciones, máquinas u operadores en el proceso de extracción y empaque de dispositivos (donde se toman las imágenes a inspeccionar). Al observar los resultados el experimento para evaluar el comportamiento actual del sistema, se tiene una variación promedio de 10.4 % en la exactitud de clasificación. Con esto y con el criterio de los departamentos involucrados se determinó el valor marginal de precisión en 90 % y en 95 % el valor ideal para tener un comportamiento como mínimo igual que el actual.

Tabla 5.3: Desglose de los grados por métrica. Elaboración propia.

No. Métrica	Grado 1	Grado 2	Grado 3
8	El sistema no requiere intervenciones para funcionar correctamente fuera de la acción de inicio y fin del programa.	El sistema requiere mínimas intervenciones, las cuales son periódicas y están bien establecidas y documentadas.	El sistema requiere intervenciones constantemente, las cuales son esporádicas y no están establecidas ni documentadas.
9	En el sistema no se pierde ningún tipo de información.	En el sistema pierde cierta cantidad de información que no pone un riesgo de calidad en el material ,pero sí podría pasar desapercibido condiciones no críticas de la máquina.	En el sistema pierde información que pone un riesgo de calidad en el material y podría pasar desapercibido condiciones de la máquina.
11	No existe diferencia en la precisión o exactitud del sistema según el operario.	Existe una variación inferior al 10 % de la precisión o exactitud del sistema con respecto al operario.	Existe una variación superior al 10 % de la precisión o exactitud del sistema con respecto al operario.

Continúa en la siguiente página

Tabla 5.3 – Continuación de la página anterior

No. Métrica	Grado 1	Grado 2	Grado 3
12	La información solo puede ser accesible y entendida por cierta cantidad de personas con permisos y/o capacitadas.	La información solo puede ser accesible y entendida por los usuarios de departamentos específicos.	La información puede ser accesible y entendida por cualquier colaborador de la empresa.

Finalmente, se resume este conjunto de métricas y valores en la Tabla 5.4 con su unidad.

Tabla 5.4: Especificaciones objetivo del sistema. Elaboración propia.

Núm.	Métrica	Unidad	Valor marginal	Valor ideal
1	Exactitud de detección de fallas críticas.	% Exactitud	≥90	100
2	Exactitud de clasificación por categoría.	% Exactitud	≥80	≥90
3	Cantidad de tipos de fallos detectables.	Cantidad de categorías	≥4	8
4	Velocidad de clasificación	s/imagen	≤0.50	≤0.30
5	Memoria RAM consumida por el sistema.	Gb	≤3.00	≤1.00
6	Ancho de banda consumido por el sistema.	Mbps	≤3.00	≤2.00
7	Costo del software utilizado.	$	≤200	0
8	Grado de intervenciones necesarias para operación del sistema.	Grado de intervenciones	2	1

Continúa en la siguiente página

Tabla 5.4 – Continuación de la página anterior

Núm.	Métrica	Unidad	Valor marginal	Valor ideal
9	Grado de información perdida.	Grado de perdida	2	1
10	Precisión del sistema.	% Precisión	≥90	≥95
11	Grado de oscilación en desempeño según operario.	Grado de oscilación	2	1
12	Grado de dispersión de la información.	Grado de dispersión	2	3

Reflexión

Al seguir la metodología de diseño de ingeniería se llevó a cabo una reflexión una vez terminada esta etapa del diseño para comprobar la correcta utilización del enfoque metodológico. En esta reflexión se determina que con las especificaciones establecidas se puede cuantificar los atributos que requiere el producto final del proyecto correctamente, por lo que se puede concluir que las métricas son correctas. Estas están bien redactadas, ya que hacen referencia a un parámetro cuantificable sin tener un sesgo en su redacción. Estas métricas son variables independientes, prácticas (no necesitan equipos complejos de medición), sin métricas binarias y cada una se relaciona con las necesidades que se plantearon como se muestra en la matriz de la Figura 5.1.

Con respecto a los valores objetivos, se relacionan directamente con las métricas que se plantearon, son parámetros medibles con sus unidades y se definieron mediante un proceso de recolección y validación de información, lo que evita sesgar (sin enfocarse en ninguna tecnología en particular). Es importante mencionar que estos están bien expresados con sus rangos y unidades, además de que se indagó la presencia de elementos estandarizados y la posibilidad de aplicar normas en estas métricas. Por esto, se puede comprobar que estos valores se eligieron correctamente.

5.2. Descomposición funcional del problema

Con la determinación correcta de los requerimientos del sistema por diseñar se realiza la etapa de generación de conceptos. Primero, se lleva a cabo una descomposición funcional, donde se desglosa el problema en subproblemas para plantear posibles conceptos que posteriormente se evalúan y así seleccionar el concepto por desarrollar.

Para generar la descomposición funcional del problema se identificaron las entradas y salidas del sistema por desarrollar mediante un estudio del problema y con un modelo de caja negra se determinaron estas entradas y salidas. Este modelo se muestra en la Figura 5.2, lo que facilita la descomposición funcional del problema en subproblemas más pequeños y así generar posibles conceptos para desarrollar el sistema.

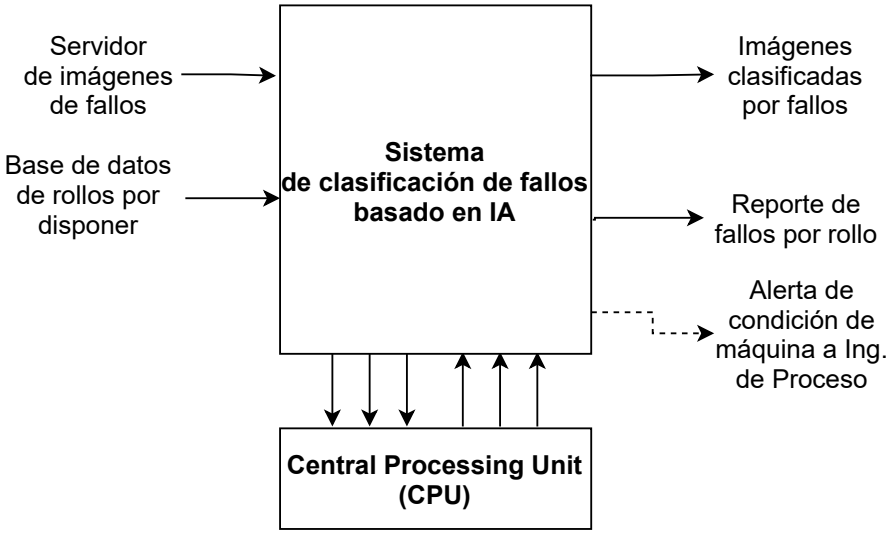


Figura 5.2: Representación del modelo de caja negra de las entradas y salidas del sistema. Elaboración propia.

En el modelo de caja negra de la Figura 5.2 se pueden observar las entradas establecidas por el sistema, estas corresponden únicamente al servidor de imágenes donde se encuentra la imagen de cada fallo detectado en las máquinas de recolección y empaque de *chips*, estas están segregadas por modelo de *chip* y número de identificación del rollo. La otra entrada corresponde a la base de datos de estos rollos, la empresa cuenta con un sistema centralizado para darle seguimiento a todas las líneas de producción, lo que permite determinar cuáles son los rollos que requieren de la inspección de *DTR Map* y permite determinar cuáles imágenes debe clasificar el

sistema.

Con este modelo también se extrajeron las salidas del sistema, estas corresponden a las imágenes correspondientes a cada rollo clasificadas por fallos, el reporte que muestre la información de fallos por rollo y los datos de producción para que cualquier línea o departamento pueda observar los fallos presentes en el proceso de estudio.

Es importante mencionar que es necesaria una comunicación constante con entre el sistema y una Unidad Central de Procesamiento o CPU, ya que al ser un sistema externo a la máquina se necesita de recursos computacionales para llevar a cabo el sistema de clasificación. Esto se representa en la Figura 5.2.

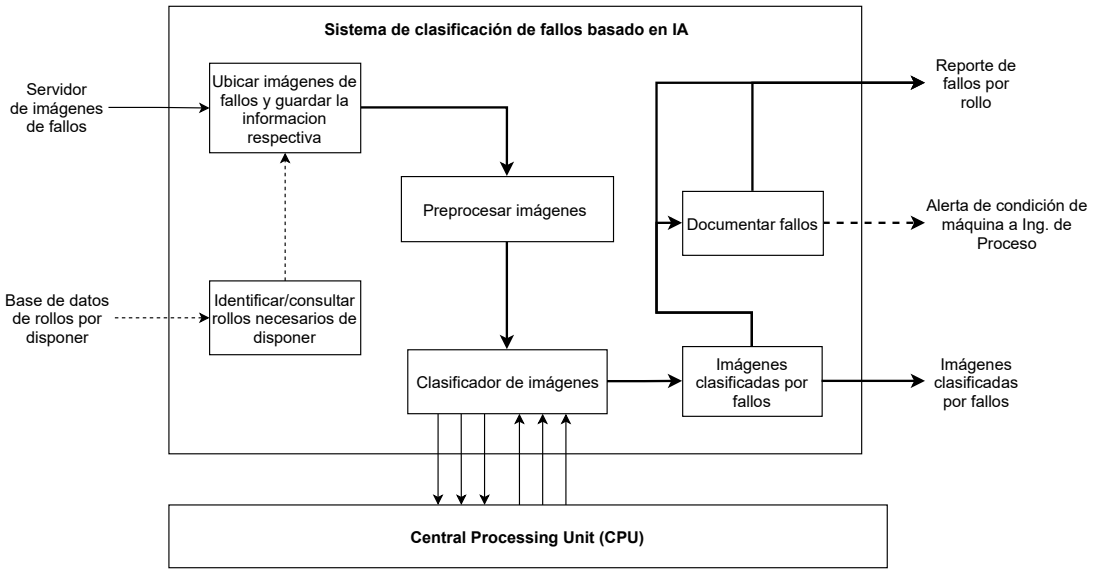


Figura 5.3: Descomposición funcional del modelo de caja negra del sistema. Elaboración propia.

Este modelo facilita la descomposición funcional, la cual se muestra en la Figura 5.3. A partir de esta es posible descomponer el problema en subproblemas que deben solventarse por el sistema por desarrollar. Primero, se deben identificar las imágenes que se deben clasificar por el sistema, esto a partir de la base de datos de los rollos a disponer, para esto, se debe establecer una arquitectura de comunicación que pueda extraer la información de los rollos que necesitan disposición de la base de datos correspondiente. Con esto se debe ubicar cada imagen dentro del servidor de imágenes de los fallos y guardar su información para posteriormente llevar a cabo un preprocesado de estas imágenes.

Con las imágenes preprocesadas se debe determinar a qué clase de fallo pertenece cada

imagen al procesarla en el clasificador de imágenes, lo que da como salida únicamente la categoría de fallo que corresponde. A partir de esto, solo queda documentar el fallo mediante un reporte que pueda vincularse con los datos de producción de la empresa mediante una estructura de comunicación que permita manipular la base de datos y pueda observarse por el personal de los diferentes departamentos. Esto permite informar a los departamentos en caso de un problema de máquina mediante un sistema de umbrales con base en los resultados de clasificación.

Con esto es posible enumerar los siguientes subproblemas para solventar:

- Identificación de rollos necesarios de disponer.
- Ubicación de las imágenes que requieren inspección y lectura de la información.
- Preprocesado y clasificación de imágenes.
- Documentar fallos.

5.2.1. Reflexión

Al seguir la metodología de diseño de ingeniería, se llevó a cabo una reflexión una vez terminada esta etapa del diseño que comprende la descomposición funcional del sistema, en esta se contempla que existe una gran cantidad de formas de descomponer cualquier tipo de problema, según el detalle con el que se quiera profundizar. En este caso se considera que el nivel de detalle de la descomposición funcional que se planteó es adecuada para la naturaleza del problema y permite buscar posibles soluciones que abarquen correctamente los requerimientos de la solución.

5.3. Desarrollo de conceptos por subsistema

Con los requerimientos del problema establecidos, se procede a plantear posibles conceptos que solventen los subproblemas mencionados. Para esto, se llevó a cabo la investigación externa e

interna para establecer las posibles soluciones a estos subproblemas para establecer los conceptos a evaluar. Una vez evaluados se desarrolla cada concepto por subsistema.

Por la naturaleza del problema, se decidió evaluar cada subproblema por separado, ya que no afectaría negativamente a la solución final mientras se mantenga un formato común en el flujo de datos y permite encontrar los conceptos óptimos para abordar el problema.

5.3.1. Identificación de rollos necesarios de disponer

Primero, se debe establecer la estructura para identificar los rollos necesarios para disponer (lo que requieren la inspección). Para esto, se propuso establecer una arquitectura de comunicación que pueda extraer la información de los rollos que necesitan la inspección de la base de datos correspondiente, esto mediante una estructura de Internet de las cosas o IoT que permite extraer la información de los rollos que estén en la operación del proceso que necesitan la evaluación de *DTR Map* junto con la información del modelo, cantidad de piezas, etc.

Para esto, se hizo una investigación interna del sistema y se encontró que esta información se encuentra en el servidor de un Sistema de Ejecución de Manufactura o MES, por lo tanto, el acceso a los datos se complica por la gran cantidad de información presente. Para abordar esto, se llevó a cabo una investigación interna y externa, en la cual se logró extraer que en lo que respecta a manejar la información almacenada en servidores como MES es necesario una estructura de IoT y, como lo mencionan algunos estudios [39] y [14], la opción más adecuada para estos casos es desarrollar un sistema de interfaz de Programación de Aplicaciones o API, un servicio web o una compuerta de Fundación de Comunicación de Windows o WCF según sea la naturaleza del problema. Con esto se decidió evaluar estas opciones para desarrollar esta infraestructura. Es importante mencionar que durante la búsqueda interna se consideraron los protocolos de WCF y API durante una tormenta de ideas.

Una vez que se identifican los conceptos a evaluar para este subproblema, se define una lista de criterios de selección y su importancia para evaluar cada uno. Estos criterios se obtienen al resumir las necesidades mostradas en la Tabla 5.1, adecuándolas para este subproblema y con

su importancia con respecto a lo establecido en la misma tabla.

1. Facilidad de uso.
2. Durabilidad.
3. Precisión de identificar los diferentes tipos de fallos.
4. Facilidad de lectura de datos.
5. Facilidad ajuste.
6. Duración de procesamiento de la imagen.

Estas se muestran gráficamente en el Pareto con su valor porcentual en la Figura 5.4.

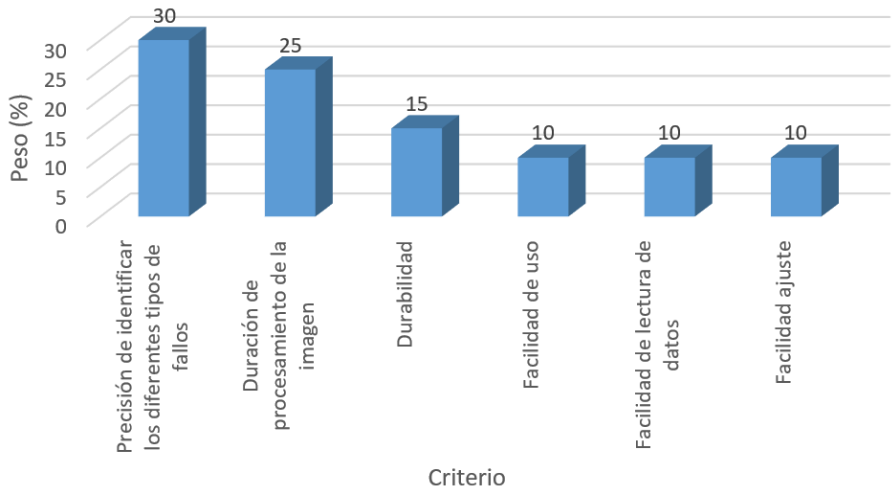


Figura 5.4: Pareto de pesos de cada criterio de selección. Elaboración propia.

Con esto definido se pasa a una etapa de filtrado de estos conceptos y se llevó a cabo una tabla de filtrado de conceptos (Tabla 5.3.1), con la cual se determinaron los conceptos que pasan a la siguiente fase de evaluación, esto lo definen las puntuaciones obtenidas y los conceptos. Es importante que mencionar que se tomó el concepto de un protocolo API como referencia.

Tabla 5.5: Tabla de filtrado de conceptos que se plantearon para identificar los rollos por disponer. Elaboración propia.

Criterios de selección	**Conceptos**		
	A API	**B Web Service**	**C CWF *gateway***
Precisión de identificar los diferentes tipos de fallos.	0	0	0
Duración de procesamiento de la imagen.	0	0	0
Durabilidad.	0	-	0
Facilidad de uso.	0	+	+
Facilidad de lectura de datos.	0	+	+
Facilidad de ajuste.	0	-	+
Suma +	0	2	3
Suma 0	6	2	3
Suma -	0	2	0
Evaluación neta	0	0	3
Lugar	2	2	1
¿Continuar?	**Sí**	No	**Sí**

Una vez realizado la etapa de filtrado se procede con la etapa de evaluación de conceptos, esta se llevó a cabo mediante una tabla de evaluación de conceptos (Tabla 5.3.1). En esta, mediante los puntajes ponderados, se determinó que el concepto ganador es el uso del protocolo WFC *gateway*, este pasó a fase de desarrollo. Cabe destacar que el concepto A tuvo un puntaje similar, pero se terminó escogiendo el concepto C, ya que mediante la investigación interna se observa que es que más se utiliza en la empresa para infraestructuras IoT y ya hay infraestructuras de este tipo en funcionamiento, por lo que es un gran apoyo durante el desarrollo.

Tabla 5.6: Tabla de evaluación de conceptos planteados para identificar los rollos a disponer. Elaboración propia.

Criterios de selección	**Peso**	**Conceptos**			
		A **API**		**C** **CWF *gateway***	
		Calificación	**Evaluación ponderada**	**Calificación**	**Evaluación ponderada**
Precisión de identificar los diferentes tipos de fallos.	30 %	3	0.9	3	0.9
Duración de procesamiento de la imagen.	25 %	3	0.75	4	1
Durabilidad.	15 %	3	0.45	2	0.3
Facilidad de uso.	10 %	3	0.3	5	0.5
Facilidad de lectura de datos.	10 %	3	0.3	4	0.4
Facilidad de ajuste.	10 %	3	0.3	1	0.1
Total puntos		18	3	19	3.2
Lugar		2		**1**	
¿Continuar?		No		**Desarrollar**	

Una vez establecido el concepto por desarrollar, se diseña un código que pueda consultar la información de los rollos que esta en la operación de *Reel Disposition* y después determinar la dirección en el servidor de las imágenes correspondientes a cada rollo que es necesario efectuar la clasificación.

Para esto, se diseñó un programa en Python 3.7 que pueda consultar al servidor MES por medio de un WCF. Como se mencionó, la empresa cuenta con WCF en el que consultan información específica del sistema MES. Para el sistema por desarrollar, se utiliza el servidor de la empresa `http://sjo0ws09.corp.qorvo.com:50014/WcfService/Service.svc?singleWsdl`, el el cual cuenta con servicios WCF variados que utiliza la empresa. Después de un estudio de los métodos del servidor y sus especificaciones, los cuales se detallan en [18], se decidió utilizar el método con la denominación LotWIPOnlineQuery_Camstar que retorna la información de los rollos que están en una operación específica de la planta (la cual se ingresa como parámetro).

Con esto, se decidió utilizar una llamada remota a este procedimiento o RPC (Remote Procedure Call) por medio de la biblioteca **suds** de Python [33] que brinda este tipo de consulta, como se muestra en el código 5.1 [1].

```
from time import time
from datetime import datetime
from suds.client import Client
from VCTQ_Automatization import *
##=============================================================
## Consultar rollos en Reel Disposition
## Retorna  diccionario con [Rollo,Linea,Modelo,Fecha (fecha en formato datetime)]
##=============================================================
def ConsultarWCF():
    # Tiempo actual para determinar tiempo de consulta
    start_time=time()
    # Conectar con servidor de consultas
    server = "http://sjo0ws09.corp.qorvo.com:50014/WcfService/Service.svc?singleWsdl"
    client = Client(server)
    # Consultar rollos en 'Reel Disposition'
    result = client.service.LotWIPOnlineQuery_Camstar("Reel Disposition")
    # Tiempo de consulta
    print('==========================================================================')
    print('Tiempo de consulta: ', seg_to_time(time()-start_time))
    print('==========================================================================')
    # Diccionario para guardar consulta
    Pendientes=[]
    # Arreglar formato de las variables del rollo
    for i in result[0]:
        rollo=i[0]
        # Identificar linea del rollo
        if not rollo.find('CR5')==(-1):
            linea='WLP'
        elif not rollo.find('CR6')==(-1):
            linea='uBAW'
        modelo=i[1].split(' ',1)[0]
        modelo=i[1].split('TR13',1)[0]
        qty=[2]
        # Pasar fecha a formato datetime
        fecha=datetime.strptime(i[3], '%Y-%m-%dT%H:%M:%S.%f%z')
        # Guardar rollo en diccionario
        Pendientes.append({'Rollo':rollo,'Linea':linea,'Modelo':modelo, 'Fecha':fecha})
    # Retornar diccionario
```

[1] En este código se utiliza la biblioteca de elaboración propia anexada en A.1.1 y sus funciones.

```
39     return Pendientes
```

Código 5.1: Código para consultar datos de rollos a clasificar mediante WCF.

Como retorno de este procedimiento se obtienen cuatro variables que contienen la identificación del rollo, el modelo, la cantidad de piezas en el rollo y la fecha en que se agregó a la operación en formato tring', tring', 'int' y tring' respectivamente [18]. Al desarrollarlo, se identificó que es necesario detectar de que línea proviene el rollo (**Línea A** y **Línea B**), el cual se puede extraer de la estructura de la identificación del rollo (prefijo CR5 y CR6 para las Líneas A y B respectivamente). Además, se modificó la estructura que denomina al modelo del rollo, ya que no correspondía con la del servidor donde se encuentran las imágenes por rollo; todo esto también se muestra en el código. 5.1.

5.3.2. Ubicación de las imágenes que requieren inspección y lectura de la información

Una vez establecido el concepto para la identificación, se necesita establecer la forma de acceder a la información de las imágenes que se deben clasificar y cómo hacer su lectura. Para esto, primero se hizo un análisis del sistema actual de inspección y cómo está compuesto el sistema de que obtiene las imágenes que se inspeccionan.

Aquí interviene una de las limitantes que se mencionaron anteriormente, la cual es la imposibilidad de acceder al *software* propio de la máquina de DTR, por lo que solo se tiene acceso a la información de salida del sistema de visión (las propias imágenes), las cuales se alojan en un servidor de almacenamiento de datos propio de la empresa que se basa en Windows Server 2019 con la dirección '
sjo0fs02_images'. Al llevar a cabo una investigación interna y externa, se encontró que el protocolo para acceder a un servidor de almacenamiento de datos deriva directamente de los sistemas operativos que se estén comunicando [12]. Por lo tanto, la decisión reside en determinar en que sistema operativo se utiliza para operar el sistema.

En la empresa se dispone de servidores que se basan en Windows y Linux, por lo que la

decisión se limita a una comunicación con protocolo SMB (Server Message Block) o protocolo NFS (Network File System) [13]. Es importante mencionar que el sistema se está diseñando en Python por lo que no afecta si se diseña en un sistema que se basa en Linux o Windows.

Para determinar esto, se llevó a cabo una tabla de evaluación de conceptos (Tabla 5.3.2), con base en los criterios aplicables que se plantearon anteriormente (5.4). En esta tabla se suma un punto por cada '+' y se resta un punto por cada '-' tomando el concepto de SMB como referencia por conveniencia.

Tabla 5.7: Tabla de evaluación de conceptos que se plantearon para obtener información de las imágenes de cada rollo. Elaboración propia.

Criterios de selección	**Conceptos**	
	A **Protocolo SMB**	**B** **Protocolo NFS**
Durabilidad.	0	0
Facilidad de uso.	0	-
Facilidad de lectura de datos.	0	-
Facilidad de ajuste.	0	+
Total puntos	0	-1
Lugar	**1**	2
¿Continuar?	**Desarrollar**	No

Con esto, se determina que el concepto por desarrollar es un protocolo SMB por su facilidad de ajuste, uso y porque tiene mayor compatibilidad con la mayoría de los servidores y bases de datos de la empresa que se basan en Windows.

Para el desarrollo de este subsistema se utilizó un servidor que se basa en Windows Server 2019 y se decidió mapear una localización de red a la dirección ' sjo0fs02_images' (donde se almacenan las imágenes) con base en un protocolo SMB, el cual se llevó a cabo con éxito.

Una vez establecida la conexión, se genera un código que pueda determinar la ubicación de las imágenes con base en la información del rollo (número de identificación, modelo y línea),

además de que durante la investigación interna se detectó que estas imágenes se encuentran siempre en la subcarpeta que cuenta con la palabra 'station1' o 'top' entre las subcarpetas con la identificación y modelo. Esto se muestra en el código 5.2.

```
import os
##=============================================================
## Obtener direcci n del rollo respectivo
## Parametro = Diccionario con informaci n del rollo
##=============================================================
def Obtener_path(rollo): # rollo = return de ConsultarWCF()
    ## Obteniendo ubicaci
    root=os.path.join('\\sjo0fs02\DTR\DTR_images', rollo['Modelo'],rollo['Rollo'])
    if not os.path.exists(root): # Verificar si existe
        return None
    else:
        ## Determinando carpeta a clasificar
        for i in os.listdir(root):
            # Ubicar carpeta que contenga 'station1'
            if re.search('station1', i, re.IGNORECASE):
                # Ubicar carpeta que contenga 'Presence'
                if os.path.exists(os.path.join(root,i,'Presence')):
                    return os.path.join(root,i,'Presence')
                else:
                    return os.path.join(root,i)
            # Ubicar carpeta que contenga 'top'
            elif re.search('top', i, re.IGNORECASE):
                if os.path.exists(os.path.join(root,i,'Presence')):
                    return os.path.join(root,i,'Presence')
                else:
                    return os.path.join(root,i)
```

Código 5.2: Código para determinar dirección de imágenes del rollo correspondiente.

Cabe mencionar que por la naturaleza de este subproblema no es necesario generar una validación, ya que una vez establecido la conexión con el protocolo SMB y verificando su acceso al servidor se comprueba su funcionamiento. La validación completa se hace junto con los demás subsistemas.

5.3.3. Preprocesado y clasificación de imágenes

Para desarrollar el sistema de clasificación de imágenes se llevó a cabo la investigación interna y externa, según la metodología que se planteó. En esta se determinaron conceptos posibles para la solución del problema.

Primero se debe establecer el concepto o paradigma por desarrollar para la clasificación de imágenes, ya que los posibles preprocesados necesarios dependen del paradigma determinado. Para esto, se procedió a establecer el concepto bajo el cual se desarrolla el sistema de clasificación de imágenes.

Se encontró que los conceptos por utilizar más eficaces para un sistema de clasificación de imágenes son los sistemas de visión computacional y paradigmas de inteligencia artificial [15] [24] [29] [37]. Este último es un concepto que se puede subdividir en múltiples paradigmas; los más relevantes para la naturaleza del problema son los aprendizajes automáticos supervisados y no supervisados. Para este problema se pueden extraer del aprendizaje supervisado las redes neuronales (NN), redes neuronales convolucionales (CNN), perceptrón multicapa (MLP) y redes neuronales recurrentes (RNN) y del no supervisado el análisis de componentes principales (PCA) [24].

Del aprendizaje automático, las redes neuronales convolucionales superaron a los demás métodos y han sido exitosas en esta área [4], [22], [25] y [38] Estas destacan por su eficacia en varios entornos (aprendizaje supervisado, por capas, entrenamiento previo sin supervisión, etc.).

Con esto se determinó que los posibles conceptos por desarrollar se basan en sistemas de visión, redes neuronales convolucionales y análisis de componentes principales. Para esto se llevó a cabo una tabla de evaluación de conceptos (Tabla 5.3.3), con base en los criterios descritos (5.4). En esta tabla se suma un punto por cada '+' y se resta un punto por cada '-' tomando al concepto de un sistema de visión que es el que actualmente se encuentra en la empresa.

Tabla 5.8: Tabla de evaluación de conceptos planteados para el sistema de clasificación. Elaboración propia.

Criterios de selección	**Conceptos**		
	A Sistema de Visión	**B CNN**	**C PCA**
Precisión de identificar los diferentes tipos de fallos.	0	+	+
Duración de procesamiento de la imagen.	0	0	0
Durabilidad.	0	+	+
Facilidad de uso.	0	0	0
Facilidad de lectura de datos.	0	+	+
Facilidad de ajuste.	0	-	-
Total puntos	0	2	2
Lugar	2	**1**	**1**
¿Continuar?	No	**Combinar**	**Combinar**

En la Tabla 5.3.3, se determina que los conceptos de CNN y PCA se deben combinar y desarrollar, ya que son los más eficaces según los criterios establecidos. Los sistemas de visión se descartan, principalmente por la imposibilidad de controlar las condiciones toma de imágenes, ya que únicamente se tiene acceso a las imágenes en baja calidad que guarda la máquina de DTR después de su inspección, lo cual genera una importante disminución en la precisión que se puede conseguir y la durabilidad que tienen. Esto último se debe a que se cambian constantemente las propiedades de las imágenes, según la máquina en que se tome, el operario que la esté usando, el material que se procese, etc.

Una vez establecido el concepto que se desarrolla, se procede a elaborar un conjunto de datos de entrenamiento, ya que para el paradigma que se planteó de aprendizaje automático es necesario tener un conjunto de imágenes clasificadas bajo las categorías establecidas en el estudio mecánico descrito en 4.

Para desarrollar este conjunto de datos, primero se debe definir la clasificación que se desea obtener del sistema. Esto se define a partir del estudio mecánico descrito en 4, con base en el cual, junto con el Departamento de Disposición y de proceso, se tomó la decisión de juntar las categorías de contaminación, problemas de visión y mal recolectado. A pesar de que esto lo provocan diferentes causas, estas vienen directamente de la máquina de DTR y las acciones por tomar son muy similares. Además, ninguna representa un problema de calidad y solo conciernen al Departamento de Proceso para que intervenga la máquina. Con esto se facilitó tanto la clasificación como las decisiones para la disposición del material. Las categorías definidas son las siguientes:

- Líneas de pulido (*BG_Lines*)
- Dispositivo quebrado (*Broken_Die*)
- Muescan en dispositivo (*Chip_Out*)
- Boquilla vacía (*Empty_Nozzel*)
- Fallo por aguja (*Needle_Issue*)
- Problema de recolección (*Recolection_Issue*)

Se desarrolló un conjunto de datos que cuenta con 15004 imágenes repartidas en las diferentes categorías, esto se muestra en la Tabla 5.9. El conjunto de datos se desarrolló eligiendo cinco rollos al azar de cada modelo procesado en las líneas y clasificando manualmente cada una de las imágenes. Es importante mencionar que a raíz de la diferencia de las cantidades de imágenes por categoría es necesario balancear el conjunto de datos y darle mayor peso a las categorías que tienen menor cantidad de imágenes, esto se muestra en la función de 'EntrenamientoCNN' de la biblioteca mostrada en A.1.1.

Después de obtener el conjunto de datos de entrenamiento se establece el preprocesado de las imágenes. Para este se hizo un análisis de las imágenes con la ayuda de los histogramas de las imágenes (en escala de grises), como se muestra en la Figura 5.5, donde se muestra un

Categoría	# imágenes
Líneas de pulido	3141
Dispositivo quebrado	1269
Muesca en dispositivo	1293
Boquilla vacía	4230
Fallo por aguja	630
Problema de recolección	4441
Total	15004

Tabla 5.9: Descripción del conjunto de datos por categoría.

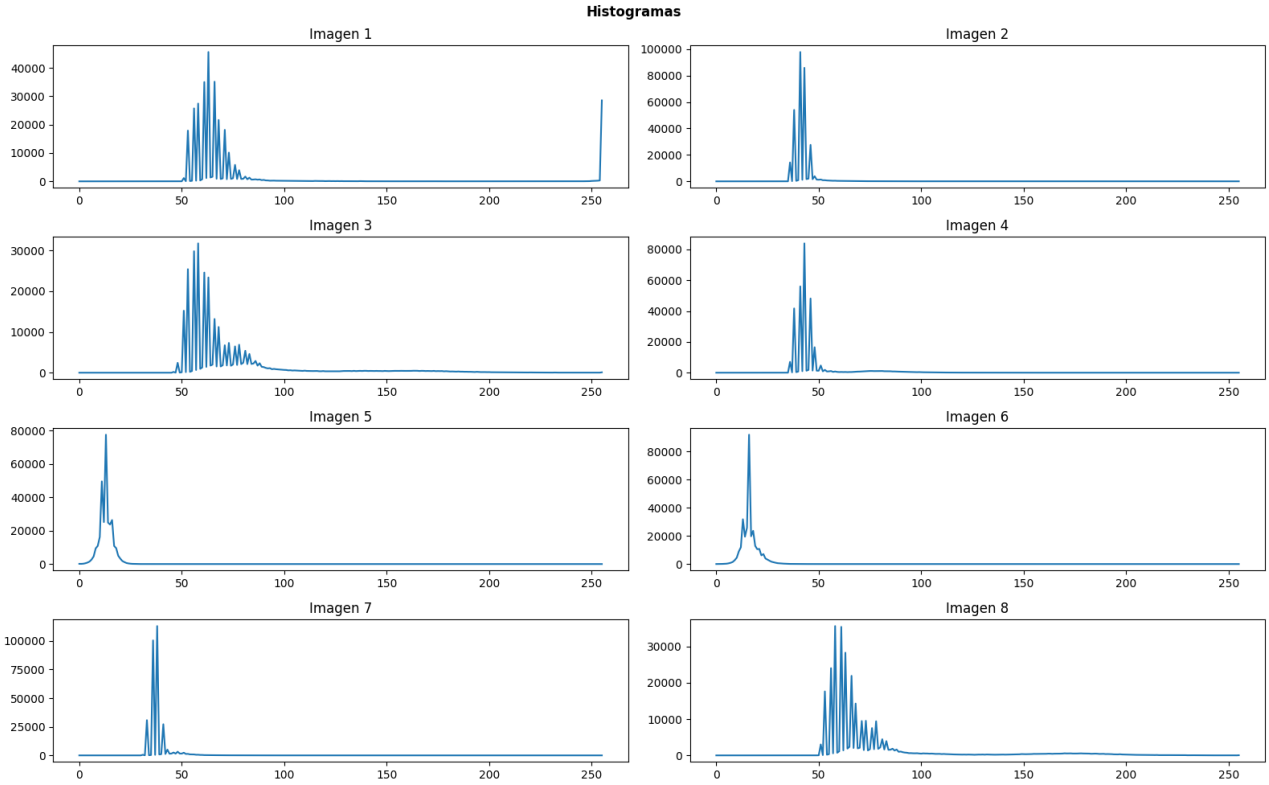


Figura 5.5: Histogramas de una muestra de 8 imágenes del conjunto de datos de entrenamiento. Elaboración propia.

indicio de ruido gaussiano con base en el rizado presente en cada histograma y un evidente bajo contraste.

Para solventar este problema, primero se estandarizó el tamaño de cada imagen a 66 x 50 píxeles, que es el tamaño mínimo que se encuentra en cada una de las máquinas. Posteriormente, se plantearon tres posibles algoritmos que mejoran el contraste y facilitan la clasificación mediante un sistema de aprendizaje automático, los cuales se evaluarán posteriormente junto con el algoritmo de clasificación principal con un factorial completo de todos los factores pertinentes. Estos se basan en algoritmos de ecualización ('Clean'), detección de bordes Canny ('Canny') y

Binarización adaptativa junto con un filtro Sobel respectivamente ('Sobel'). Estos se muestran en el código 5.3.

```
import cv2
##=============================================================
## Preprocesar la imagen mediante Sobel y binarizacion adaptativa
## Parametros: (Direccion de la imagen)
## Return (Imagen binarizada)
##=============================================================
def Sobel(img_path):
  ## Recibe direccion de la imagenes como parametro
  ## Cargando imagen
  img=cv2.imread(img_path, cv2.IMREAD_GRAYSCALE)
  ## Filtro de ruido gaussiano
  img = cv2.GaussianBlur(img,(3,3),0)
  ## Aplicando convoluccion (sobel)
  sobelx = cv2.Sobel(img,cv2.CV_64F,1,0,ksize=5)  # En X
  sobely = cv2.Sobel(img,cv2.CV_64F,0,1,ksize=5)  # En Y
  img=sobelx+sobely # AND para obtener sobel en X y Y
  ## Normalizando datos entre 0 y 255 para poder binarizar
  img=cv2.normalize(img, None, alpha = 0, beta = 255, norm_type = cv2.NORM_MINMAX, dtype = cv2
    .CV_16UC1)
  ## Binarizando usando Otsu (binarizado adaptativo)
  ## Se tiene binarizado en valores de 0 y 255
  (T, img) = cv2.threshold(img, 0, 255, cv2.THRESH_BINARY_INV | cv2.THRESH_OTSU)
  return img

##=============================================================
## Preprocesar la imagen unicamente con filtro de ruido y ecualizacion
## Parametros: (Direccion de la imagen)
## Return (Imagen binarizada)
##=============================================================
def Clean(img_path):
  ## Recibe direccion de la imagenes como parametro
  ## Cargando imagen
  img=cv2.imread(img_path, cv2.IMREAD_GRAYSCALE)
  ## Filtro de ruido gaussiano
  img = cv2.GaussianBlur(img,(3,3),0)
  ## Ecualizar
  clahe = cv2.createCLAHE()
  equalized = clahe.apply(img)
  return equalized

##=============================================================
## Preprocesar la imagen para solo tener contornos mediante Canny
## Parametros: (Direccion de la imagen)
```

```
## Return (Imagen de contornos)
##=============================================================
def Canny(path):
  img=cv2.imread(path, cv2.IMREAD_GRAYSCALE)
  gray = cv2.GaussianBlur(img,(3,3),0)
  # Encuentra el gradiente en la direcci          n X
  grad_x = cv2.Sobel(gray, cv2.CV_16SC1, 1, 0)
  # Encuentra el gradiente en la direcci          n y
  grad_y = cv2.Sobel(gray, cv2.CV_16SC1, 0, 1)
  # Convertir el valor del gradiente a 8 bits
  x_grad = cv2.convertScaleAbs(grad_x)
  y_grad = cv2.convertScaleAbs(grad_y)
  # Combina dos gradientes
  src1 = cv2.addWeighted(x_grad, 0.5, y_grad, 0.5, 0)
  # Combinar gradientes con algoritmo canny, donde 50 y 100 son umbrales
  edge = cv2.Canny(src1, 50, 100)
  #print(edge)
  return edge
```

Código 5.3: Funciones para preprocesado de las imágenes.

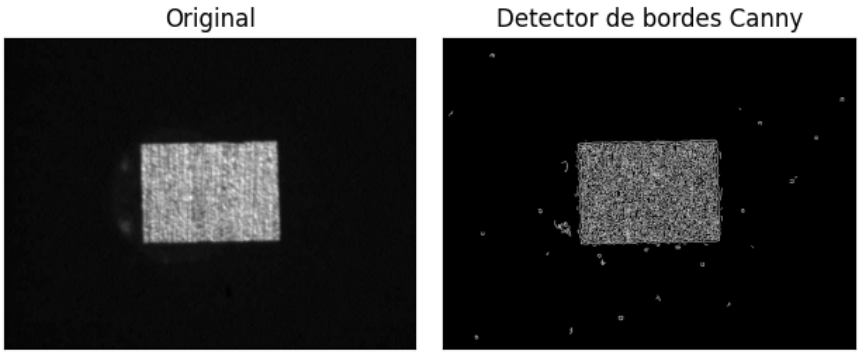


Figura 5.6: Procesamiento de la imagen mediante detector de borde Canny. Elaboración propia.

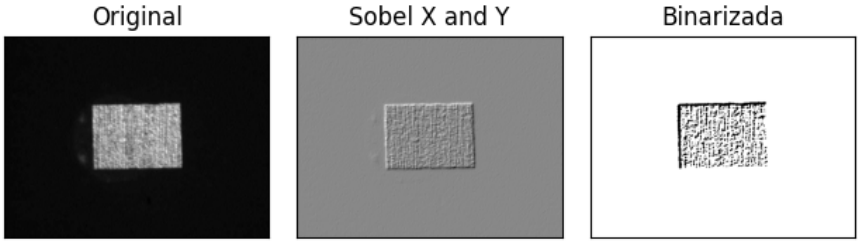


Figura 5.7: Procesamiento de la imagen mediante filtro Sobel y binarización adaptativa. Elaboración propia.

Los resultados se representan en la Figura 5.6 y la Figura 5.7, las cuales corresponden al preprocesamiento de una imagen del conjunto de datos con el detector de bordes Canny y la

binarización adaptativa con un filtro Sobel respectivamente. Además, se observa en la Figura 5.8 la imagen únicamente con el filtro gaussiano y la ecualización.

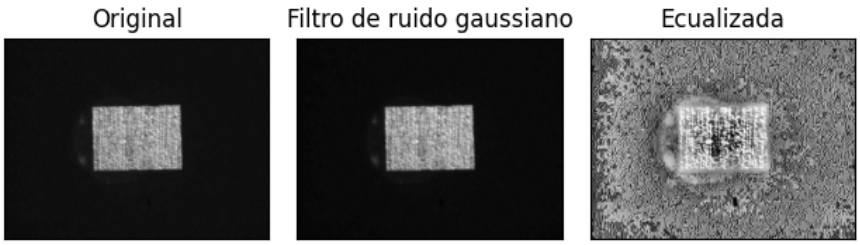


Figura 5.8: Procesamiento de la imagen con filtro gaussiano y ecualización. Elaboración propia.

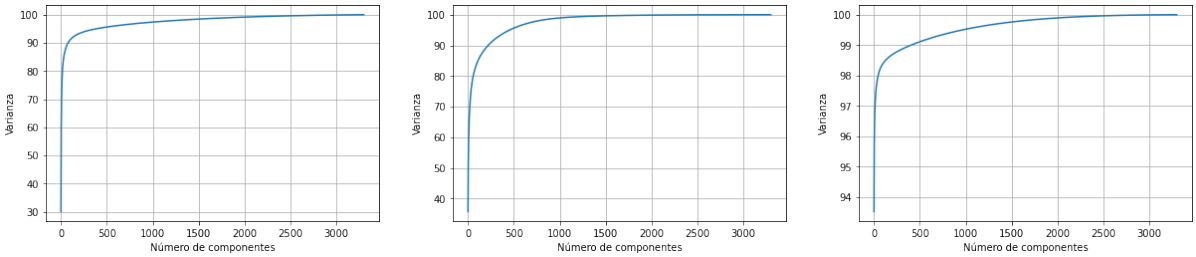
Una vez determinados los algoritmos de preprocesamiento posibles, se aplica el análisis de componentes principales sobre las imágenes del conjunto de datos de entrenamiento. Para este se utiliza el algoritmo PCA de la biblioteca 'Sklearn', como se muestra en la función 'Imgs_PCA' del código 5.9 de la biblioteca propia A.1.1.

```
1  ##=============================================================
2  ## Creando modelo de analisis PCA y transformando DataSet
3  ## (Guardandolo si es requerido)
4  ## Return: Modelo, Datos transformados
5  ##=============================================================
6  def Img_PCA(images, label, nombre="", PCA_save=False, Data_save=False):
7    ## Tiempo de inicio
8    start_time = time()
9    ## Crear modelo PCA de n componentes
10   pca = PCA()
11   ## Creando modelo basado en informacion del DataSet
12   pca = pca.fit(images)
13   ## Transformando el data Set
14   pcas = pca.transform(images)
15   print('==============================================================================')
16   print("Se ha creado correctamente el modelo PCA y transformado el DataSet")
17   print('==============================================================================')
18   ## Guardando modelo de ser necesario
19   if (PCA_save):
20     with open(nombre+'_PCA.pkl', 'wb') as f:
21       pickle.dump(pca, f)
22   print('==============================================================================')
23   print("Se ha guardado exitosamente el modelo PCA")
24   print('==============================================================================')
25   ## Guardando DataSet con PCA de ser necesario
26   if (Data_save):
27     columns=[]
28     for i in range (0,n):
```

```
29        columns.append('C%i' %(i+1))
30
31      principalDF = pd.DataFrame(data = pcas, columns = columns)
32      finalDF = principalDF.to_dict()
33      namess={}
34      namesss={}
35      for i in range(0,len(finalDF['C1'])):
36        namess[i]=label['Fail'][i]
37
38      finalDF.update({'Fails':namess})
39      df=pd.DataFrame.from_dict(finalDF)
40      df.to_csv (os.path.join(nombre+"_with_PCA.csv"), index = False, mode="a", header=not os.
        path.isfile(nombre+"_with_PCA.csv"))
41    print('==========================================================================================')
42    print("Se a guardado exitosamente el DataSet transformado con el modelo PCA")
43    print('==========================================================================================')
44    ## Finalizando la funcion
45    ## Tiempo de ejecucion
46    print('==========================================================================================')
47    print("Tiempo de carga de la base de datos: ", seg_to_time(time() - start_time))
48    print('==========================================================================================')
49    ## Return modelo y array de imagenes por componente
50    return pca, pcas
```

Código 5.4: Función para análisis de componentes principales del conjunto de datos de entrenamiento.

Para utilizar este algoritmo es necesario establecer el número de componentes (dimensiones) que se desea calcular, para esto, se utiliza el algoritmo con el número original de componentes (66x50 = 3300) para obtener posteriormente la gráfica de proporción de varianza explicada acumulada para los conjuntos de entrenamiento con cada uno de los preprocesamientos. Estas gráficas se muestran en la Figura 5.9, y permiten observar el porcentaje de información que está presente en los datos según cuantos componentes tengan. Esto permite definir un número adecuado de componentes principales por utilizar, que en este caso se definió en 500, ya que con este valor se tiene un porcentaje superior a 90 % de la información en estos componentes, lo cual es aceptable para la aplicación, según el criterio propio y a la documentación de [3]. Al aplicar esto se pasa de 3300 a 500 componentes a evaluar, obteniendo una reducción del 84.84 % y conservando más del 90 % de las características de las imágenes para los tres algoritmos de preprocesamientos.



(a) Ecualización (b) Sobel con binarización adaptativa (c) Detector de bordes Canny

Figura 5.9: Proporción de varianza explicada acumulada vs cantidad de componentes al aplicar PCA al conjunto de entrenamiento con los distintos preprocesados.

Una vez aplicado PCA a los conjuntos de datos de entrenamiento, se procede a diseñar el sistema de clasificación que se basa en CNN utilizando la biblioteca de TensorFlow en Python 3.7 [16]. Para este sistema se estableció una arquitectura fuente de red neuronal convolucional como punto de partida para los experimentos de desarrollo de la red. Esta arquitectura fuente se definió con base en los estudios [15] [24], criterio propio y pruebas preliminares con un conjunto de entrenamiento reducido (6274 y únicamente 3 categorías). Esta se define con el código 5.4 y consta de capa convolucional de 64 neuronas, una capa de 'Max Pooling' con un kernel de 2x2, una capa de 'Dropout', una capa conectada completamente de 32 neuronas y otra capa de 'Dropout', en ese orden, para terminar en la capa de salida. Esta se representa en la Figura 5.10 y se utilizó la función 'CreacionCNN1' del código A.1.1.

```
1  ##=============================================================
2  ## Creando la redes neuronales convolutiva
3  ## Parametros: nClasses,activation
4  ## Return: failures_model
5  ##=============================================================
6  def CreacionCNN1(nClasses,activation):
7    ## Tiempo de inicio de ejecucion
8    start_time = time()
9    failures_model = Sequential()
10   failures_model.add(Conv2D(64, kernel_size=(3, 3),activation=activation,padding='same',
       input_shape=(50,66,1)))
11   failures_model.add(LeakyReLU(alpha=0.1))
12   failures_model.add(MaxPooling2D((2, 2),padding='same'))
13   failures_model.add(Dropout(0.5))
14   failures_model.add(Flatten())
15   failures_model.add(Dense(32, activation=activation))
16   failures_model.add(LeakyReLU(alpha=0.1))
17   failures_model.add(Dropout(0.5))
```

```
18  failures_model.add(Dense(nClasses, activation=activation))
19  ## Mostrando resumen de red
20  failures_model.summary()
21  return failures_model
```

Código 5.5: Código para creación de la arquitectura de la red neuronal fuente.

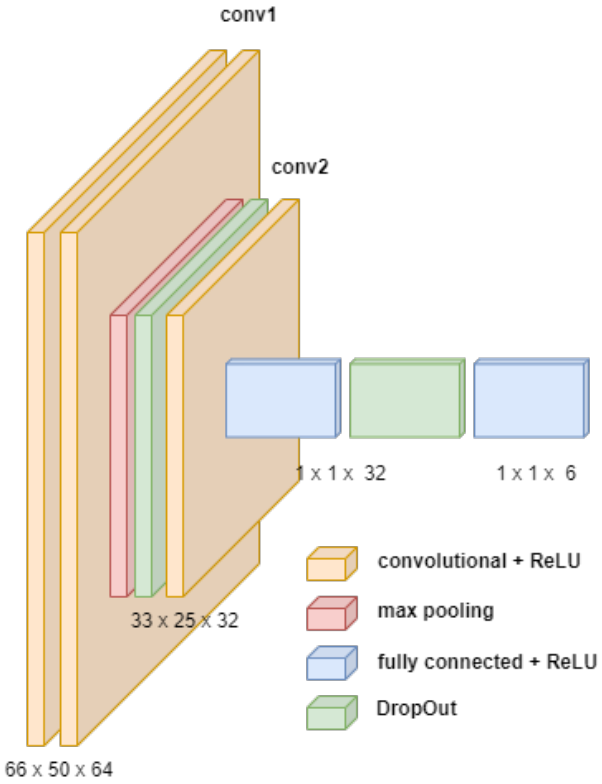


Figura 5.10: Arquitectura de la CNN fuente. Elaboración propia.

Establecida la arquitectura base o fuente se lleva a cabo un experimento factorial completo con todos los hiperparámetros adecuados, los distintos preprocesados y tres distintas CNN para evaluar el comportamiento de los sistemas y elegir el adecuado. Para elegir todas estas variables se toma en cuenta los estudios [15] [24] y la documentación [8] [16], además del criterio propio y pruebas preliminares con un conjunto de entrenamiento reducido.

Para las tres distintas redes se toma la arquitectura fuentes de la Figura 5.10 y se hace dos variaciones, la primera cambiando la cantidad de neuronas de la capa convolucional a 32 y en la segunda se cambia la cantidad de neuronas de la capa convolucional a 64 y la cantidad de neuronas de la capa conectada completamente a 64. Estas tres redes se denominarán 'Arquitectura 1', 'Arquitectura 2' y 'Arquitectura 3' en el resto del documento, en ese orden.

En el caso de los hiperparámetros, se tomó en consideración lo descrito [8], estos son la taza de aprendizaje, las generaciones, el tamaño del lote, el optimizador, el método para calcular el error y la función de activación. Para la taza de aprendizaje se escogieron los valores de 0.0001, 0.00001 y 0.00001 y para el tamaño del lote 10, 100 y 200; estos dos hiperparámetros se relacionan directamente [8] y los valores elegidos son los que se comportaron mejor durante las pruebas preliminares.

Para el optimizador y el método de calcular el error se escogieron los que se pueden adecuar mejor a la naturaleza del problema (clasificación de más de dos categorías) [16], [15], [24], [8] y el criterio propio. Los optimizadores escogidos son el **optimizador de gradiente descendiente con momentum o SGD**, el optimizador que se basa en el algoritmo **RMSprop** el optimizador que se basa en el algoritmo **Adagrad**. En el caso de los métodos de calcular el error se escogen el cálculo de la media de los cuadrados de errores o **'MeanSquaredError'**, el cálculo de la pérdida de entropía cruzada o **'CategoricalCrossentropy'** y el cálculo de pérdida de eje categórica o **'CategoricalHinge'**.

Con respecto a las funciones de activación, se escogió únicamente la función **'relu'**, a raíz de que, según la documentación que se mencionó, es la más adecuada para redes neuronales, lo cual se validó en las pruebas preliminares.

Para la cantidad de generaciones únicamente se escogió un valor de 30, ya que según las pruebas preliminares es suficiente para ver el comportamiento del entrenamiento y en los casos en los que se necesitaron más generaciones se evalúan en otros experimentos. Los valores y variables para evaluar mediante el factorial completo se muestran en la Tabla 5.10. Esto implica que se entrena un total de 729 redes neuronales con diferente configuración de estas variables para evaluar su comportamiento. Una vez definido este experimento se procede a ejecutarlo con el código 5.6, con un tiempo de ejecución aproximado de 7.5 días.

```
## Importando TensorFlow
import tensorflow as tf
## Verificando GPU
device_name = tf.test.gpu_device_name()
if device_name != '/device:GPU:0':
  raise SystemError('GPU device not found')
print('Found GPU at: {}'.format(device_name))
```

Arquitectura de la red	Taza de aprendizaje	Generaciones	Tamaño del lote	Optimizador	Cálculo del error	Función de activación	Preprocesado
Arquitectura 1	0.0001	30	10	SGD	Mean Squared Error	Relu	Ecualización
Arquitectura 2	0.00001		100	RMSprop	Categorical Crossentropy		Sobel con binarización adaptativa
Arquitectura 3	0.000001		200	Adagrad	Categorical Hinge		Detector de bordes Canny

Tabla 5.10: Valores de las variables para el experimento de factorial completo. Elaboración propia

```
## Importando bibliotecas necesarias
import sys,os,re
from time import time
import pickle
## Importando biblioteca propia
from VCTQ_Automatization import *
## Definiendo variables
DataSet='Data Set/PCA'
TestSize=0.2
n=500
## Definiendo experimento
generaciones=[30]
batches=[10,100,200]
LRs=[0.0001,0.00001,0.000001]
preprocess=['Clean','Canny','Sobel']
activations=['relu']
optimizers=['SGD','RMSprop','Adagrad']
losses=['CategoricalCrossentropy','MeanSquaredError','CategoricalHinge']
redes=[1,2,3]
experimento=1
##=================================================================
## Experimento (factorial completo)
##=================================================================
## Tiempo de inicio de ejecucion
start_time = time()

## Experimento (factorial completo)
## Tiempo de inicio de ejecucion
start_time = time()

for preproces in preprocess:
  ## Cargando Base de datos
  train_img, valid_img, train_label, valid_label, test_img, test_Y_one_hot, nClasses, failures
    =ConjuntosEntrenamiento_PCA(os.path.join(DataSet,preproces),n,TestSize)
  for red in redes:
```

```
    for activation in activations:
      for epoch in generaciones:
        for batch in batches:
          for lr in LRs:
            for loss in losses:
              for optimizer in optimizers:
                if(experimento>0): # Empezar en determinado experimento/iteracion
                ## Crear CNN
                  if red==1:
                    failures_model = CreacionCNN1(nClasses,activation)
                  elif red==2:
                    failures_model = CreacionCNN2(nClasses,activation)
                  elif red==3:
                    failures_model = CreacionCNN3 (nClasses,activation)
                  print('==========================================================================
    ')
                  print('Experimento #', experimento)
                  print('==========================================================================
    ')
                  EntrenamientoCNN('Experimento PCA/Experimento_'+str(experimento)+'.h5py',
    experimento, red, failures_model, failures, preproces, epoch, batch, lr, loss, optimizer,
    activation, train_img, valid_img, train_label, valid_label, test_img, test_Y_one_hot)

                  failures_model=0
                experimento+=1
  train_img=0
  valid_img=0
  train_line=0
  valid_line=0
  train_labe=0
  valid_label=0
  test_img=0
  test_line_one_hot=0
  test_Y_one_hot=0
  failures=0
  linea=0

print('==========================================================================')
print('El experimento finalizo con exito')
print("Tiempo de duracion del experimento: ", seg_to_time(time() - start_time))
print('==========================================================================')
```

Código 5.6: Código para ejecución del experimento de evaluación de variables de entrenamiento (factorial completo).

En la primera ejecución del experimento se volvió evidente que no se estaba entrenando correctamente las redes neuronales, ya que al observar la gráfica de precisión de cada uno de los entrenamientos se generaba una asíntota en el mejor de los casos, en aproximadamente 30 % de precisión, como se ejemplifica en la Figura 5.11 que muestra la iteración 11, esta fue la que tuvo mejor comportamiento. Con base en esta gráfica, se plantea que las redes se están sobreentrenando y que el sistema está clasificando las imágenes únicamente en dos de las seis categorías, debido al valor de la asíntota en aproximadamente 30 %, que coincide con 2/6 que es la precisión que se obtendría si todas las clasificaciones dan como resultado únicamente dos categorías.

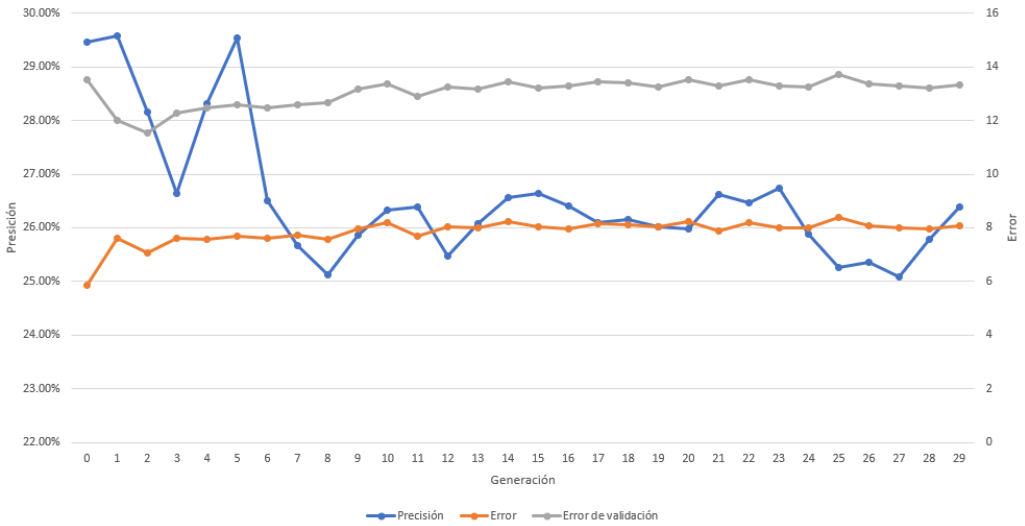


Figura 5.11: Gráfica de precisión y error de la iteración 11 de la primera ejecución del experimento. Elaboración propia.

Categoría	**# imágenes**
Líneas de pulido	223
Dispositivo quebrado	30
Muesca en dispositivo	3
Boquilla vacía	1375
Fallo por aguja	22
Problema de recolección	341
Total	1969

Tabla 5.11: Descripción del conjunto de datos para evaluación por categoría.

Para validar esta hipótesis se hizo un conjunto de imágenes de validación de 1994 imágenes provenientes 6 lotes de producción, las cuales están distribuidas en las categorías según la Tabla 5.11 y se usó el sistema obtenido de la iteración 11 de la Figura 5.11 para clasificar este conjunto. Esto para observar cómo se están clasificando las imágenes, lo cual se observa en la matriz de

	Boquilla vacía	Problema de recolección	Líneas de pulido	Dispositivo quebrado	Muesca en dispositivo	Fallo por aguja
Boquilla vacía	1302	73	198	7	1	19
Problema de recolección	39	302	25	23	2	3
Líneas de pulido	0	0	0	0	0	0
Dispositivo quebrado	0	0	0	0	0	0
Muesca en dispositivo	0	0	0	0	0	0
Fallo por aguja	0	0	0	0	0	0

Tabla 5.12: Matriz de confusión de la iteración 11 del primer experimento. Elaboración propia.

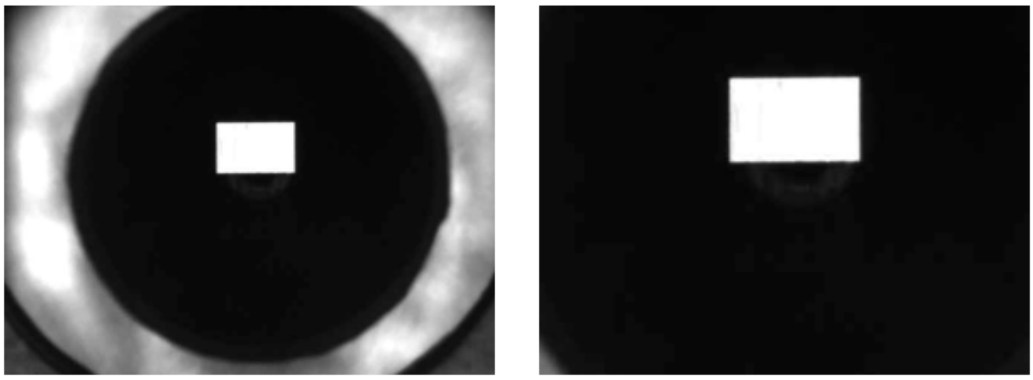
confusión de la Tabla 5.12, la cual confirmó esta hipótesis, ya que se puede observar que todas las imágenes se clasifican entre las categorías de boquilla vacía y problema de recolección.

Con base en estos resultados, se plantearon dos posibles razones por las que el sistema puede comportarse de esta manera. La primera y más posible es un problema en el conjunto de datos de entrenamiento y la segunda es un problema con la arquitectura de la CNN, pero en vista de que en las pruebas preliminares no hubo problema, se examinó el conjunto de datos de entrenamiento en busca del problema.

Después de analizar el conjunto de datos de entrenamiento, se encontraron dos problemas. El primero y más crítico fue que la mayoría de las imágenes presentaban un marco blanco alrededor de la imagen, como se muestra en la Figura 5.12(a), a causa de los múltiples ajustes que se le hacen al sistema de visión de la máquina de DTR. El segundo problema viene de la similitud que hay entre las categorías de muesca en dispositivo, el fallo por aguja y el dispositivo quebrado, esto con base en el estudio antes descrito en la sección 4 y en las pruebas preliminares que al tener estas categorías juntas mejoró considerablemente la precisión.

Para solventar el problema del borde blanco en las imágenes se hizo un recorte de la imagen de 30 %, pero únicamente en los bordes y equitativamente en ancho y largo. Esto para eliminar el marco blanco y preservando toda la información de interés que siempre se encuentra en la parte central de la imagen, esta parte muestra el dispositivo y la boquilla. Es importante mencionar que el recorte se efectúa de manera que se guardan las dimensiones iniciales de las imágenes. Esto se muestra en la Figura 5.12 con base en el código 5.7 con la ayuda de la biblioteca de 'openCV' de Python [5].

```
1 ##=============================================================
2 ## Recortar la imagen para quitar borde blanco
3 ## Parametros: (Direccion de la imagen)
4 ## Return (Imagen sin borde)
5 ##=============================================================
```



(a) Original (b) Recorte de bordes

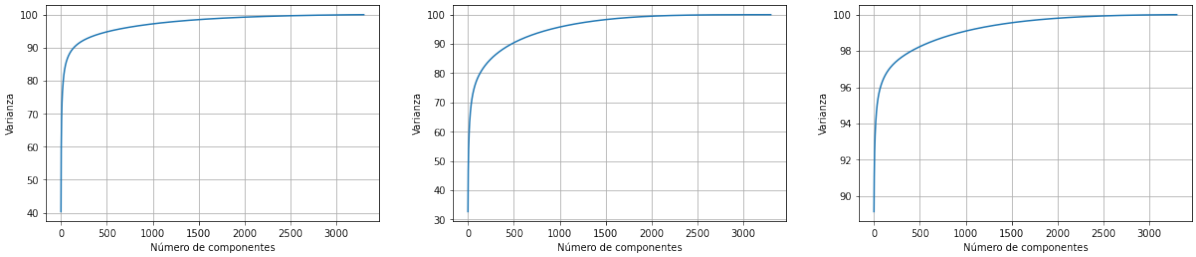
Figura 5.12: Representación de recorte de los bordes de la imagen. Elaboración propia.

```
import cv2
def Canny(path):
    #Abriendo imagen
  img=cv2.imread(path, cv2.IMREAD_GRAYSCALE)
  #Estandarizando tama o
  img=cv2.resize(img, (66,50), interpolation=cv2.INTER_AREA)
  #Recortando imagen
  img=img[10:40, 13:53]
  #Volviendo a tama o original
  img=cv2.resize(img, (66,50), interpolation=cv2.INTER_AREA)
  return img
```

Código 5.7: Código para recortar borde de la imagen.

Para el problema de las tres categorías similares, se consultó al Departamento de Proceso y disposición para determinar la viabilidad de unir estas categorías en la categoría de 'Dispositivo quebrado'. Al no ser categorías críticas para la disposición es posible hacerlo, además de que se estaría cumpliendo con las métricas involucradas. Es importante mencionar que el sistema se ve afectado en relación con la cantidad de modos de fallo que se le pueden reportar al Departamento de Proceso, pero como se estaría reportando el modo de fallo crítico este cambio se validó por parte de todos los involucrados.

Una vez solventados estos problemas se realizó el mismo procedimiento que se detalló, el único punto que cambia a causa de estos cambios es la proporción de varianza explicada acumulada del conjunto de datos de entrenamiento por el recorte que se llevó a cabo en la imagen. Se hizo el análisis de la proporción de varianza explicada acumulada de nuevo y como se muestra



(a) Ecualización (b) Sobel con binarización adaptativa (c) Detector de bordes Canny

Figura 5.13: Proporción de varianza explicada acumulada vs cantidad de componentes al aplicar PCA al conjunto de entrenamiento con los bordes recortados y con cada uno de los preprocesamientos.

en la Figura 5.13, se cumple que al usar 500 componentes por medio de PCA se conserva más del 90 % de las características de las imágenes para los tres algoritmos de preprocesamientos.

Con base en este resultado se realiza el mismo experimento que se planteó en la Tabla 5.10. En esta segunda ejecución del experimento se observó una mejoría muy significativa en el comportamiento de las redes entrenadas. Los resultados de este experimento, mostrados en la Figura 5.14, muestran el análisis estadístico hecho por el *software* 'JMP', que es un *software* externo que cuenta la empresa para análisis estadísticos. En estos resultados se muestra la mejor configuración de variables que corresponde a la mostrada en la Tabla 5.13. Es importante mencionar que las variables de la arquitectura es lo que menos influyó en el desempeño del entrenamiento de las redes.

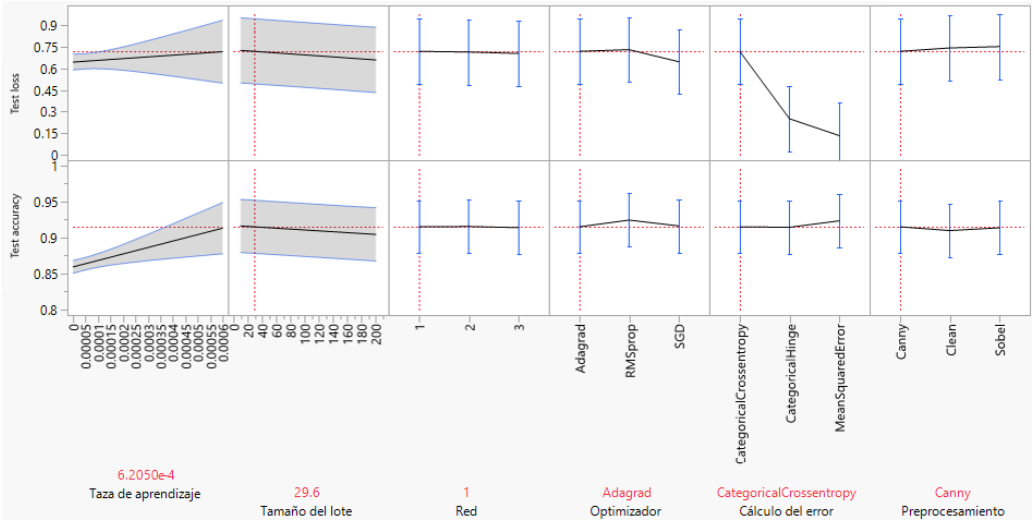
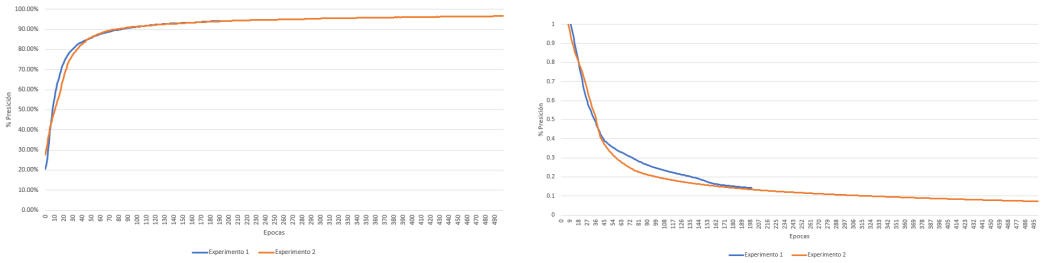


Figura 5.14: Resultados de la segunda ejecución del experimento factorial completo. Elaboración propia.

Variable	Resultado
Arquitectura de la red	Arquitectura 1
Tasa de aprendizaje	0.000001
Tamaño del lote	10
Optimizador	RMSprop
Cálculo del error	MeanSquaredError
Función de activación	Relu
Preprocesado	Canny

Tabla 5.13: Mejor configuración de variables resultado del experimento de factorial completo. Elaboración propia



(a) Gráfica de precisión (b) Gráfica de error

Figura 5.15: Gráficas de precisión y error de las redes con la mejor configuración de variables obtenida con 200 y 500 generaciones.

Una vez establecida la mejor configuración de variables, se evalúa al entrenar la red con un valor mayor de generaciones (200 y 500) y así examinar con detalle su comportamiento. Esto se muestra en la Figura 5.15, donde se observa el entrenamiento de la red, el 'Experimento 1' es a 200 generaciones y 'Experimento 2' es a 500 generaciones. Con base en esta gráfica, se determina que el desempeño de la red a 500 generaciones es más estable y se obtiene mejor precisión final, con un valor de **93.13 % de precisión y 0.17 % de error**.

Finalmente, una vez elegida la arquitectura, los hiperparámetros y el preprocesamiento que tiene, se tiene el sistema final que solventa el subproblema que se planteó. En la Figura 5.16 se muestra la estructura del sistema final.

5.3.4. Documentar fallos

Establecido el sistema de clasificación y procesado, se debe establecer la estructura para documentar los fallos clasificados. Para esto se propuso establecer una base de datos donde se pueda almacenar la información de la clasificación por cada uno de los rollos y que pueda poner

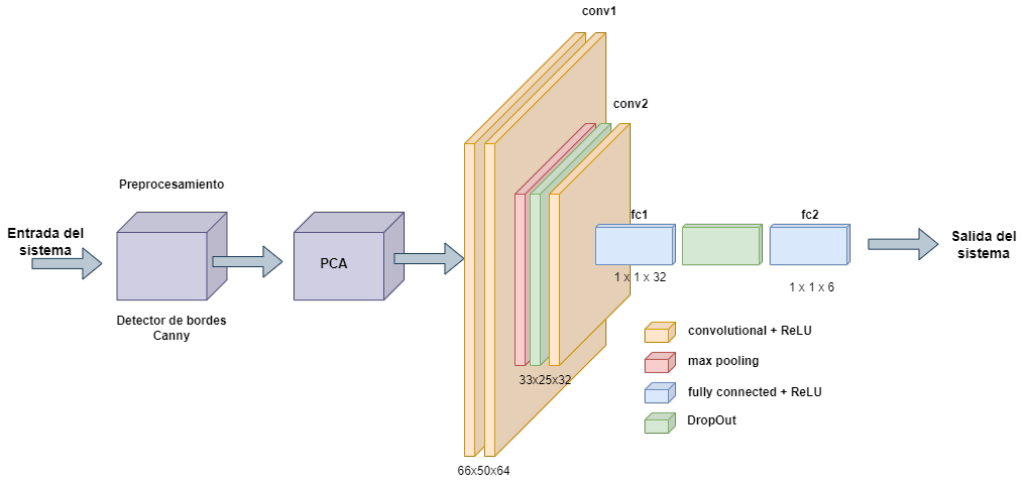


Figura 5.16: Estructura final del sistema de clasificación. Elaboración propia.

Categoría	Umbral
Boquilla vacía	0.1 %
Problema de recolección	0.1 %
Líneas de pulido	0.15 %
Quebradura en dispositivo	3 unidades

Tabla 5.14: Definición de umbrales para alertas del sistema de clasificación. Elaboración propia

a disposición dicha información con un reporte que pueda visualizar fácilmente todo el personal pertinente. Además, este sistema debe poder alertar al departamento de proceso de la empresa en caso de que una de las categorías de fallos supere un umbral porcentual (con respecto al total de piezas del rollo), ha excepción de la categoría de 'Quebradura en dispositivo' que, según lo descrito en 4, cualquier rollo que supere 3 unidades con quebradura representa un problema de calidad. Estos umbrales se muestran en la Tabla 5.14 y son definidos a partir del estudio mecánico hecho (4) y en base al criterio tanto del departamento de disposición y proceso.

Para esta esto, se hizo una investigación interna y externa, según la metodología que se planteó. En esta se determinaron conceptos posibles para la solución del problema.

Para este caso se debe establecer bajo que concepto de bases de datos se va a desarrollar este sistema. Según lo investigado se encontró los conceptos más adecuados para desarrollar el sistema, según la naturaleza del mismos, es mediante sistemas de gestión de servidores basados en un Windows SQL, IBM DB2 o Oracle DataBase [10] [17] [2].

Estos conceptos son muy similares al evaluarlos en los criterios establecidos para este

problema en la figura 5.4, por lo que se procede a compararlos con respecto a su rendimiento y costos. Según [2] y [32], Oracle y IMB DB2 han demostraron significativas fortalezas contra los servidores de Windows SQL, en lo que respecta a mantenimiento, en la facilidad de uso, facilidad de gestión y el diagnóstico [17] [2].

También, según el estudio [32], Oracle es 88 % más eficiente y 93 % menos complejo en cuanto a diagnóstico que IBM DB2, teniendo fortalezas en el área de diagnóstico, ajuste de rendimiento y copia de seguridad y recuperación, lo cual se traduce en un ahorro de costos de aproximadamente 93 %.

En base a estos datos, se puede concluir que la opción óptima es utilizar una base de datos basada en Oracle. Pero en la investigación interna se encontró que en la empresa se tiene implementados servidores basados en Windows SQL, y se tiene como preferencia seguirlo utilizando. Con esto, se establece que la mejor opción para solventar este subproblema es utilizar un servidor Windows SQL, a causa de que usar Oracle se subiría el costo del proyecto al tener que pagar la licencia respectiva e integrar un nuevo sistema a la empresa. Con esto se procede a utilizar el concepto de un sevidor de Windows SQL para la base de datos necesaria.

Una vez establecido el concepto a desarrollar se procede crear una base de datos basado en Windows SQL denominda 'VCTQ_Report', esta se creo en un servidor, el cual fue brindado por la empresa, con la denominación 'SJO2SQL10'. Para crear esta base de datos es necesario considerar la cantidad de registros necesarios necesita, aproximadamente, dicha base de datos, para esto definió las columnas que tendría la base de datos creada. Las columnas y el tipo de formato de la variable que guarda (en caso de caracteres, se especifica su longitud entre paréntesis) se muestran en la Tabla 5.15.

Una vez definido las columnas de la base de datos, se obtiene que por cada rollo clasificado se necesita 10 registro. Para calcular la cantidad de registros necesarios por día se investigó la cantidad de rollos procesados en el mes de diciembre, la cual fue un total de 11632 rollos con un promedio de 361 rollos por día. Con esto se puede estimar que se necesitan aproximadamente 3610 registros por día aproximadamente.

Con la base de datos creada se procede a diseñar un código que pueda enviar datos a la

Columna	Variable
Rollo	Caracter (10)
Línea	Caracter (10)
Producto	Caracter (10)
Fecha de disposición	Fecha-Hora o DateTime
Cantidad clasificada	Entero o int
Líneas de pulido	Entero o int
Dispositivo Quebrado	Entero o int
Boquilla vacía	Entero o int
Fallo de recolección	Entero o int
Fecha de clasificación	Fecha-Hora o DateTime

Tabla 5.15: Definición de columnas establecidas para la base de datos. Elaboración propia

misma. Para esta se diseño un código con la ayuda de la biblioteca 'pyodbc' de Python, con la cual se estableció una comunicación bajo el protocolo ODBC mediante el Código 5.8.

```
import pyodbc
## ========================================================================
## Establecer conexion
## ========================================================================
server='SJO2SQL10'
bd='VCTQ_Report'
user='sjo_usrsql_vctq'
password='aWLQVj9UfxaWUA&a'
conexion=pyodbc.connect('DRIVER={SQL server}; SERVER='+server+'; DATABASE='+bd+'; UID='+user+'
    ; PWD='+password)
cursor=conexion.cursor()
## ========================================================================
## Finaliza conexion
## ========================================================================
cursor.close()
conexion.close()
```

Código 5.8: Código para establecer conexión con base de datos.

Posteriormente, se requiere diseñar un código que pueda enviar los datos respectivos a la base de datos, el cual establece utilizando la misma biblioteca 'pyodbc'. Se diseña una función que pueda ejecutar este envío de datos en base a la información que tiene como parámetros, tal y como se muestra en el Código 5.9.

```
import pyodbc
def EnviarSQL(Rollo,Linea,Producto,Fecha,Qty,BG,Broken,Empty,Recolection,CFecha):
```

```
## ============================================================
## Establecer conexion
## ============================================================
    server='SJO2SQL10'
    bd='VCTQ_Report'
    user='sjo_usrsql_vctq'
    password='aWLQVj9UfxaWUA&a'
    conexion=pyodbc.connect('DRIVER={SQL server}; SERVER='+server+'; DATABASE='+bd+'; UID='+
    user+'; PWD='+password)
    cursor=conexion.cursor()
## ============================================================
## Enviar Datos
## ============================================================
    ## Crear consulta
    consulta="INSERT INTO [VCTQ_Report].[dbo].[VCTQ_Report]([Reel],[Line],[Product],[Date Reel
     Disposition],[Clasification Qty],[BG Lines],[Broken Die],[Empty Nozzel],[Recolection
    Issue],[Clasification Date]) VALUES(?,?,?,?,?,?,?,?,?,?)"
    ## Enviar datos en par metros
    cursor.execute(consulta,Rollo,Linea,Producto,Fecha,Qty,BG,Broken,Empty,Recolection,CFecha)
    cursor.commit()
## ============================================================
## Finalizar conexi n
## ============================================================
    cursor.close()
    conexion.close()
```

Código 5.9: Código para enviar datos a la base de datos SQL.

Paralelamente a este envió de datos se debe evaluar si los resultados de la clasificación superó alguno de los umbrales establecidos en 5.14, en dicho caso es necesario informar al departamento de proceso para que hagan las acciones correctivas necesarias. Para esto esto se hizo un algoritmo que envié un correo en caso de esto ocurra, utilizando un protocolo SMTP, con el servidor 'Smtprelay-central.corp.qorvo.com' que cuenta la empresa para este fin. Este correo se envía al departamento de proceso y disposición con la información del rollo y el umbral que se superó. Esto se muestra en la función elaborada en el Código 5.10.

```
## Importando bibliotecas
from email.mime.multipart import MIMEMultipart
from email.mime.text import MIMEText
import smtplib
def EnviarCorreo(mensaje):
    ## Creando instancia de objeto de mensaje
    msg = MIMEMultipart()
```

```
    ## Datos para envi
    msg['From'] = "Smtprelay-central.corp.qorvo.com"
    msg['To'] = "ProcesoDTR@Qorvo.com;Provisions@qorvo.com"
    msg['Subject'] = "Alerta Por Umbral en DTR"
    ## Agregar a cuerpo de mensaje
    msg.attach(MIMEText(mensaje, 'plain'))
    ## Abriendo servidor SMTP
    server = smtplib.SMTP('Smtprelay-central.corp.qorvo.com')
    # Enviando via el servidor
    server.sendmail(msg['From'], msg['To'], msg.as_string())
    ## Cerrando servidor
    server.quit()
    print ("Mensaje enviado con exito a: %s" % (msg['To']))
```

Código 5.10: Código para enviar correos mediante servidor SMTP de la empresa.

Por último, es necesario crear sistema de reporteo que facilite a las personas pertinente acceder a la clasificación de cada uno de los rollos y poder visualizar a la cantidad de fallos por categoría de cada uno de ellos.

Para desarrollar este reporte se utilizó un software externo brindado por la empresa para generar reportes de esta naturaleza, este software es *TIBCO Sportfire*, con el cual se establece una conexión ODBC con la base de datos antes descrita; esto se logra con el Código 5.11.

```
import pyodbc
def EnviarSQL(Rollo,Linea,Producto,Fecha,Qty,BG,Broken,Empty,Recolection,CFecha):
## ================================================================
## Establecer conexion
## ================================================================
    server='SJO2SQL10'
    bd='VCTQ_Report'
    user='sjo_usrsql_vctq'
    password='aWLQVj9UfxaWUA&a'
    conexion=pyodbc.connect('DRIVER={SQL server}; SERVER='+server+'; DATABASE='+bd+'; UID='+
    user+'; PWD='+password)
    cursor=conexion.cursor()
## ================================================================
## Enviar Datos
## ================================================================
    ## Crear consulta
    consulta="INSERT INTO [VCTQ_Report].[dbo].[VCTQ_Report]([Reel],[Line],[Product],[Date Reel
     Disposition],[Clasification Qty],[BG Lines],[Broken Die],[Empty Nozzel],[Recolection
    Issue],[Clasification Date]) VALUES(?,?,?,?,?,?,?,?,?,?)"
    ## Enviar datos en par metros
```

```
18      cursor.execute(consulta,Rollo,Linea,Producto,Fecha,Qty,BG,Broken,Empty,Recolection,CFecha)
19      cursor.commit()
20  ##=============================================================
21  ## Finalizar conexi n
22  ##=============================================================
23      cursor.close()
24      conexion.close()
```

Código 5.11: Código para consulta de la base de datos en el servidor de SQL.

Una vez establecida dicha conexión, se procede a diseñar la parte gráfica del reporte. Esta se establece mediante la interfaz de diseño gráfico del mismo software *TIBCO Sportfire* y se obtiene el reporte que se muestra en la Figura 5.17. Dicho reporte puede ser consulta por cualquier colaborador de la empresa mediante el enlace: `https://dfwspotfire.corp.qorvo.com/spotfire/wp/OpenAnalysis?file=b0c72bce-8880-40b9-963b-8415bd1ea53e`. Además, este reporte cuenta con las funciones de descargar la información de la tabla de datos en formato 'csv', visualizar los datos con filtros y un pareto de incidencia de fallos por categoría, modelo y línea de producción.

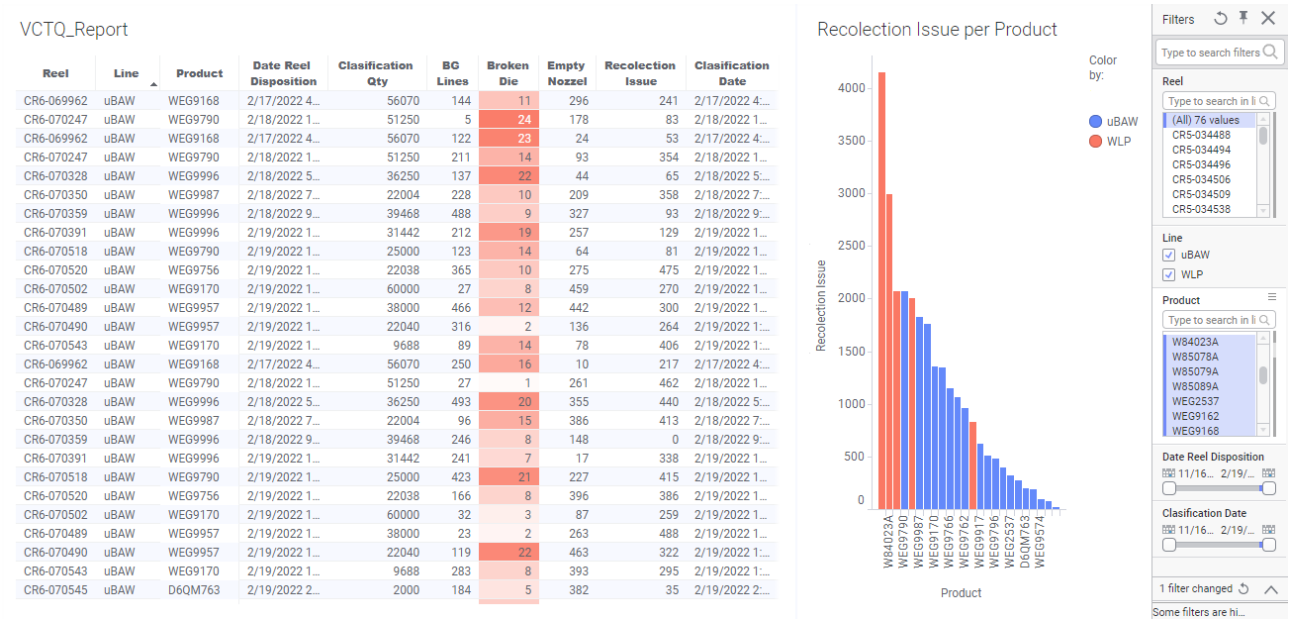


Figura 5.17: Visualización del reporte generado para ver resultados de la clasificación. Elaboración propia.

5.3.5. Reflexión

Siguiendo la metodología, una vez terminada esta etapa del diseño se realiza una reflexión para comprobar la correcta utilización del enfoque metodológico.

En esta reflexión se determina que se ha realizado una amplia exploración tanto por medio de la búsqueda interna como externa, sin embargo, dada la gran cantidad de posibles soluciones resulta imposible abarcar la totalidad de la literatura relativa a los conceptos. Dicho esto, Los principios que se evaluaron presentaban similitud y se exploró diversas ideas, evitando sesgo de conceptos.

Para la la evaluación de los conceptos se realizó para cada subsistema y se utilizaron todos los criterios establecidos previamente para evaluarlos. Es importante mencionar que para establecer el concepto de referencia se utilizó uno de los sistemas encontrados durante la búsqueda externa o el ya existente en la empresa, según sea el caso.

Según lo analizado se puede establecer la correcta utilización del enfoque metodológico en la etapa del diseño.

5.4. Sistema Final

Una vez diseñado la solución para solventar cada uno de los subproblemas establecidos en la descomposición funcional, se procede a establecer el algoritmo que ejecuta el sistema completo. Dicho algoritmo se presenta mediante el diagrama de flujo de la Figura 5.18 y se presenta su respectivo en el Anexo A.1.2.

Finalmente, la arquitectura del sistema diseñado se presenta en la 5.19.

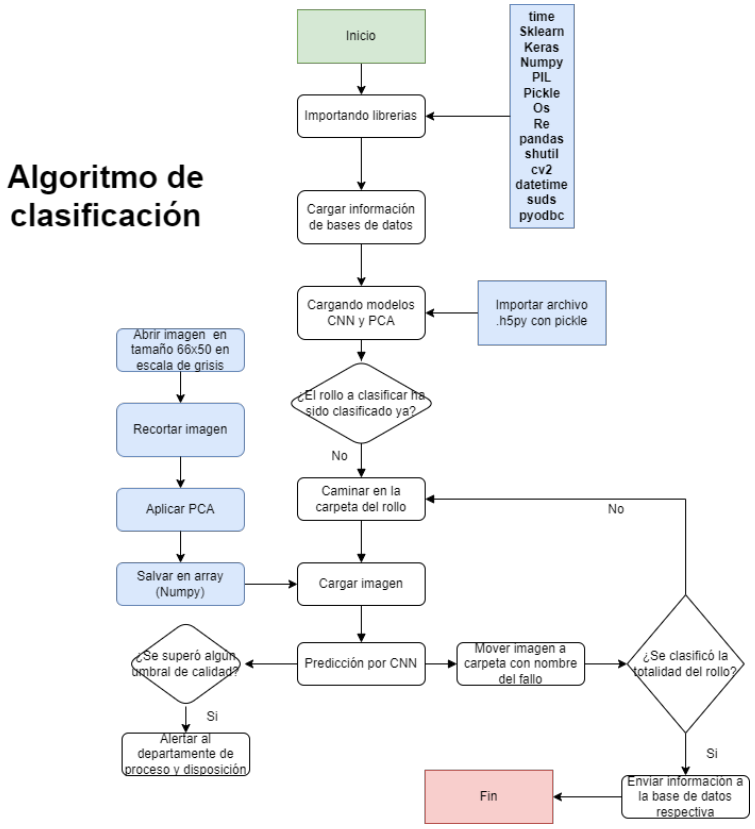


Figura 5.18: Algoritmo final del sistema de clasificación. Elaboración propia.

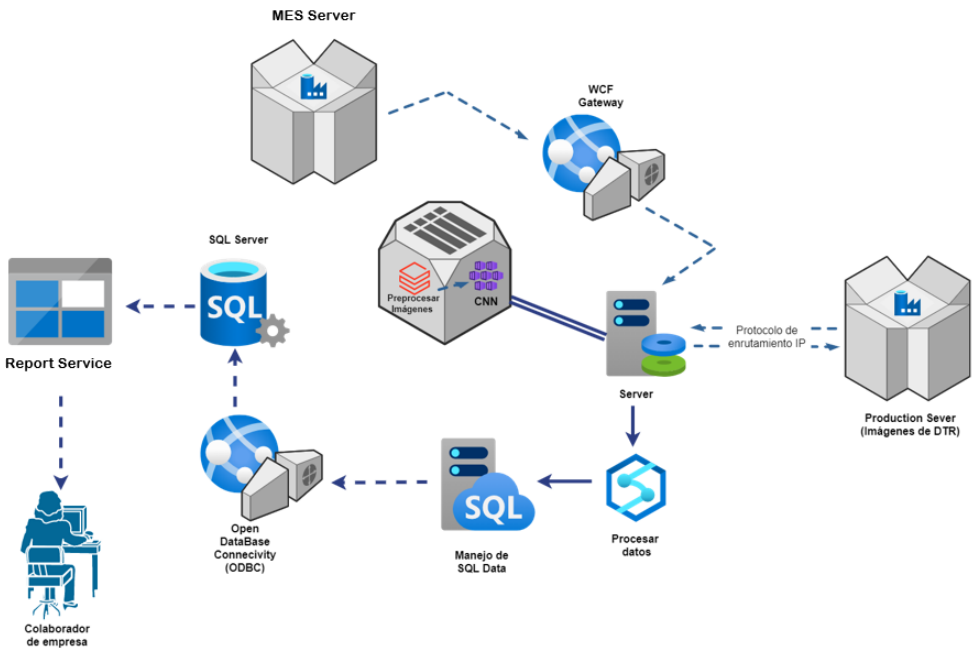


Figura 5.19: Arquitectura final del sistema diseñado. Elaboración propia.

Capítulo 6: Resultados y análisis

Tal y como se plantea en la metodología, una vez planteado, diseñado y programado el sistema descrito se procede a comprobar la funcionalidad de los sistemas desarrollados. Esto se hace mediante pruebas o experimentos junto con el Departamento de Calidad de la empresa para validar el funcionamiento correcto y cumplimiento de las especificaciones del sistema.

También, en esta etapa se desarrolla un análisis económico para determinar el costo del proyecto, su viabilidad, el retorno de inversión y si es factible su implementación en la empresa.

6.1. Pruebas de concepto

Para generar un experimento se debe establecer los , estos se definen en base a las métricas planteadas en la Tabla 5.4. Estas factores son los siguientes:

- Exactitud de detección de fallas críticas.
- Exactitud de clasificación por categoría.
- Velocidad de clasificación.
- Memoria RAM consumida.
- Ancho de banda consumida.
- Grado de intervenciones.
- Grado de información perdida.

Clasificación	Técnico		Operario		Total	
	Iteración 1	Iteración 2	Iteración 1	Iteración 2	Promedio	% DER
Boquilla vacía	3672	3652	3694	3643	3665	0.6187 %
Fallo de recolección	5010	4913	4969	4984	4969	0.8250 %
Líneas de pulido	2933	3055	2955	2992	2984	1.7889 %
Quedradura en dispositivo	469	464	466	465	466	0.4636 %
Tiempo de clasificación (min)	236	253	213	228	233	7.1670 %

Tabla 6.1: Clasificación hecha por el operario y técnico para el DOE diseñando. Elaboración propia.

- Precisión del sistema.
- Grado de oscilación en desempeño según operario.
- Grado de dispersión de la información.

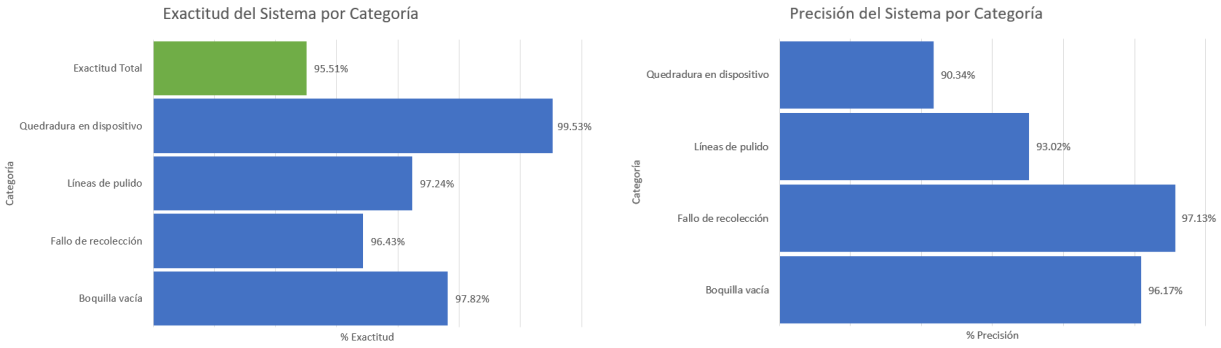
Una vez establecido los factores, se plantea el experimento que pueda evaluarlos. Primeramente, se diseño un experimento donde se evalúan estos factores mediante la clasificación de un conjunto de 30 rollos o lotes de producción (15 de línea A y 15 de la línea B) con el sistema diseñado, y paralelamente un operario y un técnico clasifica el mismo conjunto de rollos dos veces (en dos días distintos), para evaluar repetibilidad. Este conjunto de datos tiene un total de 12094 imágenes provenientes de las dos diferentes líneas pero también de 5 diferentes máquinas de DTR.

Primeramente, se tiene obtiene los resultados de evaluar la clasificación hecha por el operario y técnico de la empresa, la cual se muestra en la Tabla 6.1. En esta tabla se puede observar que existe una variación entre la clasificación entre el operario y el técnico, e incluso varia cuando la misma persona hace la misma clasificación de nuevo, lo cual se evidencia en los porcentajes de desviación estándar relativa (DER), la cual puede llegar hasta un 1.8 % aproximadamente. También es importante mencionar que, en promedio, se duró 3.9 horas realizando cada una de estas clasificaciones.

Una vez obtenido estos resultados, se procede a realizar la misma clasificación con el sistema de clasificación basado en inteligencia artificial diseñado en este proyecto, con el cual se obtuvo los resultados mostrados en la matriz de confusión de la Tabla 6.2. Este experimento demoró un total de 46.32 minutos, lo que nos da como resultado que cada imagen se clasifica en 0.23 segundos aproximadamente, lo que cumple con la especificación definida en la Tabla 5.4 y

	Boquilla vacía	Fallo de recolección	Líneas de pulido	Quedradura en dispositivo
Boquilla vacía	3542	94	47	0
Fallo de recolección	73	4676	63	2
Líneas de pulido	36	173	2865	6
Quedradura en dispositivo	14	26	9	458

Tabla 6.2: Matriz de confusión de los resultados del experimento. Elaboración propia.



(a) Pareto de exactitud (b) Pareto de precisión

Figura 6.1: Pareto de resultados de exactitud y precisión del experimento. Elaboración propia.

muestra una mejoría promedio del **80.12 %** en el tiempo de clasificación con respecto al sistema de clasificación actual.

Obtenida la matriz de confusión, se puede usar para calcular la exactitud total del sistema y la exactitud por categoría, tal y como se muestra en el pareto de la Figura 6.1(a). En este pareto se muestra que la exactitud del sistema total es de **95.51 %** y la exactitud por categoría que no es menor a **96.43 %** en ningún caso. Con estos datos se puede determinar que el sistema cumple con las métricas métricas respectivas, mostradas en la Tabla 5.4. También se observa que, la exactitud de las fallas críticas (quebradura en dispositivo) cuenta con una exactitud de **99.53 %** la cual es muy cercana a la especificación ideal definida de 100 %, y cumple con el valor marginal al ser mayor a 90 %, por lo que se puede concluir que cumple con el requerimiento.

Posterior a esto, se puede calcular la precisión total del sistema y por categoría con la misma matriz de confusión de la Tabla 6.2. Al calcular estos datos se obtiene el pareto de la Figura 6.1(b), donde se puede observar que la precisión del sistema es todas las categorías es mayor a **90.34 %**, lo cual no cumple con el valor ideal de la métrica respectiva en la Tabla 5.4, sin embargo, esta si cumple con el valor marginal. Con esto se puede concluir que cumple con el requerimiento del cliente.

Toma de	Iteración 1		Iteración 2	
	RAM (Mb)	Ancho de banda (Kbps)	RAM (Mb)	Ancho de banda (Kbps)
1	623	328	593	281
2	702	405	684	367
3	656	278	708	452
4	613	356	713	416
5	681	345	697	378

Tabla 6.3: Resultados del experimento de consumo de RAM y ancho de banda. Elaboración propia.

En cuanto a las métricas 8, 9, 10 y 11, al ser un sistema autónomo que no requiere intervenciones fuera de las acciones de inicio y fin, por lo que, cuanto a la métrica 8 y 11, se puede concluir que el sistema cumple con una especificación de grado 1 en ambas métricas, ya que no requiere intervenciones y no va a variar dependiendo de quien lo utilice. Además, al ser un sistema que cuenta un una arquitectura IoT que consulta y guarda datos directamente en bases de datos y proporciona un reporte de los mismos a todos los colaboradores de la empresa, cumple con las especificaciones de grado 1 y 3 de las métricas 9 y 12 respectivamente.

Por ultimó, para calcular la memoria RAM y el ancho de banda consumido por el sistema, se requiere realizar de nuevo el experimento con ligeras variaciones. Para calcular estos datos, se decidió ejecutar la clasificación del conjunto de imágenes anterior, pero esta vez, obteniendo los valores de consumo de RAM y ancho de banda. Como se obtuvo que ese experimento dura aproximadamente 46.32 minutos, se procede a hacer 5 toma de datos cada 10 minutos durante la ejecución del mismo, además se realiza dos veces para examinar repetibilidad. Los resultados de esto se muestra en la Tabla 6.3.

Con dicho experimento se obtiene que, en promedio, el sistema consume 667 Mb de RAM aproximadamente durante su ejecución, lo cual equivale a 0.667 Gb, cumpliendo así con la expecificación. Con respecto al ancho de banda, el sistema consume un promedio de 361Kbps aproximadamente, lo que equivale a 0.361 Mbps, lo cual también cumple con la especificación.

Por último, se realizó una comparación de los datos de los rollos y su clasificación con respecto a los reportes de trazabilidad de la empresa para comprobar que los datos obtenidos y mostrados por el sistema de reporteo generado. Dicha comparación dió como resultado que el 100 % de los datos coinciden correctamente, por lo que se valida el correcto funcionamiento de

la infraestructura IoT diseñada.

Con este experimento, se puede concluir que el sistema diseñado cumple con los respectivos requerimientos planteados por el cliente para este proyecto al no incumplir con los valores de las especificaciones de cada una de las métricas.

6.2. Análisis económico

Como parte de los aspectos a analizar, se debe realizar un análisis económico para determinar el costo del proyecto, su viabilidad, el retorno de inversión y si es factible su implementación en la empresa.

Para analizar las consideraciones económicas del proyecto, primeramente, se calculó la inversión para la realización del proyecto. Estos se dividen en dos tipos de gastos, los materiales y herramientas y los servicios generales. Cada uno de los materiales, con su descripción y precio se muestran en la Tabla 6.4. Es importante mencionar que se toma en cuenta ₡200 000 para imprevistos en cuanto a infraestructura, siendo el valor dado por la empresa como parte del análisis previo del problema. También se considera los gastos con respecto a la remuneración para el diseño del proyecto, para el cual se tomó el salario base dado por la empresa a practicantes (₡482 000 mensuales) durante 26 semanas, lo que equivale a 7 meses.

Después de hacer el cálculo del costo de este proyecto, se puede determinar que los recursos por utilizar son únicamente recursos computacionales, ya que no es necesario nada más para resolver el problema que se planteó; estos recursos se centran en alto nivel de procesamiento computacional y *software*. Sin embargo, después de dialogar con la empresa donde se realiza el proyecto se pudo determinar que cualquier recurso, ya sea de infraestructura o *software*, ya se encuentra disponible en la compañía, por lo que se hizo uso de estos para el diseño del proyecto. Por lo tanto, todos los gastos se resumen en los gastos en servicios generales, los cuales de una u otra forma son parte del desarrollo de este. Esto se puede observar en la Tabla 6.4.

Todos los materiales y herramientas son provistos por el Departamento de Ing. de

Tabla 6.4: Tabla resumen del costo del proyecto. Elaboración propia.

Descripción	**Cantidad**	**Valor estimado por unidad (Colones)**	**Subtotal (Colones)**	**Disponible en la empresa actualmente**
Materiales y Herramientas				
Computadora de alto rendimiento	1	570 000	570 000	SI
Windows Server	1	800 000	800 000	SI
Microsoft 365 Business	1	50 000	50 000	SI
JMP / Software estadístico	1	1 250 000	1 250 000	SI
TIBCO Spotfire	1	800 000	800 000	SI
Servicios Generales				
Remuneración económica para proyecto de graduación.	7	482 400	3 376 800	—
Alimentación	140	2 000	280 000	—
Transporte	140	430	60 200	—
Imprevistos	1	200 000	200 000	—
Total			**₡3 917 000**	

Producto y de Tecnologías de Información según corresponda, todo esto ajustándose a los requerimientos del proyecto y a las restricciones económicas que pueda tener la empresa.

Una vez realizado el cálculo del costo del proyecto, el cual es de ₡3 917 000 como inversión inicial; pero también hay que considerar el mantenimiento del mismo, que para efectos de este proyecto se puede calcular con las licencias de software necesarios, que en este caso es únicamente el software de TIBCO Spotfire. Por lo que, el costo de mantenimiento del sistema se puede calcular en ₡800 000 anuales.

Gracias a esta estimación realizada se puede determinar que este proyecto es viable, ya que no solo soluciona el problema de la empresa de un inminente cuello de botella en el proceso de disposiciones, sino que al estimar que esta tarea se lleva a cabo actualmente por un operario 12 h al día durante 7 días a la semana con la ayuda de un técnico por 5 h al día durante 5 días a la semana se puede calcular, de manera conservadora, un ahorro de **₡679 810,92 al mes** según lo valores de salario mínimos dados el Ministerio de Trabajo (para un operario de computo y un técnico diplomado), lo cual es un total de ₡8 157 731 al año. Esto da un estimado de la recuperación de la inversión de**9 meses** aproximadamente, sin mencionar el aporte que se les hará a los demás departamentos de la empresa con los datos recolectados y las futuras aplicaciones de estos y del sistema en general, en otras áreas.

ROI (Basado en valor actual neto, ₡)		**2021**	**2022**	**Total**	**VA**
Beneficios	**Tipo de ahorro**	**₡0**	**₡8 157 731**	**₡8 157 731**	**₡7 093 679**
Horas de trabajo ahorradas	Horas		₡8 157 731	**₡8 157 731**	**₡7 093 679**
Costo del Proyecto	**Tipo de inversión**	**₡4 717 000**	**₡800 000**	**₡5 517 000**	**₡5 412 652**
Inversión Inicial y costo de tiempo de diseño	Capital inicial	₡3 917 000		**₡3 917 000**	**₡3 917 000**
Mantenimiento anual	Capital inicial	₡800 000	₡800 000	**₡1 600 000**	**₡695 652**
Impuesto		₡0	₡0	₡0	₡0
Cambio de capital de trabajo				₡0	₡0
Flujo de Caja Neto		-(₡4 717 000)	₡7 357 731	₡2 640 731	₡1 681 027
Valor Actual Neto (₡)					**₡1 681 027**

Tabla 6.5: Estudio económico durante 1 año para el cálculo de retorno y valor actual neto. Elaboración propia.

Beneficio Total	₡8 157 731
Costo Total	₡5 517 000
Retorno Total	₡2 640 731
VAN (Después de impuestos)	₡1 681 027
TIR	56 %
ROI	48 %
Recuperación de la Inversión (meses)	9

Tabla 6.6: Indicadores y cálculos de retorno total sobre la inversión. Elaboración propia.

Como el desarrollo de este proyecto va enfocado al ahorro de costos en una operación de la planta de producción, se procede a hacer un análisis más profundo de las ganancias a corto plazo con indicadores como el valor capital neto (VAN), el retorno sobre inversión (ROI) y la tasa interna de retorno (TIR), esto con los datos calculados anteriormente. El estudio se realizó en una proyección a 1 año y con una tasa de descuento del 15 %, debido que así el estándar de proyección de la empresa para proyectos de esta naturaleza.

Dicho estudio económico se representa en la Tabla 6.5, el cual muestra el beneficio y el costo total de llegar a implementar el sistema diseñado. En este estudio se obtiene un retorno total de **₡2 640 731** y un VAN positivo, lo cual muestra que el proyecto es viable y el TIR es del 56 % de la inversión total.

Esto datos se resumen en la Tabla 6.6, con la cual se concluye que el proyecto es viable y genera altas ganancias en el proceso, sin mencionar que esta es una proyección únicamente a un año, por lo que al considerar el beneficio a largo plazo termina por ser varias veces mayor.

Capítulo 7: Conclusiones y recomendaciones

7.1. Conclusiones

- Se estableció un estudio mecánico del proceso de recolección y empaquetado de chips que permitió definir 9 modos de fallo distintos, su procedencia y sus respectivas acciones correctivas, el cual se muestra en la sección 4.
- Se determinó, en base a investigaciones y análisis que, para la naturaleza y características de este problema, el mejor paradigma para solventar una arquitectura de IoT para la lectura, almacenamiento y envió de datos a las diferentes servidores y bases de datos es mediante protocolos de *Windows Communication Foundation Gateway* o compuertas WCF, protocolos SMB y Bases de datos basadas en Windows SQL.
- Se determinó, en base a investigación y pruebas experimentales, que un sistema basado en inteligencia artificial, con los paradigmas de análisis de componentes principales (PCA) y redes neuronales convolucionales (CNN), es el más adecuado para solventar el problema de clasificación de fallos mecánicos.
- Se concluye que, mediante pruebas experimentales, la forma óptima de procesar las imágenes antes de utilizar la red neuronal convolucional es mediante un análisis de componentes principales con 500 componentes, recortando los bordes de la imagen y con un algoritmo Canny. Obteniendo con esto un sistema que puede clasificar las imágenes

en los 4 modos de fallo críticos del sistema con una exactitud total de **95.51 %** y de un **99.58 %** en las fallas críticas, el sistema se muestra en las Figuras 5.16 y 5.19.

- Se definió los óptimos hiperparámetros para el entrenamiento de la red neuronal convolucional mediante los resultados experimentales de un factorial completo entre diferentes hiperparámetros, dichos hiperparámetros se muestra en la Tabla 5.13. Obteniendo con esto una precisión de **93.13 %** de entrenamiento con un error de **0.17 %**.

- Se logra una mejoría promedio de **80.12 %** en la velocidad de clasificación, en comparación con el sistema actual, con el diseño un sistema de evaluación automática del estado de los rollos y de la máquina del proceso de recolección y empaquetado de chips que permite identificar que rollos son necesarios de clasificar, genera la clasificación y detecta si dicha máquina esta presentando un comportamiento anormal con respecto a uno de los modos de fallo definidos, evaluando las imágenes a una velocidad de **0.23 segundos** por imagen, tal y como se muestra en el Código A.1.2.

- Se diseño una arquitectura IoT que permite enviar y almacenar información del sistema de clasificación diseñado, que permite, mediante un reporte digital, consultar, visualizar y obtener la información generada a cualquier colaborador de la empresa, el cual se muestra en la Figura 5.17.

- Se diseño un experimento que permitió estudiar la funcionalidad del sistema diseñado con respecto al sistema actual y el consumo computacional del mismo (en términos de RAM y ancho de banda), lo que permitió validar su correcto funcionamiento, su exactitud mínima de **96.43 %** con una precisión mayor a **90.34 %** en todas sus categorías y un consumo promedio de 667Mb de RAM y 361Kbps para su funcionamiento.

- Al analizar las consideraciones económicas del proyecto se obtuvo que se tendría un retorno total de **₡2 640 731** en un plazo de un año, lo que da como resultado que se recupera la inversión en **9 meses** con una tasa de recuperación de la inversión del **48 %**, lo cual valida la viabilidad económica del proyecto.

7.2. Recomendaciones

- Se recomienda estudiar la posibilidad de poder aumentar la calidad de las imágenes al solicitar al proveedor de la máquina acceso al sistema de visión respectivo y poder rediseñar el sistema para poder clasificar las imágenes directamente al obtener la imagen y sin ningún procesamiento o reducción de calidad posterior. Con esto se podría aumentar la cantidad de categorías que se podrían clasificar y su exactitud.

- Se propone considerar la inclusión de la información del rollo a la hora de diseñar el sistema de clasificación, ya que al agregar información con el tipo de material que se procesa, su tamaño y máquina podría llegar a aumentar la robustez del sistema para poder clasificar mayor número de categorías y con mayor exactitud.

- Se recomienda generar estudios para poder determinar si este sistema se puede llegar a implementar en otras estaciones del proceso o incluso con otras líneas, lo que podría llegar a generar un gran beneficio a la empresa, esto en vista de que el automatizar únicamente este proceso genera un gran ahorro de costos al proceso.

Referencias Bibliográficas

[1] N. R. Andrés Luna, *Detección y conteo de personas en espacios cerrados utilizando estrategias basadas en visión artificial*, 2017. dirección: http://hdl.handle.net/10554/38771.

[2] V. P. K. Anne, V. S. Ponnam y G. Praveen, "A significant approach for cloud database using shared-disk architecture," en *2012 CSI Sixth International Conference on Software Engineering (CONSEG)*, 2012, págs. 1-4. DOI: 10.1109/CONSEG.2012.6349478.

[3] W. O. de Araujo y C. J. Coelho, "Análise de componentes principais (PCA)," *University Center of Anápolis, Annapolis*, 2009.

[4] J. R. Bergado, C. Persello y C. Gevaert, "A deep learning approach to the classification of sub-decimetre resolution aerial images," en *2016 IEEE International Geoscience and Remote Sensing Symposium (IGARSS)*, IEEE, jul. de 2016. DOI: 10.1109/igarss.2016.7729387. dirección: https://doi.org/10.1109/igarss.2016.7729387.

[5] G. Bradski, "The OpenCV Library," *Dr. Dobb's Journal of Software Tools*, 2000.

[6] G. Chavez, M. Ortiz, G. Rojas, J. Gómez y J. Moncrieffe, "Costa Rica Assembly and Test Process Flows," Qorvo, Inc., inf. téc. 104789 <Rev. G>, ene. de 2021, [Accedido el 18-10-2021].

[7] Cohu, "Ismeca NY 32 Documentation," Cohu Company, inf. téc., sep. de 2015, [Accedido el 22-10-2021].

[8] A. P. Engelbrecht, *Computational Intelligence: An Introduction*, en, 2.ª ed. Nashville, TN: John Wiley & Sons, oct. de 2007, ISBN: 9780470512500. dirección: https://books.google.co.cr/books?id=IZosIcgJMjUC.

[9] D. Garcia-Alvarez y M. Fuente, "Estudio comparativo de técnicas de detección de fallos basadas en el Análisis de Componentes Principales (PCA)," *Revista Iberoamericana de Automática e Informática Industrial RIAI*, vol. 8, n.º 3, págs. 182-195, 2011.

[10] D. Gawlick, D. Lenkov, A. Yalamanchi y L. Chernobrod, "Applications for expression data in relational database systems," en *Proceedings. 20th International Conference on Data Engineering*, 2004, págs. 609-620. DOI: 10.1109/ICDE.2004.1320031.

[11] M. Hernández, "Criterios de aceptación y rechazo para el proceso de DTR," Qorvo, Inc., inf. téc. SPE-002017 <Rev A>, ago. de 2020, [Accedido el 19-10-2021].

[12] Z. Huili y L. Guo, "Study on resources sharing between different network flats," en *2009 2nd International Conference on Power Electronics and Intelligent Transportation System (PEITS)*, vol. 3, 2009, págs. 181-183. DOI: 10.1109/PEITS.2009.5406829.

[13] Z. Huili, R. Wenxia, L. Wenzhe, G. Guanwang e Y. Yongzhe, "Study on Access Technology of Storage Data in Heterogeneous Network," en *2007 8th International Conference on Electronic Measurement and Instruments*, 2007, págs. 3-769-3-771. DOI: 10.1109/ICEMI.2007.4351030.

[14] P. K. Illa y N. Padhi, "Practical Guide to Smart Factory Transition Using IoT, Big Data and Edge Analytics," *IEEE Access*, vol. 6, págs. 55 162-55 170, 2018. DOI: 10.1109/ACCESS.2018.2872799.

[15] M. T. Islam, B. N. Karim Siddique, S. Rahman y T. Jabid, "Food Image Classification with Convolutional Neural Network," en *2018 International Conference on Intelligent Informatics and Biomedical Sciences (ICIIBMS)*, vol. 3, 2018, págs. 257-262. DOI: 10.1109/ICIIBMS.2018.8550005.

[16] Martín Abadi, Ashish Agarwal, Paul Barham y col., *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*, Software available from tensorflow.org, 2015. dirección: https://www.tensorflow.org/.

[17] A. Mateen, B. Raza, M. Sher, M. M. Awais y T. Hussain, "Evolution of autonomic Database Management Systems," en *2010 The 2nd International Conference on Computer and Automation Engineering (ICCAE)*, vol. 1, 2010, págs. 33-37. DOI: 10.1109/ICCAE.2010.5452007.

[18] J. Migliaccio, "GTS MES Interface Specification," Qorvo, Inc., inf. téc. SPE-002107 <Rev H>, ago. de 2021, [Accedido el 17-01-2022].

[19] E. G. Moreno, *Automatización de procesos industriales*. Alfaomega Valencia, 2001.

[20] K. O'Shea y R. Nash, "An introduction to convolutional neural networks," *arXiv preprint arXiv:1511.08458*, 2015.

[21] Oracle. "¿Qué es una base de datos?" (2021), dirección: `https://www.oracle.com/mx/database/what-is-database/`.

[22] O. A. B. Penatti, K. Nogueira y J. A. dos Santos, "Do deep features generalize from everyday objects to remote sensing and aerial scenes domains?" En *2015 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, IEEE, jun. de 2015. DOI: `10.1109/cvprw.2015.7301382`. dirección: `https://doi.org/10.1109/cvprw.2015.7301382`.

[23] Qorvo, Inc. "Qorvo: All Around You," [Accedido el 02-10-2021]. (2014), dirección: `https://www.qorvocr.com/`.

[24] A. Ramanath, S. Muthusrinivasan, Y. Xie, S. Shekhar y B. Ramachandra, "NDVI Versus CNN Features in Deep Learning for Land Cover Clasification of Aerial Images," en *IGARSS 2019 - 2019 IEEE International Geoscience and Remote Sensing Symposium*, 2019, págs. 6483-6486. DOI: `10.1109/IGARSS.2019.8900165`.

[25] A. Romero, C. Gatta y G. Camps-Valls, "Unsupervised Deep Feature Extraction for Remote Sensing Image Classification," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 54, n.º 3, págs. 1349-1362, mar. de 2016. DOI: `10.1109/tgrs.2015.2478379`. dirección: `https://doi.org/10.1109/tgrs.2015.2478379`.

[26] L. Rouhiainen, "Inteligencia artificial," *Madrid: Alienta Editorial*, 2018.

[27] J. Salazar y S. Silvestre, "Internet de las cosas," *Techpedia. České vysoké učení technické v Praze Fakulta elektrotechnická*, 2016.

[28] C. Sanchez, "Tolerancias DTR," Qorvo, Inc., inf. téc. SPE-002419 <Rev G>, sep. de 2021, [Accedido el 22-10-2021].

[29] M. Shaha y M. Pawar, "Transfer Learning for Image Classification," en *2018 Second International Conference on Electronics, Communication and Aerospace Technology (ICECA)*, 2018, págs. 656-660. DOI: 10.1109/ICECA.2018.8474802.

[30] I. society. "LA INTERNET DE LAS COSAS." (2015), dirección: https://www.internetsociety.org/wp-content/uploads/2017/09/report-InternetOfThings-20160817-es-1.pdf.

[31] R. Soto, "Disposición de Material en Localización CTQ," Qorvo, Inc., inf. téc. WIS-003866 <Rev A>, oct. de 2019, [Accedido el 22-10-2021].

[32] B. C. Steve Mintz Craiqueg Norris, "Oracle Database 11g vs.IBM DB2® Enterprise 9.5," 2008. dirección: %5Curl%7Bhttp://hosteddocs.ittoolbox.com/em%5C_us%5C_en%5C_wp%5C_11gibm.pdf%7D.

[33] *Suds-py3 Documentation <Release 1.3.1>*, https://suds-py3.readthedocs.io/_/downloads/en/latest/pdf/, [Accedido el 17-01-2022].

[34] TDK, "Method and apparatus for picking up work piece and mounting machine," TDK Corp, inf. téc. US7445688B2, 2004, [Accedido el 22-10-2021].

[35] K. Ulrich, S. Eppinger y R. Alvarez, *Diseño y desarrollo de productos: enfoque multidisciplinario*, ép. Educación / McGraw Hill. McGraw Hill, 2004, ISBN: 9789701047934. dirección: https://books.google.co.cr/books?id=z%5C_5MOgAACAAJ.

[36] L. Villarroel, J. Alvarez y D. Maldonado, "Aplicación del análisis de componentes principales en el desarrollo de productos," *Acta nova*, vol. 2, n.º 3, págs. 399-408, 2003.

[37] E. Winarno, I. Husni Al Amin, H. Februariyanti, P. W. Adi, W. Hadikurniawati y M. T. Anwar, "Attendance System Based on Face Recognition System Using CNN-PCA Method and Real-time Camera," en *2019 International Seminar on Research of Information Technology and Intelligent Systems (ISRITI)*, 2019, págs. 301-304. DOI: 10.1109/ISRITI48646.2019.9034596.

[38] X. X. Zhu, D. Tuia, L. Mou y col., "Deep Learning in Remote Sensing: A Comprehensive Review and List of Resources," *IEEE Geoscience and Remote Sensing Magazine*, vol. 5, n.º 4, págs. 8-36, dic. de 2017. DOI: 10.1109/mgrs.2017.2762307. dirección: https://doi.org/10.1109/mgrs.2017.2762307.

[39] B. M. Zolotová I. y T. Lojka, *Industry IoT Gateway for Cloud Connectivity*. Management Towards Sustainable Growth, 2015, vol. 460, ISBN: 978-3-319-22758-0.

Anexo A: Anexos

A.1. Códigos

En esta sección se anexa los diferentes códigos y bibliotecas que se desarrollaron durante el proyecto que facilitan el entendimiento del sistema diseñando.

A.1.1. Biblioteca desarrollada para crear el sistema de clasificación

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
##
##    VCTQ_Automatization.py
##    Copyright 2021
##    Qorvo, Inc
##            Jeremy Fuentes <Ext. 508677>
##    August 2021

##=============================================================================
## Bibliotecas necesarias
##=============================================================================
def Bibliotecas():
  ## Importando bibliotecas necesarias
   global sys,os,re,cv2,Image,ImageOps,np,time
   global train_test_split,classification_report,to_categorical
   global keras,Sequential,Input,Model,Dense,Dropout,Flatten,Conv2D,MaxPooling2D,LeakyReLU,
     layers,Conv1D, MaxPooling1D
   global pickle,pd,plt,redirect_stdout,SGD,RMSprop,Adagrad,CategoricalCrossentropy,
     MeanSquaredError,CategoricalHinge,compute_class_weight
   global urljoin
   global Reshape
```

```
21   global StandardScaler
22   ## General
23   import sys, os,re
24   from time import time
25   ## Preprocesamiento
26   import cv2
27   import numpy as np
28   ## Conjuntos de entrenamiento
29   from sklearn.model_selection import train_test_split
30   from sklearn.metrics import classification_report
31   from tensorflow.keras.utils import to_categorical
32   from sklearn.preprocessing import StandardScaler
33   ## Creacion de la red
34   import keras
35   from keras.models import Sequential,Input,Model
36   from keras.layers import Dense,Dropout,Flatten
37   from keras.layers import Conv2D,MaxPooling2D
38   from keras.layers.advanced_activations import LeakyReLU
39   from keras import layers
40   from keras.layers import Conv1D, MaxPooling1D
41   ## Entrenamiento
42   import pickle
43   import matplotlib.pyplot as plt
44   from contextlib import redirect_stdout
45   import pandas as pd
46   from tensorflow.keras.optimizers import SGD,RMSprop,Adagrad
47   from keras.losses import CategoricalCrossentropy,MeanSquaredError,CategoricalHinge
48   from sklearn.utils import compute_class_weight
49   from keras.layers import Reshape
50   from tensorflow import keras
51   ## Clasificar
52   from posixpath import join as urljoin
53   print('================================================================================')
54   print('Importacion de bibliotecas completa')
55   print('================================================================================')
56   ## Se importaron correctamente
57   return True
58
59 ##=============================================================
60 ## Definir tiempo de ejecucion
61 ##=============================================================
62 def seg_to_time(seg):
63   days = int(seg // 86400)
64   seg -= 86400*days
65
66   hrs = int(seg // 3600)
```

```
67    seg -= 3600*hrs
68
69    mins = int(seg // 60)
70    seg -= 60*mins
71
72    seg = round(seg,3)
73
74    formato = str(days)+' dias : '+str(hrs)+' hrs : '+str(mins)+' min : '+str(seg)+' seg'
75    return formato
76
77 ##=============================================================
78 ## Preprocesar la imagen mediante Sobel y binarizacion adaptativa
79 ## Parametros: (Direccion de la imagen)
80 ## Return (Imagen binarizada)
81 ##=============================================================
82 def Sobel(img_path):
83   ## Recibe direccion de la imagenes como parametro
84   ## Cargando imagen
85   img=cv2.imread(img_path, cv2.IMREAD_GRAYSCALE)
86   ## Filtro de ruido gaussiano
87   img = cv2.GaussianBlur(img,(3,3),0)
88   ## Aplicando convoluccion (sobel)
89   sobelx = cv2.Sobel(img,cv2.CV_64F,1,0,ksize=5)  # En X
90   sobely = cv2.Sobel(img,cv2.CV_64F,0,1,ksize=5)  # En Y
91   img=sobelx+sobely # AND para obtener sobel en X y Y
92   ## Normalizando datos entre 0 y 255 para poder binarizar
93   img=cv2.normalize(img, None, alpha = 0, beta = 255, norm_type = cv2.NORM_MINMAX, dtype = cv2
       .CV_16UC1)
94   ## Binarizando usando Otsu (binarizado adaptativo)
95   ## Se tiene binarizado en valores de 0 y 255
96   (T, img) = cv2.threshold(img, 0, 255, cv2.THRESH_BINARY_INV | cv2.THRESH_OTSU)
97   return img
98
99 ##=============================================================
100 ## Preprocesar la imagen unicamente con filtro de ruido y ecualizacion
101 ## Parametros: (Direccion de la imagen)
102 ## Return (Imagen binarizada)
103 ##=============================================================
104 def Clean(img_path):
105   ## Recibe direccion de la imagenes como parametro
106   ## Cargando imagen
107   img=cv2.imread(img_path, cv2.IMREAD_GRAYSCALE)
108   ## Filtro de ruido gaussiano
109   img = cv2.GaussianBlur(img,(3,3),0)
110   ## Ecualizar
111   clahe = cv2.createCLAHE()
```

```
112   equalized = clahe.apply(img)
113   #(T, img) = cv2.threshold(equalized, 0, 255, cv2.THRESH_BINARY_INV | cv2.THRESH_OTSU)
114   return equalized
115
116 ##=====================================================================
117 ## Preprocesar la imagen para solo tener contornos mediante Canny
118 ## Parametros: (Direccion de la imagen)
119 ## Return (Imagen de contornos)
120 ##=====================================================================
121 def Canny(path):
122   img=cv2.imread(path, cv2.IMREAD_GRAYSCALE)
123   gray = cv2.GaussianBlur(img,(3,3),0)
124   # Encuentra el gradiente en la direcci n X
125   grad_x = cv2.Sobel(gray, cv2.CV_16SC1, 1, 0)
126   # Encuentra el gradiente en la direcci n y
127   grad_y = cv2.Sobel(gray, cv2.CV_16SC1, 0, 1)
128   # Convertir el valor del gradiente a 8 bits
129   x_grad = cv2.convertScaleAbs(grad_x)
130   y_grad = cv2.convertScaleAbs(grad_y)
131   # Combina dos gradientes
132   src1 = cv2.addWeighted(x_grad, 0.5, y_grad, 0.5, 0)
133   # Combine gradientes con algoritmo canny, donde 50 y 100 son umbrales
134   edge = cv2.Canny(src1, 50, 100)
135   #cv.imshow("Canny_edge_1", edge)
136   #print(edge)
137   return edge
138
139
140 ##=====================================================================
141 ## Creando modelo de analisis PCA y transformando DataSet
142 ## (Guardandolo si es requerido)
143 ## Return: Modelo, Datos transformados
144 ##=====================================================================
145 def Img_PCA(images, label, n=115, nombre="", PCA_save=False, Data_save=False):
146   ## Tiempo de inicio
147   start_time = time()
148   ## Crear modelo PCA de n componentes
149   pca = PCA(n_components=n)
150   ## Creando modelo basado en informacion del DataSet
151   pca = pca.fit(images)
152   ## Transformando el data Set
153   pcas = pca.transform(images)
154   print('=====================================================================')
155   print("Se ha creado correctamente el modelo PCA y transformado el DataSet")
156   print('=====================================================================')
157   ## Guardando modelo de ser necesario
```

```
158   if (PCA_save):
159     with open(nombre+'_PCA.pkl', 'wb') as f:
160       pickle.dump(pca, f)
161   print('==========================================================================================')
162   print("Se ha guardado exitosamente el modelo PCA")
163   print('==========================================================================================')
164
165   ## Guardando DataSet con PCA de ser necesario
166   if (Data_save):
167     columns=[]
168     for i in range (0,n):
169       columns.append('C%i' %(i+1))
170
171     principalDF = pd.DataFrame(data = pcas, columns = columns)
172     finalDF = principalDF.to_dict()
173     namess={}
174     namesss={}
175     for i in range(0,len(finalDF['C1'])):
176       namess[i]=label['Fail'][i]
177       namesss[i]=label['Line'][i]
178
179     finalDF.update({'Fails':namess})
180     finalDF.update({'Line':namesss})
181     df=pd.DataFrame.from_dict(finalDF)
182     df.to_csv (os.path.join(nombre+"_with_PCA.csv"), index = False, mode="a", header=not os.
      path.isfile(nombre+"_with_PCA.csv"))
183   print('==========================================================================================')
184   print("Se a guardado exitosamente el DataSet transformado con el modelo PCA")
185   print('==========================================================================================')
186
187   ## Finalizando la funcion
188   ## Tiempo de ejecucion
189   print('==========================================================================================')
190   print("Tiempo de carga de la base de datos: ", seg_to_time(time() - start_time))
191   print('==========================================================================================')
192   ## Return modelo y array de imagenes por componente
193   return pca, pcas
194
195 ##=============================================================
196 ## Cargando en arrays las bases de datos
197 ## Parametros: dirname (Direccion del Data Set), preprocesado
198 ## Return: images, qty, label
199 ## (Array de imagenes 456x492),(Diccionario con cantidad de fallos por
200 ##     tipo de fallos), (Diccionario con fallo y linea por imagen)
201 ##=============================================================
202 def CargarImgs(dirname, preprocess='Clean'):
```

```
203 ## Tiempo de inicio
204 start_time = time()
205
206 imgpath = dirname + os.sep
207 ## Inicializando variables
208 images = []
209 qty={}
210 label={'Fail':[],'Line':[]}
211 cant=0
212 print('==========================================================================================')
213 print("Leyendo imagenes de ",imgpath)
214
215 ##Leyendo imagenes y convertirlas en array
216 for root, dirnames, filenames in os.walk(imgpath):
217   for filename in filenames:
218     if re.search("\.(jpg|jpeg|png|bmp|tiff|BMP|JPG)$", filename):
219       cant=cant+1
220       filepath = os.path.join(root, filename)
221       ## Abriendo imagen con el preprocesado
222       if (preprocess=="Sobel"):
223         img=Sobel(filepath)
224       elif (preprocess=="Canny"):
225         img=Canny(filepath)
226       elif (preprocess=="Clean"):
227         img=Clean(filepath)
228       ## Estandarizar tamano
229       img=cv2.resize(img, (656,492), interpolation = cv2.INTER_AREA)
230       ## Guardando en lista como array
231       img=np.asarray(img)
232       images.append(img.flatten())
233       ##Guardar lista de categoria por imagen
234       label['Fail'].append(os.path.basename(root))
235       label['Line'].append(os.path.basename(os.path.dirname(root)))
236       ## Para ubicar cual se esta leyendo
237       print ("Leyendo.......", os.path.basename(os.path.dirname(root)),".......",os.path.
  basename(root),".......",str(cant))
238       if not os.path.basename(root) in qty:
239         qty[os.path.basename(root)]=1
240       else:
241         qty[os.path.basename(root)]+=1
242 print('==========================================================================================')
243
244 print('==========================================================================================')
245 print("Tiempo de carga de la base de datos: ", seg_to_time(time() - start_time))
246 print('Carpetas leidas:',len(qty))
247 print("Imagenes por carpeta", qty)
```

```
248   print('Total de imagenes leidas:',cant)
249   print('=============================================================================')
250
251   ## Salidas de la funcion
252   return images, qty, label
253
254 ##=================================================================
255 ## Creando conjuntos de entrenamiento
256 ## Parametros: images, qty, label, TestSize, preprocess
257 ## Return: train_X, valid_X, train_label, valid_label, test_X, test_Y_one_hot, nClasses,
      failures
258 ##=================================================================
259 def ConjuntosEntrenamiento(images, qty, label, preprocess, TestSize=0.2):
260   ## Tiempo de inicio
261   start_time = time()
262   ## Correcion de fallos de string a int
263   failures=list(qty)
264   ## Correcion de lineas de string a int
265   lineas=list(np.unique(label['Line']))
266   ## Para guardar fallos y lineas por imagen
267   labels=[]
268   lines=[]
269   ## Orden:
270   ## Fails:[BG,Broke,ChipOut,Empty,Needle,Recolection]
271   ## Lines:[WLP,uBAW]
272   for i in label['Fail']:
273     labels.append(failures.index(i))
274   for i in label['Line']:
275     lines.append(lineas.index(i))
276   ## Definiendo variables de entrenamiento y sus formatos
277   y = np.array(labels)
278   X_img = np.array(images, dtype=np.uint8) # Convertir lista a numpy
279   X_line = np.array(lines)
280   ## Encontrar un unico numero para cada clase
281   classes = np.array(list(qty))
282   nClasses = len(classes)
283   print('=============================================================================')
284   print('Total de numero de salidas : ', nClasses)
285   print('Clases de salida : ', classes)
286   print('=============================================================================')
287
288   ## Creando grupos de entrenamiento y test
289   train_img,test_img,train_line,test_line,train_Y,test_Y = train_test_split(X_img,X_line,y,
        test_size= TestSize)
290   print('=============================================================================')
291   print('Training data shape : ', train_img.shape, train_line.shape, train_Y.shape)
```

```
292   print('Testing data shape : ', test_img.shape, test_line.shape, test_Y.shape)
293
294   train_img = train_img.astype('float32')
295   test_img = test_img.astype('float32')
296   if (preprocess=='Clean'):
297     train_img = train_img/255
298     test_img = test_img/255
299
300   ## Cambiando fallos de categorical a one-hot encoding
301   train_line_one_hot = to_categorical(train_line)
302   test_line_one_hot = to_categorical(test_line)
303   ## Cambiando fallos de categorical a one-hot encoding
304   train_Y_one_hot = to_categorical(train_Y)
305   test_Y_one_hot = to_categorical(test_Y)
306
307   ## Mostrando cambio de las clases de categorical a one-hot encoding
308   print('Clase original:', train_Y[0])
309   print('Despues de la conversion a one-hot:', train_Y_one_hot[0])
310   print('Linea original:', train_line[0])
311   print('Despues de la conversion a one-hot:', train_line_one_hot[0])
312   print('================================================================================')
313
314   ## Creando subconjunto de validacion a partir del subconjunto de entrenamiento
315   train_img,valid_img,train_line,valid_line,train_label,valid_label = train_test_split(
        train_img, train_line_one_hot, train_Y_one_hot, test_size= TestSize, random_state=13)
316
317   print('================================================================================')
318   print("Tiempo de creacion de los grupos para el entrenamiento: ", seg_to_time(time() -
        start_time))
319   print('================================================================================')
320
321   ## Salidas de la funcion
322   return train_img, valid_img, train_line, valid_line, train_label, valid_label, test_img,
        test_line_one_hot, test_Y_one_hot, nClasses, failures, lineas
323
324 ##============================================================
325 ## Creando conjuntos de entrenamiento
326 ## Parametros: images, qty, label, TestSize, preprocess
327 ## Return: train_X, valid_X, train_label, valid_label, test_X, test_Y_one_hot, nClasses,
      failures
328 ##============================================================
329 def ConjuntosEntrenamiento_PCA(path, n, TestSize=0.2):
330   ## Tiempo de inicio
331   start_time = time()
332
333   # Carga DataSet desde CSV
```

```
DataSet_O=pd.read_csv(path)
print('==========================================================================================')
print('Datos del DataSet con PCA original:')
print(DataSet_O)
DataSet_O.info()
print('==========================================================================================')
i=0
DataSet=pd.DataFrame()
if ((n+1)!=len(DataSet_O.columns)):
  for column in DataSet_O:
    if (column == ('C%i' %(n+1))):
      DataSet.insert(i,'Fails',DataSet_O['Fails'])
      break
    else:
      DataSet.insert(i,column,DataSet_O[column])
    i+=1
else:
  DataSet=DataSet_O

## Correcion de fallos de string a int
failures=list(DataSet['Fails'].unique())
## Correcion de lineas de string a int
#lineas=list(DataSet['Line'].unique())
## Para guardar fallos y lineas por imagen
labels=[]
lines=[]
## Orden:
##  Fails:[BG,Broke,ChipOut,Empty,Needle,Recolection]
##  Lines:[WLP,uBAW]
for i in DataSet['Fails']:
  #labels.append(failures.index(i))
  if i == 'Empty_Nozzel':
    labels.append(0)
  elif i == 'Recolection_Issue':
    labels.append(1)
  elif i == 'BG_Lines':
    labels.append(2)
  else:
    labels.append(3)
failures=['Empty_Nozzel', 'Recolection_Issue', 'BG_Lines','Broken_Die']

#for i in DataSet['Line']:
  #lines.append(lineas.index(i))
## Definiendo variables de entrenamiento y sus formatos
DataSet=DataSet.drop(['Fails'],axis=1)
DataSet=DataSet.drop(['Line'],axis=1).values.tolist()
```

```
y = np.array(labels)
X_img = np.array(DataSet) # Convertir lista a numpy
#X_line = np.array(lines)
## Encontrar un unico numero para cada clase
classes = np.array(failures)
nClasses = len(classes)
print('========================================================================================')
print('Total de numero de salidas : ', nClasses)
print('Clases de salida : ', classes)
print('========================================================================================')

## Creando grupos de entrenamiento y test
train_img,test_img,train_Y,test_Y = train_test_split(X_img,y,test_size= TestSize,shuffle=
  True)
#train_img,test_img,train_line,test_line,train_Y,test_Y = train_test_split(X_img,X_line,y,
  test_size= TestSize,shuffle=True)
print('========================================================================================')
print('Training data shape : ', train_img.shape, train_Y.shape)
print('Testing data shape : ', test_img.shape, test_Y.shape)

## Cambiando fallos de categorical a one-hot encoding
#train_line_one_hot = to_categorical(train_line)
#test_line_one_hot = to_categorical(test_line)
## Cambiando fallos de categorical a one-hot encoding
train_Y_one_hot = to_categorical(train_Y)
test_Y_one_hot = to_categorical(test_Y)

## Mostrando cambio de las clases de categorical a one-hot encoding
print('Clase original:', train_Y[0])
print('Despues de la conversion a one-hot:', train_Y_one_hot[0])
#print('Linea original:', train_line[0])
#print('Despues de la conversion a one-hot:', train_line_one_hot[0])
print('========================================================================================')

## Creando subconjunto de validacion a partir del subconjunto de entrenamiento
train_img,valid_img,train_label,valid_label = train_test_split(train_img, train_Y_one_hot,
  test_size= TestSize, random_state=13, shuffle=True)
#train_img,valid_img,train_line,valid_line,train_label,valid_label = train_test_split(
  train_img, train_line_one_hot, train_Y_one_hot, test_size= TestSize, random_state=13,
  shuffle=True)

train_img = train_img.astype('float32')
test_img = test_img.astype('float32')
valid_img = valid_img.astype('float32')
## Estandarizar
std1=StandardScaler()
```

```
421   std1=std1.fit(train_img)
422   train_img=std1.transform(train_img)
423   std2=StandardScaler()
424   std2=std2.fit(test_img)
425   test_img=std2.transform(test_img)
426   std3=StandardScaler()
427   std3=std3.fit(valid_img)
428   valid_img=std3.transform(valid_img)
429
430   print('==========================================================================')
431   print("Tiempo de creacion de los grupos para el entrenamiento: ", seg_to_time(time() -
        start_time))
432   print('==========================================================================')
433
434   ## Salidas de la funcion
435   #return train_img, valid_img, train_line, valid_line, train_label, valid_label, test_img,
        test_line_one_hot, test_Y_one_hot, nClasses, failures, lineas
436   return train_img, valid_img, train_label, valid_label, test_img, test_Y_one_hot, nClasses,
        failures
437
438
439 ##=============================================================
440 ## Creando la redes neuronales convolutiva
441 ## Parametros: nClasses,activation
442 ## Return: failures_model
443 ##=============================================================
444 def CreacionCNN1(nClasses,activation):
445   ## Tiempo de inicio de ejecucion
446   start_time = time()
447
448   ## Entradas de las redes
449   Input_IMG=Input(shape=(492, 656,1), name="IMG")
450   Input_Line=Input(shape=(2,), name="Line")
451
452   ## Creacion de modelo para la imagen
453   model_img=Conv2D(128, kernel_size=(3, 3),activation=activation,padding='same')(Input_IMG)
454   model_img=(Conv2D(64, kernel_size=(3, 3),activation=activation,padding='same'))(model_img)
455   model_img=MaxPooling2D((2, 2),padding='same')(model_img)
456   model_img=Flatten()(model_img)
457   model_img=Dense(64, activation=activation)(model_img)
458   model_img=Dropout(0.1)(model_img)
459   model_img = Model(inputs=Input_IMG, outputs=model_img) # Definiendo modelo de img
460
461   ## Creacion de modelo final
462   failures_model=layers.concatenate([model_img.output, Input_Line])
463   failures_model=Dense(32, activation=activation)(failures_model)
```

```
464   failures_model = Dense(nClasses, activation=activation)(failures_model)
465   failures_model = Model(inputs=[model_img.input, Input_Line], outputs=failures_model) #
        Modelo final
466
467   ## Compilando y optimizando red
468   failures_model.summary()
469
470   return failures_model
471
472 ##=============================================================
473 ## Agregando una capa oculta de 64
474 def CreacionCNN2(nClasses,activation):
475   ## Tiempo de inicio de ejecucion
476   start_time = time()
477
478   ## Entradas de las redes
479   Input_IMG=Input(shape=(492, 656,1), name="IMG")
480   Input_Line=Input(shape=(2,), name="Line")
481
482   ## Creacion de modelo para la imagen
483   model_img=Conv2D(128, kernel_size=(3, 3),activation=activation,padding='same')(Input_IMG)
484   model_img=(Conv2D(64, kernel_size=(3, 3),activation=activation,padding='same'))(model_img)
485   model_img=MaxPooling2D((2, 2),padding='same')(model_img)
486   model_img=(Conv2D(64, kernel_size=(3, 3),activation=activation,padding='same'))(model_img)
487   model_img=MaxPooling2D((2, 2),padding='same')(model_img)
488   model_img=Flatten()(model_img)
489   model_img=Dense(64, activation=activation)(model_img)
490   model_img=Dropout(0.1)(model_img)
491   model_img = Model(inputs=Input_IMG, outputs=model_img) # Definiendo modelo de img
492
493   ## Creacion de modelo final
494   failures_model=layers.concatenate([model_img.output, Input_Line])
495   failures_model=Dense(32, activation=activation)(failures_model)
496   failures_model = Dense(nClasses, activation=activation)(failures_model)
497   failures_model = Model(inputs=[model_img.input, Input_Line], outputs=failures_model) #
        Modelo final
498
499   ## Compilando y optimizando red
500   failures_model.summary()
501
502   return failures_model
503
504 ##=============================================================
505 ## Quitando Capa oculta
506 def CreacionCNN3(nClasses,activation):
507   ## Tiempo de inicio de ejecucion
```

```
508   start_time = time()
509
510   ## Entradas de las redes
511   Input_IMG=Input(shape=(492, 656,1), name="IMG")
512   Input_Line=Input(shape=(2,), name="Line")
513
514   ## Creacion de modelo para la imagen
515   model_img=Conv2D(128, kernel_size=(3, 3),activation=activation ,padding='same')(Input_IMG)
516   model_img=MaxPooling2D((2, 2),padding='same')(model_img)
517   model_img=Flatten()(model_img)
518   model_img=Dense(64, activation=activation)(model_img)
519   model_img=Dropout(0.1)(model_img)
520   model_img = Model(inputs=Input_IMG, outputs=model_img) # Definiendo modelo de img
521
522   ## Creacion de modelo final
523   failures_model=layers.concatenate([model_img.output, Input_Line])
524   failures_model=Dense(32, activation=activation)(failures_model)
525   failures_model = Dense(nClasses, activation=activation)(failures_model)
526   failures_model = Model(inputs=[model_img.input, Input_Line], outputs=failures_model) #
        Modelo final
527
528   ## Compilando y optimizando red
529   failures_model.summary()
530
531   return failures_model
532
533
534 ##=============================================================
535 ## Entrenar la CNN
536 ## La ubicacion de guardado deber terminar en .h5py
537 ## Parametros: ubicacion, experimento, red, failures_model, failures, lineas, preprocess,
      epochs, batch_size, INIT_LR, losss, optimizerr, activation, train_img, valid_img,
      train_line, valid_line, train_label, valid_label, test_img, test_line_one_hot,
      test_Y_one_hot
538 ## Return: test_accuracy, test_loss, sobre_entrenamiento (valores)
539 ##=============================================================
540 def EntrenamientoCNN(ubicacion, experimento, red, failures_model, failures, lineas, preprocess
      , epochs, batch_size, INIT_LR, losss, optimizerr, activation, train_img, valid_img,
      train_line, valid_line, train_label, valid_label, test_img, test_line_one_hot,
      test_Y_one_hot):
541   ## Tiempo de inicio de ejecucion
542   start_time = time()
543
544   ## Redimensionando
545   if (preprocess=='Clean'):
546     train_img = train_img.reshape(-1, 492, 656, 1)
```

```
547     valid_img = valid_img.reshape(-1, 492, 656, 1)
548     test_img = test_img.reshape(-1, 492, 656, 1)
549 
550   ## Balancear Data Set
551   y_integers = np.argmax(train_label, axis=1)
552   classWeight = compute_class_weight('balanced', np.unique(y_integers), y_integers)
553   classWeight = dict(enumerate(classWeight))
554 
555   ## Estableciendo Optimiezers
556   if (optimizerr == 'SGD'):
557     optimizer=SGD(learning_rate=INIT_LR)
558   elif (optimizerr=='RMSprop'):
559     optimizer=RMSprop(learning_rate=INIT_LR)
560   elif (optimizerr=='Adagrad'):
561     optimizer=Adagrad(learning_rate=INIT_LR)
562   ## Estableciendo Losses
563   if (losss == 'CategoricalCrossentropy'):
564     loss=CategoricalCrossentropy()
565   elif (losss=='MeanSquaredError'):
566     loss=MeanSquaredError()
567   elif (losss=='CategoricalHinge'):
568     loss=CategoricalHinge()
569 
570   failures_model.compile(loss=loss, optimizer=optimizer,metrics=['accuracy'])
571   ## Entrenando
572   failures_model_dropout = failures_model.fit({'IMG':train_img, 'Line':train_line},
      train_label, batch_size=batch_size,epochs=epochs,verbose=1,validation_data=({'IMG':
      valid_img, 'Line':valid_line}, valid_label), shuffle=False, class_weight=classWeight) #,
      callbacks=[tensorboard_callback]
573 
574   ## Finalizando la funcion y guardando el modelo y sus datos
575   ## Guardar modelo
576   failures_model.save(ubicacion)
577 
578   ## Guardar tipos de fallos
579   with open(os.path.join(ubicacion,"FAILURES"), "wb") as f:
580     pickle.dump(failures,f)
581     f.close()
582   ## Guardar tipos de lineas
583   with open(os.path.join(ubicacion,"LINES"), "wb") as f:
584     pickle.dump(lineas,f)
585     f.close()
586 
587   ## Evaluando modelos con datos de test
588   test_eval = failures_model.evaluate({'IMG':test_img, 'Line':test_line_one_hot},
      test_Y_one_hot, verbose=1)
```

```
589   test_loss = test_eval[0]
590   test_accuracy = test_eval[1]
591   print('=================================================================================')
592   print('Test loss:', test_loss)
593   print('Test accuracy:', test_accuracy)
594   print('=================================================================================')
595
596   ## Tiempo de entrenamiento
597   tiempo=time()-start_time
598
599   ## Guardar graficas de entrenamiento
600   ## Guardando imagen con la arquitectura
601   keras.utils.plot_model(failures_model, os.path.join(ubicacion,"Arquitectura.png"),
        show_shapes=True)
602
603   ## Guardando graficas de entrenamiento (presicion y perdida)
604   accuracy = failures_model_dropout.history['accuracy']
605   val_accuracy = failures_model_dropout.history['accuracy']
606   loss = failures_model_dropout.history['loss']
607   val_loss = failures_model_dropout.history['val_loss']
608   epoch = range(len(accuracy))
609
610   plt.plot(epoch, accuracy, 'bo', label='Training accuracy')
611   plt.plot(epoch, val_accuracy, 'b', label='Validation accuracy')
612   plt.title('Training and validation accuracy')
613   plt.xlabel('Epoch')
614   plt.ylabel('Accuracy')
615   plt.legend()
616   plt.savefig(os.path.join(ubicacion,"Training and validation accuracy.png"))
617   plt.figure()
618   plt.plot(epoch, loss, 'bo', label='Training loss')
619   plt.plot(epoch, val_loss, 'b', label='Validation loss')
620   plt.title('Training and validation loss')
621   plt.xlabel('Epoch')
622   plt.ylabel('Loss')
623   plt.legend()
624   plt.savefig(os.path.join(ubicacion,'Training and validation loss.png'))
625
626 ## Guardando datos de entranamiento en CSV
627 ## Calculando sobreentrenamiento (aproximadamente)
628   if (loss[len(epoch)-1]<val_loss[len(epoch)-1]):
629     sobre_entrenamiento="YES"
630   else:
631     sobre_entrenamiento="NO"
632
633   ## Creadon lista con el numero de experimento para cada epoca
```

```
634   nom_experimento=[]
635   for i in list(epoch):
636     nom_experimento.append("Experimento %s" % experimento)
637   ## Guardar datos de grafica en CSV
638   data={'Experimento': nom_experimento, 'epoch':list(epoch),'Accuracy':accuracy, 'Validation
        Accuracy':val_accuracy, 'Loss':loss, 'Validation Loss':val_loss}
639   df=pd.DataFrame.from_dict(data)
640   df.to_csv ("Training_Data.csv", index = False, mode="a", header=not os.path.isfile("
        Training_Data.csv"))
641
642   ## Guardar resultados de experimentos en CSV
643   resultados={'Experimento': [experimento], 'Learning Rate':[float(INIT_LR)],'Epochs':[float(
        epochs)], 'Batch Size':[float(batch_size)], 'Red':[red], 'Optimizer':[optimizerr], 'Loss'
        :[losss],'Activation':[activation], 'Test loss':[test_eval[0]], 'Test accuracy':[test_eval
        [1]], 'Sobreentrenado':[sobre_entrenamiento], 'Tiempo':[float(tiempo)],'Preprocess':[
        preprocess]}
644   df2=pd.DataFrame.from_dict(resultados)
645   df2.to_csv ("Results_Data.csv", index = False, mode="a", header=not os.path.isfile("
        Results_Data.csv"))
646
647 ## Guardar parametros y resultados de entrenamiento
648   f=open(os.path.join(ubicacion,"DATA.txt"), "wb")
649   f.write(b"Learning Rate:   %f \n" % float(INIT_LR))
650   f.write(b"Epochs:   %f \n" % float(epochs))
651   f.write(b"Batch Size:   %f \n" % float(batch_size))
652   f.write(b"Test loss:   %f \n" % test_eval[0])
653   f.write(b"Test accuracy:   %f \n" % test_eval[1])
654   f.write(b"Training network time: %0.4f seconds. \n" % float(tiempo))
655   f.close()
656
657   # Guardar estructura de CNN
658   with open(os.path.join(ubicacion,'DATA.txt'), 'a') as f:
659     with redirect_stdout(f):
660       failures_model.summary()
661
662   ## Tiempo de ejecucion
663   print('==========================================================================================')
664   print("Tiempo de entrenamiento de la CNN: ", seg_to_time(time() - start_time))
665   print('==========================================================================================')
666
667   #Salidas de la funcion
668   return test_accuracy, test_loss, sobre_entrenamiento
669
670 ##=================================================================
671 ## Cargar modelo CNN
672 ## Parametro: path
```

```
673 ## Return: pca_model
674 ##========================================================================
675 def CargarPCA(path):
676   ## Tiempo de inicio de ejecucion
677   start_time = time()
678 
679   with open(path, 'rb') as f:
680     pca_model=pickle.load(f)
681 
682   ## Tiempo de ejecucion
683   print('========================================================================')
684   print("Tiempo de carga del modelo PCA: ", seg_to_time(time() - start_time))
685   print('========================================================================')
686 
687   return pca_model
688 
689 ##========================================================================
690 ## Cargar modelo CNN
691 ## Parametro: path
692 ## Return: failures_model, failures, lineas
693 ##========================================================================
694 def CargarModeloCNN(path):
695   ## Tiempo de inicio de ejecucion
696   start_time = time()
697 
698   # Cargado modelo
699   failures_model = keras.models.load_model(path)
700 
701   ## Cargando fallos
702   with open(os.path.join(path,"FAILURES"), "rb") as f:
703     failures=pickle.load(f)
704     f.close()
705 
706 ## Cargando lineas
707   with open(os.path.join(ubicacion,"LINES"), "rb") as f:
708     lineas=pickle.load(f)
709     f.close()
710 
711   ## Tiempo de ejecucion
712   print('========================================================================')
713   print("Tiempo de carga del modelo CNN: ", seg_to_time(time() - start_time))
714   print('========================================================================')
715 
716   return failures_model, failures, lineas
717 
718 ##========================================================================
```

```
719 ## Cargar modelo CNN
720 ## Parametro: CNN_path,pca_path
721 ## Return: failures_model, failures, lineas, pca_model
722 ##=============================================================================
723 def CargarModelos(CNN_path,pca_path):
724   ## Tiempo de inicio de ejecucion
725   start_time = time()
726 
727   ## Cargando modelo CNN
728   failures_model, failures, lineas = CargarModeloCNN(CNN_path)
729 
730   ## Cargando modelo PCA
731   pca_model = CargarPCA(pca_path)
732 
733   ## Tiempo de ejecucion
734   print('==============================================================================')
735   print("Tiempo de carga de los modelos: ", seg_to_time(time() - start_time))
736   print('==============================================================================')
737 
738   ## Salidas de la funcion
739   return failures_model, failures, lineas, pca_model
```

Código A.1: Biblioteca propia desarrollada para el proyecto.

A.1.2. Código principal para ejecución del sistema desarrollado

```
1 #!/usr/bin/env python
2 # -*- coding: utf-8 -*-
3 ##
4 ##   VCTQ_Proyect.py
5 ##   Copyright 2021
6 ##   Qorvo, Inc
7 ##           Jeremy Fuentes <Ext. 508677>
8 ##   August 2021
9 ##=============================================================================
10 ## Importando bibliotecas
11 ##=============================================================================
12 import sys,os,re
13 from time import time
14 from PIL import Image, ImageOps
15 import numpy as np
16 import keras
17 import pickle
18 import pandas as pd
19 import shutil
```

```
import cv2
from datetime import datetime
from suds.client import Client
from urllib.parse import urljoin
import pyodbc
def seg_to_time(seg):
    try:
        # Para informacion de fallos
        import sys

        days = int(seg // 86400)
        seg -= 86400*days

        hrs = int(seg // 3600)
        seg -= 3600*hrs

        mins = int(seg // 60)
        seg -= 60*mins

        seg = round(seg,3)

        formato = str(days)+' dias : '+str(hrs)+' hrs : '+str(mins)+' min : '+str(seg)+' seg'
        return formato

    except:
        # Alerta, tipo de error, descripcion and  localizacion
        exc_type, exc_obj, exc_tb = sys.exc_info()
        fname = os.path.split(exc_tb.tb_frame.f_code.co_filename)[1]
        print("Error al establecer el formato de tiempo de ejecucion: \n   ", sys.exc_info()
    [0], "  \n   ", sys.exc_info()[1], "    \n   ", os.path.split(exc_tb.tb_frame.f_code.
    co_filename)[1], " -- Line ", sys.exc_info()[2].tb_lineno)
##=============================================================
## Preprocesar la imagen para solo tener contornos mediante Canny
## Parametros: (Direccion de la imagen)
## Return (Imagen de contornos)
##=============================================================
def Canny(path):
  img=cv2.imread(path, cv2.IMREAD_GRAYSCALE)
  img=cv2.resize(img, (66,50), interpolation=cv2.INTER_AREA)
  img=img[10:40, 13:53]
  gray = cv2.GaussianBlur(img,(3,3),0)
  # Encuentra el gradiente en la direcci          n X
  grad_x = cv2.Sobel(gray, cv2.CV_16SC1, 1, 0)
  # Encuentra el gradiente en la direcci          n y
  grad_y = cv2.Sobel(gray, cv2.CV_16SC1, 0, 1)
  # Convertir el valor del gradiente a 8 bits
```

```
  x_grad = cv2.convertScaleAbs(grad_x)
  y_grad = cv2.convertScaleAbs(grad_y)
  # Combina dos gradientes
  src1 = cv2.addWeighted(x_grad, 0.5, y_grad, 0.5, 0)
  # Combine gradientes con algoritmo canny, donde 50 y 100 son umbrales
  edge = cv2.Canny(src1, 50, 100)
  #cv.imshow("Canny_edge_1", edge)
  #print(edge)
  return edge
##================================================================
## Clasificacion mediante NN
## Parametros: (Direccion de carpeta a clasificar)
## Return diccionario con cant fallos,tiempo de clasificacion carpeta
##================================================================
def Clasificar(imgpath):
    fallos={}
    cant=0
    for n in failures:
        fallos[n]=[0]
    start_time = time()
    for root, dirnames, filenames in os.walk(imgpath):
        for filename in filenames:
            if re.search("\.(jpg|jpeg|png|bmp|tiff|BMP|JPG)$", filename):
                filepath = os.path.join(root, filename)
                ## Abrir y preprocesar imagen
                img=Canny(filepath)
                ## Estandarizar tamano
                img=cv2.resize(img, (66,50), interpolation = cv2.INTER_AREA)
                ## Guardando en lista como array
                img=np.asarray(img)
                ## Haciedo el flatten
                img=img.flatten()
                ## Pasar por modelo PCA
                img=pca_model.transform([img,img])
                ## Dandole formato para
                img=img.astype('float32')
                img=img[0]
                ## Prediciendo
                predicted_fail = failures_model.predict(img)
                predicted_fail[0] = [float(i)/max(predicted_fail[0]) for i in predicted_fail
    [0]]
                ## El que sea mayor
                for i, img_tagged in enumerate(predicted_fail):
                    fail=failures[predicted_fail[0].tolist().index(max(img_tagged))]
                    print(filename, '.....', fail,'.....',img_tagged)
                    if not os.path.exists(os.path.join(root,fail)):
```

```
109                         os.makedirs(os.path.join(root,fail))
110                     shutil.move(os.path.join(root,filename), os.path.join(root,fail,filename))
111                     fallos[fail][0]+=1
112                     cant+=1
113         break
114     tiempo=time()-start_time
115     return fallos,tiempo,cant
116 ##=================================================================
117 ## Consultar rollos en Reel Disposition
118 ## Return  diccionario con 'Rollo','Linea','Modelo','Fecha' (fecha en formato datetime)
119 ##=================================================================
120 def ConsultarWCF():
121     start_time=time()
122     wsdl = "http://sjo0ws09.corp.qorvo.com:50014/WcfService/Service.svc?singleWsdl"
123     client = Client(wsdl)
124     result = client.service.LotWIPOnlineQuery_Camstar("Reel Disposition")
125     print('==========================================================================')
126     print('Tiempo de consulta: ', seg_to_time(time()-start_time))
127     print('==========================================================================')
128 
129     Pendientes=[]
130 
131     for i in result[0]:
132         rollo=i[0]
133         if not rollo.find('CR5')==(-1):
134             linea='WLP'
135         elif not rollo.find('CR6')==(-1):
136             linea='uBAW'
137         modelo=i[1].split('  ',1)[0]
138         modelo=i[1].split('TR13' ,1)[0]
139         qty=[2]
140         fecha=datetime.strptime(i[3], '%Y-%m-%dT%H:%M:%S.%f%z')
141 
142         Pendientes.append({'Rollo':rollo,'Linea':linea,'Modelo':modelo, 'Fecha':fecha})#fecha.
    strftime('%d/%m/%Y  %H:%M:%S')
143     return Pendientes
144 ##=================================================================
145 ## Ubicar carpeta a clasificar en base a la informacion de los rollos de Reel Disposition
146 ## Return url de la ubicacion
147 ##=================================================================
148 def Obtener_path(rollo):
149     ## Obteniendo ubicacion
150     #root = urljoin('\\sjo0fs02\DTR\DTR_images', rollo['Modelo'],rollo['Rollo'])
151     root=os.path.join('Y:', rollo['Modelo'],rollo['Rollo'])
152     if not os.path.exists(root): # Verificar si existe
153         return None
```

```
154     else:
155         ## Determinando carpeta a clasificar
156         for i in os.listdir(root):
157             if re.search('station1', i, re.IGNORECASE):
158                 if os.path.exists(os.path.join(root,i,'Presence')):
159                     return os.path.join(root,i,'Presence')
160                 else:
161                     return os.path.join(root,i)
162             elif re.search('top', i, re.IGNORECASE):
163                 if os.path.exists(os.path.join(root,i,'Presence')):
164                     return os.path.join(root,i,'Presence')
165                 else:
166                     return os.path.join(root,i)
167 ##=============================================================
168 ## Definir cuales rollos ya se han clasificado
169 ##=============================================================
170 def Obtener_rollos_Clasificados():
171     rollos=[]
172 ## ============================================================
173 ## Establecer conexion
174 ## ============================================================
175     server='SJO2SQL10'
176     bd='VCTQ_Report'
177     user='sjo_usrsql_vctq'
178     password='aWLQVj9UfxaWUA&a'
179     conexion=pyodbc.connect('DRIVER={SQL server}; SERVER='+server+'; DATABASE='+bd+'; UID='+
        user+'; PWD='+password)
180     cursor=conexion.cursor()
181 ## ============================================================
182 ## Recibir datos
183 ## ============================================================
184     cursor.execute('select [dbo].[VCTQ_Report].[Reel] from VCTQ_Report')
185     reel=cursor.fetchone()
186     while reel:
187       rollos.append(reel[0])
188       reel=cursor.fetchone()
189     return rollos
190 ##=============================================================
191 ## Enviar correo en caso de ser necesario
192 ##=============================================================
193 def EnviarCorreo(mensaje):
194     ## Creando instancia de objeto de mensaje
195     msg = MIMEMultipart()
196     ## Datos para envi
197     msg['From'] = "Smtprelay-central.corp.qorvo.com"
198     msg['To'] = "ProcesoDTR@Qorvo.com;Provisions@qorvo.com"
```

```
199     msg['Subject'] = "Alerta Por Umbral en DTR"
200     ## Agregar a cuerpo de mensaje
201     msg.attach(MIMEText(mensaje, 'plain'))
202     ## Abriendo servidor SMTP
203     server = smtplib.SMTP('Smtprelay-central.corp.qorvo.com')
204     # Enviando via el servidor
205     server.sendmail(msg['From'], msg['To'], msg.as_string())
206     ## Cerrando servidor
207     server.quit()
208     print ("Mensaje enviado con exito a: %s" % (msg['To']))
209
210 print('================================================================================')
211 print('Bibliotecas cargadas correctamente ')
212 print('================================================================================')
213 ##============================================================================
214 ## Entradas
215 ##============================================================================
216 modelo = 'VCTQ_Network.h5py'
217 pca = 'Canny_PCA.pkl'
218 ##============================================================================
219 ## Cargando modelos
220 ##============================================================================
221 failures_model = keras.models.load_model(modelo)
222 with open(os.path.join(modelo,"FAILURES"), "rb") as f:
223   failures=pickle.load(f)
224   f.close()
225 print(failures)
226 with open(pca, 'rb') as f:
227                 pca_model=pickle.load(f)
228 print('================================================================================')
229 print('Modelos cargados correctamente')
230 print('================================================================================')
231 ##============================================================================
232 ## Ciclo principal
233 ##============================================================================
234 while(True):
235     ## Obteniendo rollos ya clasificados
236     clasificados=Obtener_rollos_Clasificados()
237     ##========================================================================
238     ## Obteniendo rollos en Reel Disposition
239     ##========================================================================
240     try:
241         pendientes=ConsultarWCF()
242     except:
243         continue
244     ##========================================================================
```

```
    ## Clasificando por rollo
    ##=============================================================
    for info in pendientes:
        if (info['Rollo'] in clasificados):
            continue
        ## Ubicando carpeta
        path=Obtener_path(info)
        if path==None:
            continue
        fallos,tiempo,cant=Clasificar(path)
        ## Evaluar si se supero umbrales
        ## Umbral de boquilla vac a
        if (fallos['Empty_Nozzel']/info['Qty'])>0.001:
            EnviarCorreo('Alerta!, el rollo '+info['Rollo']+' supero el umbral de boquillas vac as con un total de '+fallos['Empty_Nozzel']+'por este modo de fallo.'+' Es necesario intervenir la m quina.'+)
        ## Umbral de problema de recolecci n
        if (fallos['Recolection_Issue']/info['Qty'])>0.001:
            EnviarCorreo('Alerta!, el rollo '+info['Rollo']+' supero el umbral de boquillas vac as con un total de '+fallos['Recolection_Issue']+'por este modo de fallo.'+' Es necesario intervenir la m quina.'+)
        ## Umbral de l neas de pulido
        if (fallos['BG_Lines']/info['Qty'])>0.0015:
            EnviarCorreo('Alerta!, el rollo '+info['Rollo']+' supero el umbral de boquillas vac as con un total de '+fallos['BG_Lines']+'por este modo de fallo.'+' Es necesario intervenir la m quina.'+)
        ## Umbral de quebradura en dispositivo
        if fallos['Broken_Die']>3:
            EnviarCorreo('Alerta!, el rollo '+info['Rollo']+' supero el umbral de boquillas vac as con un total de '+fallos['Broken_Die']+'por este modo de fallo.'+' Es necesario intervenir la m quina.'+)
        ## Enviar datos a la base de datos
        EnviarSQL(info['Rollo'],info['Linea'],info['Modelo'],info['Fecha'],info['Qty'],fallos['BG_Lines'],fallos['Broken_Die'],fallos['Empty_Nozzel'],fallos['Recolection_Issue'],datetime.now())
        print('==========================================================')
        print('Clasificacion de: ',info['Rollo'], ' terminada. Tiempo: ',info['Fecha'].strftime('%d/%m/%Y  %H:%M:%S'))
        print('==========================================================')
pendientes=ConsultarWCF()
print(pendientes)
```

Código A.2: Código principal para ejecución del sistema desarrollado.