



UNIVERSIDAD TÉCNICA DEL NORTE

FACULTAD DE INGENIERÍA EN CIENCIAS APLICADAS

ESCUELA DE INGENIERÍA EN MECATRÓNICA

TRABAJO DE GRADO PREVIO A LA OBTENCIÓN DEL
TÍTULO DE INGENIERO EN MECATRÓNICA

TEMA:

“DETECCIÓN DE DEFECTOS EN PAPAS TIPO HOJUELAS
(CHIPS) POR VISIÓN ARTIFICIAL”

AUTOR: BRAYAN ISMAEL DÍAZ QUINCHIGUANGO

DIRECTOR: CARLOS XAVIER ROSERO CHANDI

IBARRA-ECUADOR

2024



UNIVERSIDAD TÉCNICA DEL NORTE

BIBLIOTECA UNIVERSITARIA

AUTORIZACIÓN DE USO Y PUBLICACIÓN A FAVOR DE LA UNIVERSIDAD TÉCNICA DEL NORTE

1. IDENTIFICACIÓN DE LA OBRA

En cumplimiento del Art. 144 de la Ley de Educación Superior, hago la entrega del presente trabajo a la Universidad Técnica del Norte para que sea publicado en el Repositorio Digital Institucional, para lo cual pongo a disposición la siguiente información:

DATOS DE CONTACTO			
CÉDULA DE IDENTIDAD:	1050111523		
APELLIDOS Y NOMBRES:	Díaz Quinchiguango Brayan Ismael		
DIRECCIÓN:	Cristóbal Colón y Peñaherrera – Andrade Marín		
EMAIL:	ismaelbdiazq@outlook.com		
TELÉFONO FIJO:		TELÉFONO MÓVIL:	0990490064

DATOS DE LA OBRA	
TÍTULO:	Detección de defectos en papas tipo hojuelas (chips) por visión artificial
AUTOR (ES):	Díaz Quinchiguango Brayan Ismael
FECHA DE APROBACIÓN: DD/MM/AAAA	23 de julio de 2024
PROGRAMA:	<input checked="" type="checkbox"/> PREGRADO <input type="checkbox"/> POSGRADO
TITULO POR EL QUE OPTA:	Ingeniero en Mecatrónica
ASESOR /DIRECTOR:	Ing. Cosme Damián Mejía Echeverría, MSc. Ing. Carlos Xavier Rosero Chandi, PhD.

2. CONSTANCIAS

El autor (es) manifiesta (n) que la obra objeto de la presente autorización es original y se la desarrolló, sin violar derechos de autor de terceros, por lo tanto la obra es original y que es (son) el (los) titular (es) de los derechos patrimoniales, por lo que asume (n) la responsabilidad sobre el contenido de la misma y saldrá (n) en defensa de la Universidad en caso de reclamación por parte de terceros.

Ibarra, a los 23 días del mes de julio de 2024

EL AUTOR:


.....
Díaz Quinchiguango Brayan Ismael



Universidad Técnica del Norte
Facultad de Ingeniería en Ciencias Aplicadas
Certificación del director del trabajo de grado

En mi calidad de director del trabajo de grado **“Detección de defectos en papas tipo hojuelas (chips) por visión artificial”**, presentado por el egresado Brayan Ismael Díaz Quinchiguango, que opta por el título de ingeniero en Mecatrónica, certifico que el mencionado proyecto fue realizado bajo mi dirección.

Ibarra, 23 de julio de 2024

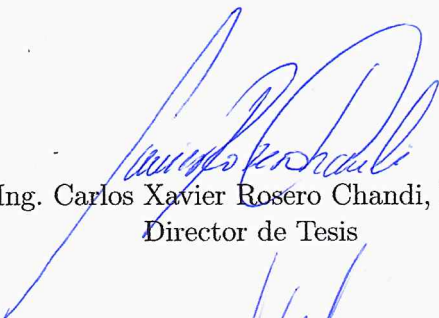


Ing. Carlos Xayier Rosero Chandi, PhD.
Director de Tesis




Universidad Técnica del Norte
Facultad de Ingeniería en Ciencias Aplicadas
Aprobación de Comité Calificador

El Tribunal Examinador del trabajo de titulación “**Detección de defectos en papas tipo hojuelas (chips) por visión artificial**”, elaborado por Brayan Ismael Díaz Quinchiguango, previo a la obtención del título de Ingeniero en Mecatrónica, aprueba el presente informe de investigación en nombre de la Universidad Técnica del Norte.



Ing. Carlos Xavier Rosero Chandi, PhD.
Director de Tesis



Ing. Cosme Damián Mejía Echeverría, MSc.
Asesor de Tesis

Dedicatorias

Este trabajo va dedicado a todas las personas y seres queridos quienes de alguna manera influyeron durante mi etapa universitaria. En especial a mis padres, cuyo amor y cuidado, han sido el pilar fundamental de todos mis logros. Su confianza y sacrificio invaluable serán siempre mi fuente de inspiración y motivación de superación personal. A mi abuelita, quien siempre se ha preocupado por mi bienestar. A mis hermanos, por su apoyo incondicional a lo largo de mi formación académica. A mis sobrinos, quienes me han levantado el ánimo con sus ocurrencias. A mis amigos, por brindarme su amistad y ser partícipes de momentos inolvidables. Este trabajo marca el fin de un etapa significativa de mi vida, todas las decisiones que he tomado me han llevado a este punto y me han permitido conocer y forjar amistades que atesoraré por siempre. Para terminar, a Karlita, quien siempre llevaré en mi corazón.

Con cariño y gratitud, dedico este trabajo a todos ustedes.

Agradecimientos

A Dios, por haberme brindado la fortaleza, entendimiento y perseverancia, permitiéndome superar los retos y alcanzar este logro académico.

Quiero expresar mis agradecimientos a mis amigos y seres queridos, quienes me apoyaron y contribuyeron al desarrollo de este trabajo. A la Universidad Técnica del Norte, por la gestión y material de apoyo para mi formación académica. A mis profesores quienes compartieron sus conocimientos.

Agradezco profundamente a mi Director de trabajo de titulación, el Ing. Xavier Rosero, y mi Asesor, el Ing. Cosme Mejía, por otorgarme su confianza, orientación, experiencia y conocimientos para poder concluir este trabajo. Sobre todo, por su paciencia y comprensión en momentos de dificultad.

A mi estimada profesora, Sheyla Rodríguez, por brindarme su amistad y valiosa ayuda en la revisión y mejora de las traducciones presentes en esta tesis.

Principalmente, deseo expresar mis más sinceros agradecimientos a mis padres y hermanos, por confiar plenamente en mí, por inculcarme valores que me han formado en la persona que soy hoy, y por ser un ejemplo de humildad y superación. Sin ellos, no hubiera sido posible alcanzar esta meta.

Índice general

Cesión de derechos de autor a favor de la Universidad Técnica del Norte	II
Certificación del director del trabajo de grado	III
Aprobación de Comité Calificador	IV
Dedicatorias	V
Agradecimientos	VI
Índice general	VII
Índice de figuras	X
Índice de tablas	XII
Resumen	XIV
Abstract	XV
Introducción	1
Objetivos	2
Objetivo general	2
Objetivos específicos	2
Justificación	2
Alcance	2
1. Revisión literaria	3
1.1. Estado del arte	3
1.1.1. Métodos de detección basados en análisis de color y texturas	3
1.1.2. Métodos de detección basados en aprendizaje automático	6
1.1.3. Métodos de detección basados en aprendizaje profundo	7
1.2. Tecnologías implicadas	10
1.2.1. Inteligencia artificial	10
1.2.2. Visión artificial	11
1.2.3. Aprendizaje automático	11
1.2.4. Redes neuronales artificiales	11
Neurona artificial	12
1.2.5. Funciones de activación	13
1.2.6. Aprendizaje profundo	13
1.2.7. Redes Neuronales Convolucionales	13
Capa convolucional	13
Capa de Pooling	14

Capa totalmente conectada (<i>Fully Connected</i>)	15
1.2.8. Métricas de rendimiento	15
1.2.9. Python	15
1.3. Recapitulación del estado del arte	15
1.4. Propuesta	18
2. Desarrollo	19
2.1. Arquitectura del método propuesto	19
2.2. Adquisición del conjunto de datos	21
2.3. Procesamiento y segmentación de la región de interés (ROI)	21
2.3.1. Creación de modelo U-Net 1	22
Creación del conjunto de datos	22
Arquitectura U-Net	25
Entrenamiento	25
2.4. Clasificador basado en CNN	28
2.4.1. Elaboración del conjunto de datos	29
Aumento de datos	29
Equilibrio de datos	30
2.4.2. Arquitectura CNN	31
2.4.3. Entrenamiento	32
2.5. Método por conteo de píxeles	32
2.5.1. Creación de modelo U-Net 2	32
Conjunto de datos	33
Arquitectura U-Net 2	33
Entrenamiento	33
2.5.2. Cálculo de píxeles	35
3. Pruebas y resultados	36
3.1. Análisis del modelo U-Net 1	36
3.1.1. Criterios de evaluación	36
3.1.2. Resultados de entrenamiento	37
3.1.3. Selección del modelo U-Net 1	37
3.1.4. Comparación de técnicas	40
3.2. Análisis del clasificador CNN	41
3.2.1. Criterios de evaluación	41
3.2.2. Resultados de entrenamiento	43
Primera configuración	43
Segunda configuración	43
Tercera configuración	44
3.2.3. Resultados de evaluación	44
3.2.4. Primer conjunto de prueba	45
3.2.5. Segundo conjunto de prueba	46
3.2.6. Selección de modelo	48
3.2.7. Comparación de técnicas	49
3.3. Análisis del modelo U-Net 2	50
3.3.1. Resultados de entrenamiento	50
3.3.2. Selección del modelo U-Net 2	51
3.3.3. Optimización del modelo U-Net 2 elegido	52
3.3.4. Comparación de técnicas	54
3.4. Análisis del método por conteo de píxeles	54
3.4.1. Criterios de evaluación	54
3.4.2. Resultados de evaluación - primer conjunto de prueba	55

Umbral del 5 %	55
Umbral del 10 %	56
Umbral del 25 %	56
Umbral de 40 %	57
3.4.3. Resultados de evaluación - segundo conjunto de prueba	58
Umbral del 5 %	58
Umbral del 10 %	58
Umbral del 25 %	59
Umbral de 40 %	60
3.4.4. Comparación con el clasificador CNN	60
4. Conclusiones, recomendaciones y trabajos futuros	64
4.1. Conclusiones	64
4.2. Recomendaciones	65
4.3. Trabajos futuros	65
Bibliografía	67
Anexos	70

Índice de figuras

1.1. Imágenes de papas chips categorizadas en 4 grupos	4
1.2. Clasificación de las cuatro categorías de calidad de las papas fritas para los datos de entrenamiento (símbolos llenos) y los datos de prueba (símbolos vacíos) utilizando LDA como criterio de selección	4
1.3. Diagrama esquemático del proceso seguido para la clasificación automática de papas chips	5
1.4. Gráfica de la media de los componentes wavelet. a) Media - componentes wavelet en primer nivel dirección horizontal, b) Media - componentes wavelet en primer nivel dirección vertical, c) Media - componentes wavelet en primer nivel dirección diagonal, d) Media - componentes wavelet en segundo nivel dirección horizontal	5
1.5. Gráficas del clasificador SVM empleando un kernel no lineal de “Polinomio - orden 3” considerando las características extraídas, con un etiquetado de: 0 para la clase positiva, 1 para la clase negativa	6
1.6. Curva ROC del clasificador basado en una ANN	7
1.7. Imágenes obtenidas aplicando el algoritmo propuesto	7
1.8. Imágenes obtenidas aplicando el modelo U-net	8
1.9. Visualización de los mapas de características aprendidas del modelo prototipo y del basado en DenseNet respectivamente	8
1.10. Precisión de clasificación de los modelos entrenados con tamaños de imagen: a) 16 X 16 píxeles, b) 32 X 32 píxeles, c) 64 X 64 píxeles.	9
1.11. Estructura de la CNN	10
1.12. Estructura de la CNN + FCNN	10
1.13. Procedimiento de un sistema de visión	11
1.14. Diagrama de una AN.	12
1.15. Capas básicas de una CNN	14
1.16. Valores de píxeles en una imagen de un solo canal de color	14
2.1. Metodología en visión artificial	19
2.2. Metodología propuesta	20
2.3. Muestras aleatorias por clase.	21
2.4. Número de muestras de imágenes distribuidas en sus respectivas carpetas y clases. a) “Train”, b) “Validation”.	22
2.5. Muestras con ruidos elegidas.	23
2.6. Muestras generadas.	23
2.7. Muestras generadas sin fondo.	24
2.8. Resultados obtenidos del algoritmo.	24
2.9. Muestra extraída del <i>dataset</i> de entrenamiento creado.	25
2.10. Arquitectura U-Net utilizada.	26
2.11. Lr vs pérdida - U-Net 1.	27
2.12. Muestras de predicción antes del entrenamiento del modelo U-Net 1.	27
2.13. Muestras de predicción después del entrenamiento del modelo U-Net 1.	28
2.14. Resultado del método propuesto por el autor.	29

2.15. Resultado aplicando el modelo U-Net 1.	29
2.16. Muestra de un árbol de directorios.	30
2.17. Resultado de <i>data augmentation</i> - directorio “Train”.	30
2.18. Resultados - directorio “Validation”.	30
2.19. Lote de imágenes extraído del <i>dataset</i> creado.	33
2.20. Lr vs pérdida - U-Net 2.	34
2.21. Muestras de predicción antes del entrenamiento del modelo U-Net 2.	34
2.22. Monitoreo de métricas - U-Net 2.	35
2.23. Muestras de predicción después del entrenamiento del modelo U-Net 2.	35
3.1. Monitoreo de métricas con un <i>lr. max</i> de 3.	37
3.2. Curvas de entrenamiento - U-Net 1.	38
3.3. Diagrama de cajas de los modelos U-Net 1.	40
3.4. Muestra de segmentación semántica - modelo U-Net 1.	40
3.5. Resultado de segmentación de ROI por cada método.	41
3.6. Estructura de una MC para problemas de clasificación binaria.	41
3.7. Curvas de entrenamiento del grupo 1.	43
3.8. Curvas de entrenamiento del grupo 2.	44
3.9. Curvas de entrenamiento del grupo 3.	45
3.10. Matrices de confusión del grupo 1 - Primer conjunto de imágenes de prueba.	45
3.11. Matrices de confusión del grupo 2 - Primer conjunto de imágenes de prueba.	46
3.12. Matrices de confusión del grupo 3 - Primer conjunto de imágenes de prueba.	46
3.13. Matrices de confusión del grupo 1 - Segundo conjunto imágenes de prueba.	47
3.14. Matrices de confusión del grupo 2 - Segundo conjunto imágenes de prueba.	47
3.15. Matrices de confusión del grupo 3 - Segundo conjunto imágenes de prueba.	48
3.16. Número de acierto del modelo 6 con el segundo conjunto de pruebas.	49
3.17. Comparación de matrices de confusión - Segundo conjunto de imágenes de prueba.	50
3.18. Curvas de entrenamiento - U-Net 2.	51
3.19. Diagramas de cajas de los modelos - U-net 2.	52
3.20. Diagrama de cajas de los modelos - U-Net 2 con aumento de datos y épocas.	53
3.21. Muestra de segmentación semántica - modelo U-Net 2 seleccionado.	53
3.22. Resultado de segmentación de ROI por cada método - U-net2.	54
3.23. Matriz de confusión con umbral 5% - primer conjunto de prueba.	55
3.24. Matriz de confusión con umbral 10% - primer conjunto de prueba.	56
3.25. Matriz de confusión con umbral 25% - primer conjunto de prueba.	57
3.26. Matriz de confusión con umbral 40% - primer conjunto de prueba.	57
3.27. Matriz de confusión con umbral 5% - segundo conjunto de prueba.	58
3.28. Matriz de confusión con umbral 10% - segundo conjunto de prueba.	59
3.29. Matriz de confusión con umbral 25% - segundo conjunto de prueba.	59
3.30. Matriz de confusión con umbral 40% - segundo conjunto de prueba.	60

Índice de tablas

1.1. Funciones de activación	16
1.2. Tabla comparativa de los diferentes trabajos literarios relacionados al presente proyecto	17
2.1. Procedimiento para generar máscaras binarias.	24
2.2. Clases y pesos	31
2.3. Estructura CNN empleada.	31
3.1. Métricas máx. de los modelos U-Net 1.	37
3.2. Descripción de resultados <i>dice</i> de los modelos U-Net 1.	38
3.3. Descripción de resultados <i>IoU</i> de los modelos U-Net 1.	39
3.4. Resultados estadísticos de los coeficientes <i>dice</i> - U-Net 1.	39
3.5. Resultados estadísticos de los coeficientes <i>IoU</i> - U-Net 1.	39
3.6. Comparativa de coeficientes calculados de cada máscara generada por cada método.	42
3.7. Métricas de rendimiento de los clasificadores - Primer conjunto de imágenes de prueba.	47
3.8. Métricas de rendimiento de los clasificadores - Segundo conjunto imágenes de prueba.	48
3.9. Métricas de rendimiento del grupo 3 - Segundo conjunto imágenes de prueba.	49
3.10. Métricas de rendimiento - Segundo conjunto imágenes de prueba.	50
3.11. Resultados del entrenamiento de los modelos U-Net 2.	50
3.12. Resultados estadísticos de los coeficientes <i>dice</i> - U-Net 2.	51
3.13. Resultados estadísticos de los coeficientes <i>IoU</i> - U-Net 2.	52
3.14. Resultados estadísticos de los coeficientes <i>IoU</i> con aumento de épocas y datos de entrenamiento.	52
3.15. Resultados estadísticos de los coeficientes <i>dice</i> con aumento de épocas y datos de entrenamiento.	53
3.16. Resultados obtenidos por cada técnica de segmentación	55
3.17. Resultados de las métricas de rendimiento y tiempo de ejecución - método con umbral de 5%.	56
3.18. Resultados de las métricas de rendimiento y tiempo de ejecución - método con umbral de 10%.	56
3.19. Resultados de las métricas de rendimiento y tiempo de ejecución - método con umbral de 25%.	57
3.20. Resultados de las métricas de rendimiento y tiempo de ejecución - método con umbral de 40%.	58
3.21. Resultados de las métricas de rendimiento y tiempo de ejecución - método con umbral de 5%.	58
3.22. Resultados de las métricas de rendimiento y tiempo de ejecución - método con umbral de 10%.	59
3.23. Resultados de las métricas de rendimiento y tiempo de ejecución - método con umbral de 25%.	60

3.24. Resultados de las métricas de rendimiento y tiempo de ejecución - método con umbral de 40 %	60
3.25. Comparación de métricas del método por conteo de píxeles.	61
3.26. Resultados de predicción con muestras aleatorias y umbrales de defecto - primer conjunto de imágenes de prueba.	62
3.27. Resultados de predicción con muestras aleatorias y umbrales de defecto - segundo conjunto de imágenes de prueba.	63

Resumen

Como alimento rico en carbohidratos y asequibles, las chips de papa son un aperitivo muy consumido en todo el mundo. Al tratarse de productos agrícolas procesados, son susceptible a presentar defectos que podrían afectar la salud del consumidor. Para cumplir los estándares de calidad, las pequeñas empresas siguen recurriendo a controles de inspección manuales, inconsistentes y laboriosas. El presente estudio propone un marco de visión por ordenador y técnicas de aprendizaje de máquina, a fin de resolver el problema de detección. Se presenta dos métodos: clasificador basado en una Red Neuronal Convolutiva (CNN) y un método por conteo de píxeles. El procedimiento que se ha llevado a cabo incluye la adquisición y estructuración de los datos de entrenamiento disponibles en Kaggle, donde el defecto fue simulado con pigmentos aleatorios creados con un marcador negro en la superficie del objeto de estudio. La tarea de extracción y clasificación del primer método se realiza mediante una arquitectura CNN, donde el preprocesamiento incluye el redimensionamiento, normalización y reducción de ruido. La segmentación de la región de interés (ROI) del área total se efectúa con un método de segmentación semántica (U-Net 1) desarrollado con imágenes y máscaras binarias obtenidas a partir del aumento de datos, eliminación del fondo y umbralización. El segundo método se lleva a cabo utilizando un valor de umbral y un porcentaje calculado a partir del recuento de píxeles de las máscaras binarias generadas por el modelo U-Net 1 y un segundo modelo U-Net 2 creado con datos procesados, que incluye el etiquetado manual de la ROI defectuosa y una transformación. La evaluación con dos conjuntos de imágenes de prueba muestra que el modelo CNN posee una capacidad de generalización aceptable, reduciendo efectivamente los falsos positivos y negativos. Por otra parte, el segundo método muestra una tendencia de mejora en la capacidad de evitar falsos negativos al disminuir el valor de umbral. En conclusión, los algoritmos propuestos ofrecen una solución robusta y eficiente para la detección de defectos en chips de papa.

Palabras clave: Visión artificial, Aprendizaje de máquina, Aprendizaje profundo, CNN, U-Net, Conteo por píxel.

Abstract

As a carbohydrate-rich and affordable food, potato crisps are a widely consumed snack around the world. As processed agricultural products, they are susceptible to defects that could affect consumer health. To meet quality standards, small companies still rely on inconsistent and laborious manual inspection controls. This study proposes a framework of computer vision and machine learning techniques to solve the detection problem. Two methods are presented: a Convolutional Neural Network (CNN) based classifier and a pixel counting method. The procedure carried out includes the acquisition and structuring of training data available in Kaggle, where the defect was simulated with random pigments created with a black marker on the surface of the object under study. The extraction and classification task of the first method is performed using a CNN architecture, where the pre-processing includes resizing, normalization, and noise reduction. The segmentation of the region of interest (ROI) of the whole area is performed using a semantic segmentation method (U-Net 1) developed with images and binary masks obtained from data enhancement, background removal, and thresholding. The second method is performed using a threshold and a percentage calculated from the pixel count of the binary masks generated by the U-Net 1 model and a second U-Net 2 model generated with processed data that includes manual labeling of the erroneous ROI and a transformation. The evaluation with two sets of test images shows that the CNN model has an acceptable generalization capability, effectively reducing false positives and negatives. On the other hand, the second method shows an improvement trend in the ability to avoid false negatives by lowering the threshold. In conclusion, the proposed algorithms provide a robust and efficient solution for defect detection in potato chips.

Keywords: Computer vision, Machine learning, Deep learning, Convolution Neural Network, U-Net, Pixel counting.

Introducción

Problema

Desde la década de los noventa hasta la actualidad los hábitos alimenticios del ser humano han estado en constante cambio. Según la Organización Mundial de la Salud (OMS) ha identificado como factores causantes: estilos de vida, caracteres sociales y económicos; hoy en día las personas requieren de alimentos de fácil manejo y consumo. Un alimento habitual que satisface temporalmente el hambre son los conocidos snacks industriales (chifles, galletas, papas chips), no obstante, desde el punto de vista nutricional no representan una fuente de energía, sin mencionar que por su alto contenido de grasa, azúcar, sal y conservantes pueden ser adictivos. En Ecuador, los snacks tienen un efecto inclusivo característico, la razón se debe a que los principales proveedores de materia prima son los pequeños productores que viven en zonas rurales, lo que brinda desarrollo económico a estos sectores vulnerables [1], específicamente en la sierra, la papa es uno de los principales cultivos tradicionales, siendo los responsables alrededor de 85,000 familias, sobre todo en la zona central (Cotopaxi, Tungurahua, Chimborazo) con 45% y en la zona norte (Carchi, Imbabura) con 45%, donde el cultivo es prácticamente todo el año.

Diariamente toda persona se encuentra expuesta a un sin número de microorganismos o toxinas, ya sea el aire que nos rodea, artículos de uso doméstico, incluso en productos de consumo. Con la evolución de los sistemas de inocuidad de los alimentos, se han generado investigaciones en donde se ha detectado la presencia de sustancias posiblemente nocivas que, si bien ingresaron por vías externas o fueron consecuencia de algún proceso durante su preparación (fritura, horneado u cocción)[2], podrían ser perjudiciales para la salud. A raíz de esto, la OMS en conjunto con la Organización para la Agricultura y la Alimentación (FAO) y la Agencia Nacional de Regulación, Control y Vigilancia Sanitaria (ARCSA) con el propósito de proteger la salud del consumidor han establecido estándares, regulaciones y sistemas de control calidad en los alimentos [3].

La visión artificial, una tecnología que emplea la recopilación y análisis de imágenes de escenas reales utilizando ordenadores y otros equipos para obtener información o controlar procesos, está jugando un papel cada vez más importante en la inspección visual automatizada de alimentos [4], empresas como PepsiCo posee un sistema de visión patentado encargado de buscar y descartar papas tipo chips defectuosas. Este tipo de sistemas, ofrecen precisión y repetibilidad en las mediciones sin contacto, garantizando la eliminación de aspectos como la subjetividad, el cansancio, la lentitud y los costos asociados a la inspección humana. Sin embargo, en las pequeñas empresas al carecer de estos sistemas, la fase de control de calidad aún se la realiza de forma manual, que si bien, cumple la función de inspección, son poco confiables debido a su naturaleza subjetiva [5].

Lo dicho hasta aquí supone que, si este proceso no es realizado de manera adecuada, se corre el riesgo de dejar pasar por alto algún defecto o elemento extraño, que ocasionaría que el producto final que llega a manos del consumidor pueda estar contaminado o no cumplir con los

estándares de calidad y por ende afectar a su salud, generando desconfianza, que se reflejaría en pérdidas económicas y sobre todo en el desprestigio de la empresa.

Objetivos

Objetivo general

Desarrollar un método para la detección de defectos en papas tipo hojuelas (chips) empleando visión artificial.

Objetivos específicos

- Determinar las características principales de las papas tipo chips.
- Aplicar técnicas de aprendizaje de máquina a fin de resolver el problema de detección.
- Validar el método planteado.

Justificación

La preocupación por la inocuidad de productos procesados se encuentra latente, eso se evidencia con las diferentes normas y estándares de calidad desarrolladas por diferentes entes de seguridad alimentaria [6, 7]. Esto ocasiona que las pequeñas empresas deban implementar métodos para mejorar la calidad de sus productos con fin de ser competitivas en el mercado de alimentos. El propósito de este proyecto consiste en reunir información relevante que permita desarrollar un sistema para la detección de defectos presentes en papas tipo chips mediante la visión por ordenador y técnicas de aprendizaje de máquina.

Entre los procesos industriales de alimentos, la fritura es un proceso multifuncional común, pero complejo. En el caso de las papas tipo chips se considera el paso más crítico, porque la calidad y seguridad del producto final intervienen muchas variables entre ellas: composición física y química del producto, tiempos y temperatura de cocción, oxidación del aceite, etc. Si este proceso falla, se lo puede visualizar en la fase de descarte manual, siendo esta, la última oportunidad para garantizar la calidad del producto final [8]. Ahora bien, el método propuesto podría ser aplicado para mejorar la eficiencia de detección de defectos que en empresas pequeñas son realizadas manualmente, eso sí, implementándolo dentro de una banda transportadora. Además, podría ser un método que permita ayudar a registrar las causas que conllevaron al resultado obtenido en diferentes situaciones, por ejemplo: diferentes tiempos y temperaturas de fritura.

Alcance

En este trabajo se presentará un algoritmo de visión artificial escrito en lenguaje de programación de alto nivel y software libre que sea capaz de reconocer al menos un tipo de defecto en papas tipo chips naturales utilizando técnicas de aprendizaje de máquina, de modo que, se iniciará identificando el procedimiento más adecuado para el procesamiento digital de imágenes. Para el desarrollo del modelo de aprendizaje de máquina se utilizará un conjunto de datos etiquetados disponible desde un repositorio; el 80 % de los datos serán destinados para el entrenamiento mientras que el 20 % para la validación, de esta manera se evitará el fenómeno de sobreajuste. Finalmente, con el fin de evaluar el rendimiento de predicción del algoritmo se hará uso de otro conjunto de datos reservados, los mismos que no tendrán ninguna relación con los datos que se dedicarán para la creación del modelo de aprendizaje de máquina.

Capítulo 1

Revisión literaria

Los alimentos de origen vegetal o animal provenientes del sector agrícola pueden ser consumidos de dos maneras: natural o ser sometidos a técnicas de procesamiento con el fin hacerlos aptos para el consumo humano o aumentar el tiempo de caducidad. Al momento de elegir un producto alimenticio, el consumidor de manera subjetiva lo realiza en función de diversos factores considerados como parámetros de calidad, como son: valor nutricional, tiempo de caducidad, precio, cantidad. Sin embargo, según [8, 9] definen que el parámetro que determina la aceptabilidad de un alimento son las características sensoriales (apariencia, color, textura, sabor, olor y aroma).

Las papas tipo hojuelas (chips) por sus características sensoriales como su textura crujiente, sabor salado, color característico y sensación grasa en la boca, forman parte de la dieta alimenticia de la mayoría de la población mundial. Dentro de la industria, existe dos maneras de proceso. El tradicional consiste en aplicar procesos de fritura a papas crudas previamente cortadas en rodajas finas (0.7 a 1.8 *mm*), adquiriendo su característica distintiva [8]. Por otra parte, las papas producidas o reestructuradas son el resultado de un proceso de moldeo a partir de una mezcla de harina de papa y posteriormente se fríe [10]. No obstante, este tipo de procesos enfrentan desafíos en términos de control de calidad, principalmente la detección de defectos (áreas quemadas, partes blandas, presencia de manchas, decoloraciones o inclusión de material extraño).

En este contexto, la visión artificial ha surgido como una tecnología prometedora que permite automatizar el proceso de inspección y mejorar la eficiencia y precisión en la detección de anomalías. Mediante el uso de algoritmos de procesamiento de imágenes y técnicas de aprendizaje automático, es posible analizar visualmente las chips de papa y detectar de manera rápida y precisa cualquier defecto o irregularidad [11, 12].

El presente capítulo proporciona una recopilación de información referente al procesamiento de imágenes y visión artificial aplicadas a la solución de problemas de detección de defectos y clasificación de alimentos. Aunque algunos estudios no se enfocan de manera específica en el tema abordado en este proyecto, suministran una base sólida para comprender las metodologías implicadas.

1.1. Estado del arte

1.1.1. Métodos de detección basados en análisis de color y texturas

En una investigación realizada en la Universidad Católica de Chile, se analizó el aspecto de chips de papas comerciales empleando métodos de análisis de imágenes en conjunto de visión por

ordenador para clasificar el estado de 4 grupos (ver Fig. 1.1). Además, mediante la adquisición de características poder modelar las preferencias visuales de los consumidores. A fin de segmentar la región de interés (ROI), se diseñó un filtro empleando un valor umbral de 60 en conjunto de un método de detección de bordes basado en LoG¹. Para el análisis de color se emplearon tres espacios de color: L*a*b, HSV y escala de grises; mientras que, mediante el uso de matrices de co-ocurrencia evaluadas en diferentes ángulos direccionales ($h = 0^\circ, 45^\circ, 90^\circ, 135^\circ$) y una distancia entre píxeles ($d = 1$), se adquirió las características de texturas (energía, entropía, contraste y homogeneidad). Cabe señalar que, las tasas de clasificación obtenidas utilizando el análisis discriminante lineal (LDA) en tres características texturales (entropía de a^* , entropía de V y energía de b^*) fue del 83.50 % en los tres primeros grupos de papas chips (A, B, C) pero un 44.50 % en el grupo D. Sin embargo, luego de implementar un algoritmo adicional para extraer características independientes de color y textura en imágenes de papas chips con manchas y/o defectos naturales, las tasas de clasificación mejoraron satisfactoriamente (ver Fig. 1.2), con un 95.80 % para los datos de entrenamiento y un 90.00 % para los datos de validación [14].



Figura 1.1: Imágenes de papas chips categorizadas en 4 grupos [14].

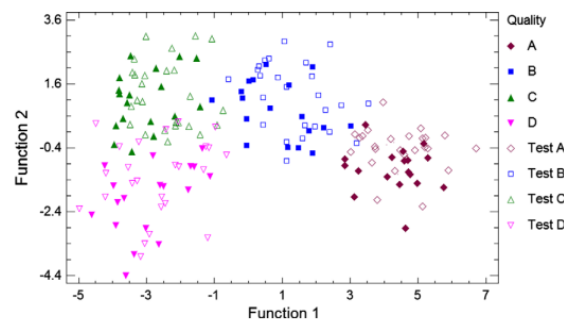


Figura 1.2: Clasificación de las cuatro categorías de calidad de las papas fritas para los datos de entrenamiento (símbolos llenos) y los datos de prueba (símbolos vacíos) utilizando LDA como criterio de selección [14].

Por otra parte, en [15] se enfocaron en el diseño de un sistema de visión artificial capaz de clasificar las papas fritas lisas y onduladas en categorías en base a su color. A diferencia de [14], los autores utilizaron un modelo de regresión lineal a fin de determinar la correlación entre las evaluaciones sensoriales realizada por un panel de evaluadores a 100 papas fritas comerciales con las mediciones predichas automáticamente en el espacio de color L^*, a^*, b^* . Finalmente, mediante un procedimiento de validación cruzada se evaluó los modelos (ver Fig. 1.3), obteniendo un error del 4 % aproximadamente para las papas lisas (sin ondulaciones) y del 7 % para las papas con ondulaciones.

¹Operador Laplaciano de Gauss (LoG): Método que combina la función Gaussiana con el operador Laplaciano a fin de reducir el ruido de la imagen y detectar los bordes en un solo procedimiento de convolución [13].

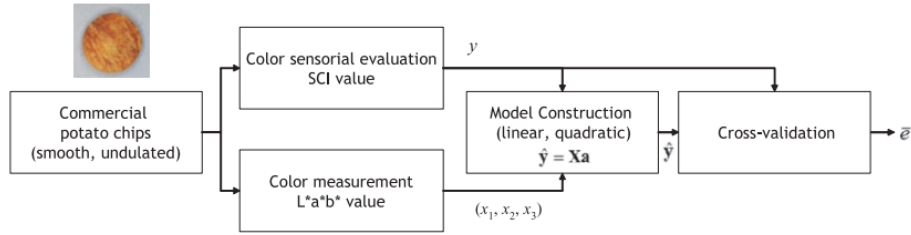


Figura 1.3: Diagrama esquemático del proceso seguido para la clasificación automática de papas chips [15].

En cambio, en [16] se propuso un método eficaz basado en el procesamiento de imágenes para la identificación de una sustancia potencialmente carcinogénica resultado del método de cocción de las papas fritas o chips de papa. Mediante el uso técnicas de conversión de espacios de color (RGB a HSI), umbralización y operaciones morfológicas fue posible segmentar ROI en escala de grises. Demostraron que usar la Transformada Discreta de Wavelet (DWT) para descomponer iterativamente una imagen en diferentes niveles y direcciones, separando sus componentes de alta y baja frecuencia, permite una representación más clara de la información. Además, facilitan la extracción de características discriminatorias que permitan determinar un valor de umbral adecuado para identificar la sustancia dicha, que en este caso fue de -0.19 (ver Fig. 1.4c). Los resultados obtenidos fueron prometedores, alcanzando una precisión del 90%.

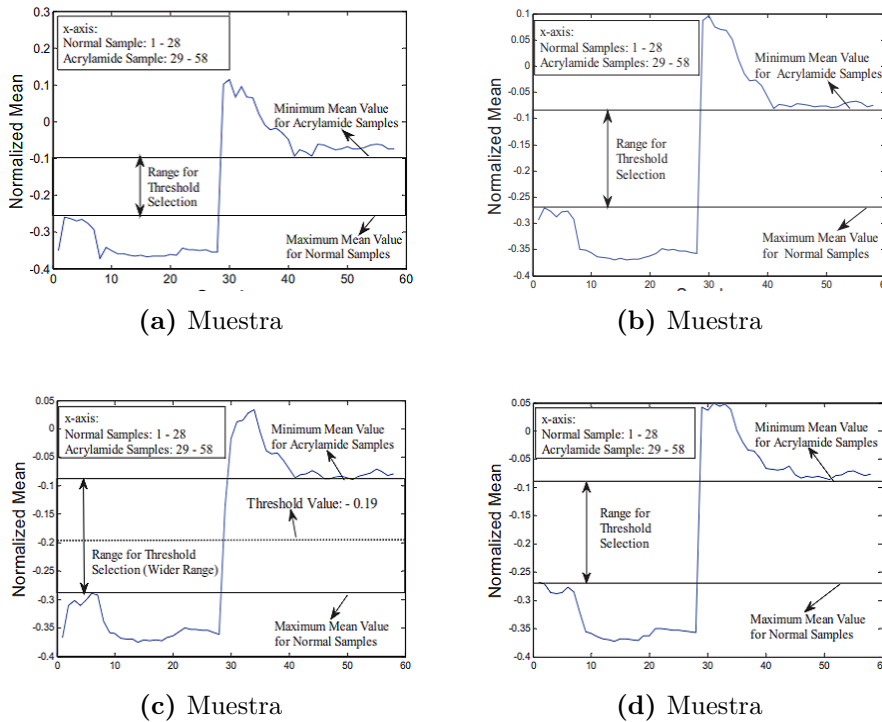


Figura 1.4: Gráfica de la media de los componentes wavelet. a) Media - componentes wavelet en primer nivel dirección horizontal, b) Media - componentes wavelet en primer nivel dirección vertical, c) Media - componentes wavelet en primer nivel dirección diagonal, d) Media - componentes wavelet en segundo nivel dirección horizontal [16].

1.1.2. Métodos de detección basados en aprendizaje automático

Dentro de este ámbito se han realizado varios intentos relacionados con la evaluación de la calidad de los alimentos. En [17] se desarrolló un método basado en visión por computador y aprendizaje supervisado para identificar esta misma sustancia. Una vez se extrajo la ROI mediante un procedimiento similar presentado en [16], se analizó la distribución del histograma en imágenes en escala de grises, observándose una diferencia de valores de intensidad de píxeles en las dos clases. En base a estas observaciones, se extrajo un vector de características estadísticas y de textura. Para mejorar la complejidad y rendimiento del modelo, el vector de características fue estandarizado mediante un método de normalización de puntuación Z y luego reducido mediante Análisis de Componentes Principales (PCA), con el fin de identificar los componentes principales con mayor varianza para el desarrollo del clasificador basado en un algoritmo de aprendizaje supervisado: Support Vector Machine (SVM), arrojando resultados satisfactorios (ver Fig. 1.5). De un conjunto de datos de 150 imágenes, 100 fueron destinadas para entrenar el modelo y 50 para realizar pruebas. Los resultados fueron prometedores, alcanzando una precisión del 94 % y una sensibilidad del 96 %.

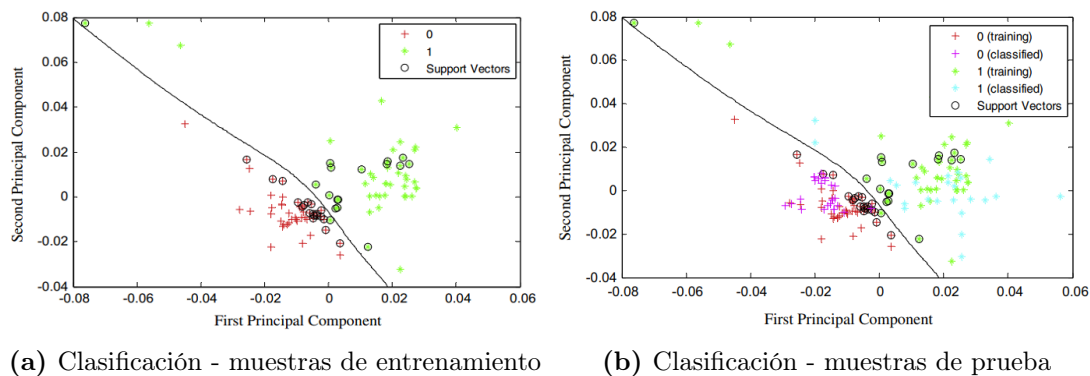


Figura 1.5: Gráficas del clasificador SVM empleando un kernel no lineal de “Polinomio - orden 3” considerando las características extraídas, con un etiquetado de: 0 para la clase positiva, 1 para la clase negativa [17].

Por su parte, en un estudio realizado en [18] se eligió trabajar en el espacio RGB para extraer la ROI. De modo que, se desarrolló un algoritmo basado en la técnica de crecimiento de región a partir del canal azul de la imagen de muestra; se aplicó operaciones morfológicas para suavizar los contornos de la máscara generada. Se consideró los espacios RGB y HSI para el análisis y extracción de 6 características estadísticas discriminatorias que sirvan para discernir entre muestras saludables y con presencia de anomalías. Las mismas que fueron base para desarrollar un clasificador de redes neuronales artificiales (ANN). De un conjunto de 84 muestras de imágenes: 70 % fue destinado para el entrenamiento y el 30 % para pruebas. Mediante una matriz de confusión se proporcionó una visión detallada de la clasificación de instancias en cada categoría. Finalmente, se calcularon las tasas de verdaderos y falsos positivos para ser representados a través de una curva ROC (ver Fig. 1.6), demostrando un rendimiento aproximado del 99 %.

Es evidente que un paso fundamental para obtener resultados satisfactorio es la segmentación adecuada de la ROI. En este contexto, en [11] se exploró la integración del algoritmo de agrupamiento K-means y operaciones morfológicas con el fin obtener máscaras binarias pertenecientes a la ROI de imágenes RGB. Debido a la presencia de cambios visuales en las muestras, se optó por extraer características estadísticas en los canales “L” y “a”. De modo que, se realizó una conversión de espacios de color: RGB a XYZ y de XYZ a Lab; luego, los canales se multiplicaron con las máscaras binarias a fin de considerar solo los píxeles pertenecientes a la ROI (ver Fig. 1.7)

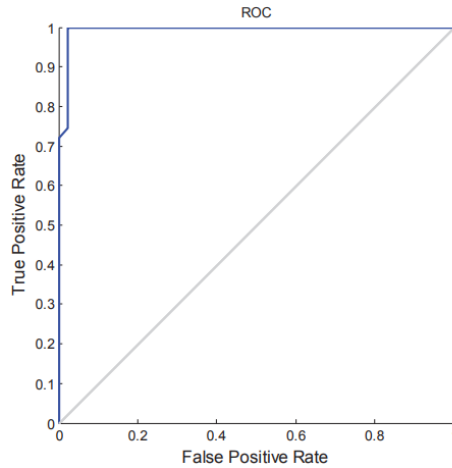


Figura 1.6: Curva ROC del clasificador basado en una ANN [18].

previo a la obtención del vector de características. El vector de características discriminatorias fue empleado como entrada para el desarrollo del modelo clasificador basado en Random Forest. De un grupo de 80 imágenes: 40 correspondientes a la clase normal y 40 la clase defectuosa; el 70 % y 30 % fue destinado para fines de entrenamiento y pruebas respectivamente. Los resultados fueron satisfactorios, alcanzando una precisión del 95.83 %, con un tiempo computacional promedio de 0.48 s por imagen.

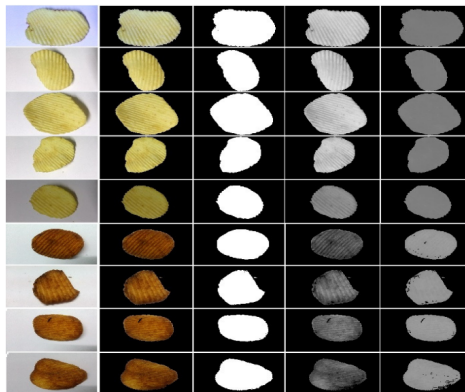


Figura 1.7: Imágenes obtenidas aplicando el algoritmo propuesto [11].

1.1.3. Métodos de detección basados en aprendizaje profundo

Otro requerimiento fundamental que se puede identificar es la extracción de características, entre más discriminatorias sean mejor es el resultado. Sin embargo, la mayoría de las técnicas mencionadas demandan de una inversión considerable de tiempo puesto que requieren del uso de procesos estadísticos manuales. Uno de los métodos que han proporcionado resultados satisfactorios son los modelos basados en Redes Neuronales Convolucionales (CNN). Dentro del contexto de la detección de anomalías en chips de papa, en [12] se propone un método integral que reducen tanto el tiempo de procesamiento de imágenes como la necesidad de procesos estadísticos manuales para extraer características discriminatorias. Mediante el uso de una arquitectura de red U-net se desarrolló un modelo entrenado que permita segmentar la ROI automáticamente (ver Fig. 1.8). Los resultados fueron satisfactorios, con un índice de Jaccard promedio de 0.9792. Para la tarea de extracción y clasificación, los autores proponen dos tipos de CNN: un modelo prototipo de CNN sin transferencia de aprendizaje y un enfoque de transferencia de aprendizaje

empleando modelos preentrenados como VGG16, DenseNet121, ResNet50 e InceptionV3 para extraer las características en conjunto de un algoritmo de aprendizaje supervisado basado en SVM para la clasificación. Se identificó que el modelo DenseNet121 obtuvo mejor rendimiento en extraer características (ver Fig. 1.9b). A través de operaciones como rotación y volteo, el conjunto de datos aumentó a 27 378 imágenes, luego se dividieron en 5 subconjuntos. Para el desarrollo del modelo se aplicó la técnica de validación cruzada (4 subconjuntos se utilizaron para el entrenamiento, mientras que el restante fue para pruebas) a fin de obtener un modelo robusto y evitar el sobre-ajuste. Los resultados fueron prometedores, con un *F1-score* promedio de 0.9251 y un tiempo computacional promedio de 2.11 ms por imagen para el modelo sin transferencia de aprendizaje; por su parte, el modelo con transferencia de aprendizaje con mejores métricas de precisión y eficiencia computacional fue el DenseNet, con un *F1-score* de 0.9644 y un tiempo computacional de 7.83 ms.

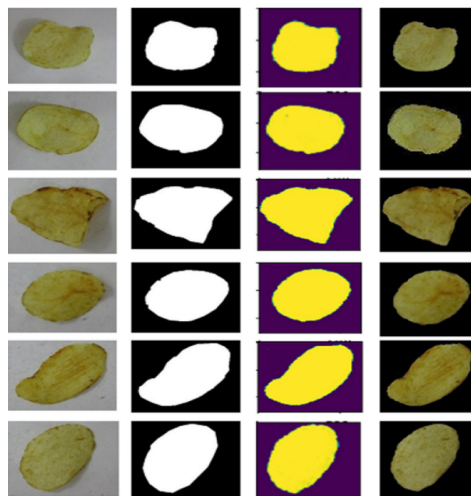


Figura 1.8: Imágenes obtenidas aplicando el modelo U-net [12].

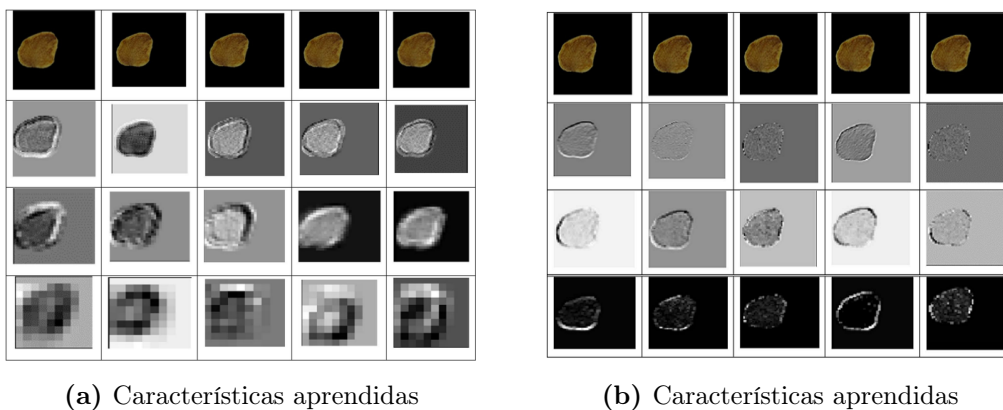
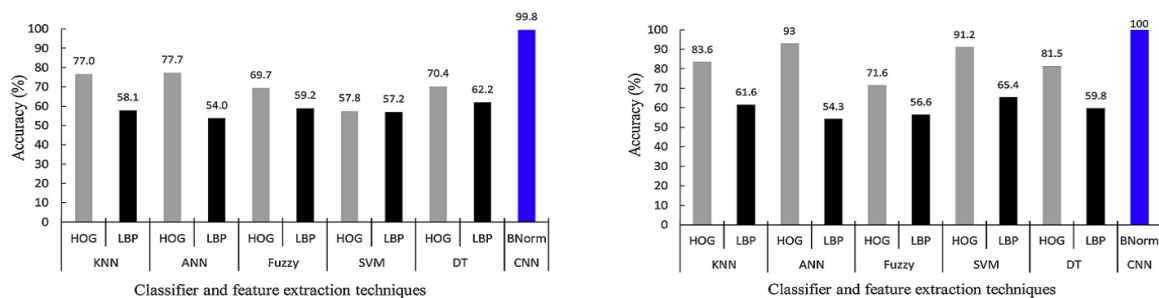


Figura 1.9: Visualización de los mapas de características aprendidas del modelo prototipo y del basado en DenseNet respectivamente [12].

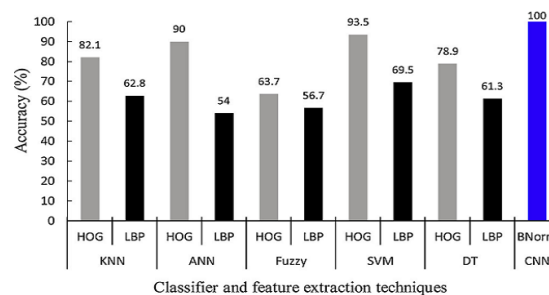
Además de su aplicación en la producción de chips de papa, los modelos de aprendizaje profundo han demostrado ser eficientes en abordar varios desafíos relacionados con productos alimenticios. En [19] se llevó a cabo un estudio para detectar defectos en limones mediante el entrenamiento de una CNN. A través de un proceso de aumento de datos (espejo, rotación, manipulación de color) se generaron nuevas imágenes; así, de conjunto inicial de 341 muestras etiquetadas, se amplió a un total de 5456 imágenes (2960 pertenecientes a la clase saludable y 2496 a la clase defectuosa). Luego, fueron redimensionadas en tres categorías: (16 × 16), (32

$\times 32$) y (64×64) píxeles. Este submuestreo permite disminuir la cantidad de parámetros y conexiones de la CNN. Para cada una de las tres categorías se ha utilizado una configuración distinta de la CNN; no obstante, implementaron una capa de agrupación estocástica basada en la dispersión y capas de normalización por lotes con el fin de mejorar el desempeño de todas las configuraciones. Del conjunto total de datos, el 70 % se utilizó para entrenar el modelo y el 30 % para validar. El modelo propuesto fue comparado con otros métodos de clasificación: k-vecinos más cercanos (KNN), redes neuronales artificiales(ANN), lógica difusa(Fuzzy), máquinas de vectores de soporte(SVM) y árboles de decisión (DT), que incluyen el uso de histograma de gradientes orientados (HOG) y patrones binarios locales (LBP). Los resultados demostraron una precisión aproximada al 100 % en el modelo propuesto en comparación con los métodos mencionados (ver Fig. 1.10).



(a) Configuración 1.

(b) Configuración 2.



(c) Configuración 3.

Figura 1.10: Precisión de clasificación de los modelos entrenados con tamaños de imagen: a) 16 X 16 píxeles, b) 32 X 32 píxeles, c) 64 X 64 píxeles [19].

En cambio, en [20] se construyó un CNN basado en AlexNet. Para extraer la ROI en imágenes hiper-espectrales de manzana, emplearon un método de segmentación por umbrales, denominado como Otsu; además, para ampliar el conjunto de datos, añadieron ruido blanco gaussiano con una relación señal-ruido variable. Con un total de 6144 imágenes pertenecientes a cuatro tipos de residuos de pesticidas presentes en manzanas, los resultados de detección mostraron que la tasa de reconocimiento promedio para imágenes de una sola banda fue del 95.35 % y, la tasa de detección verdadera positiva para el conjunto de pruebas fue del 99.09 %.

Un proceso similar fue desarrollado en [21] , en donde se propuso un sistema que permita ayudar a agricultores a identificar zanahorias frescas y no saludables. A un total de 1063 imágenes recopiladas de diferentes repositorios se emplearon técnicas de filtro difuso para eliminar el ruido gaussiano y mejorar la calidad de datos de entrada para la red. Para la extracción de características y clasificación de las clases se empleó una arquitectura CNN (ver Fig. 1.12), logrando una precisión cercana al 99.8 %.

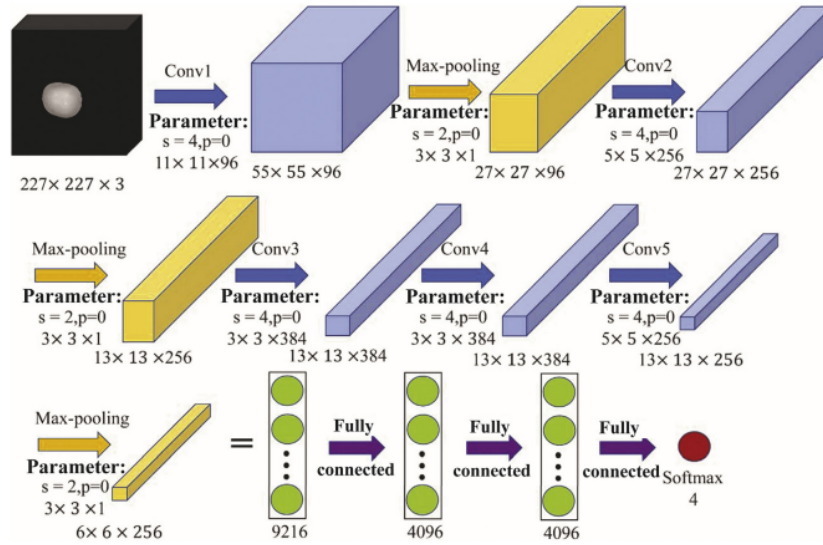


Figura 1.11: Estructura de la CNN [20].

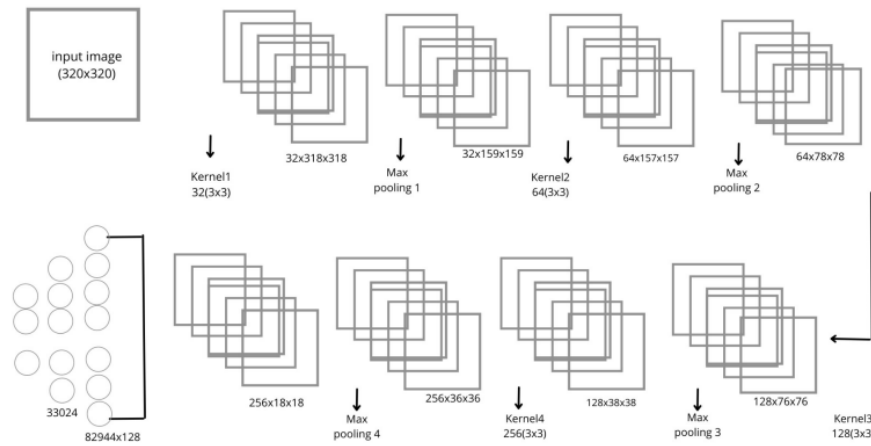


Figura 1.12: Estructura de la CNN + FCNN [21].

1.2. Tecnologías implicadas

Antes de empezar a adentrarse al desarrollo de este estudio, es fundamental conocer el significado de varios conceptos que aparecen a lo largo de este proyecto.

1.2.1. Inteligencia artificial

El término Inteligencia Artificial (IA) constituye un ámbito multidisciplinario en constante evolución que ejerce una influencia substancial en diversas ramas de investigación. Sin embargo, antes de definir la IA, es necesario comprender el significado de “inteligencia”. Según Sossa [22], la inteligencia se considera como una capacidad mental que implica diversas habilidades, tales como: el razonamiento, la planificación, la resolución de problemas, la comprensión de ideas abstractas, aprender de la experiencia, entre otros. Va más allá de solo memorizar datos, destrezas académicas; más bien, refleja una capacidad profunda de concebir el entorno, captar el significado de las cosas y ser objetivo en la toma de decisiones.

De acuerdo con Rich [23], la IA se refiere al estudio de cómo hacer que los ordenadores realicen

tareas que, por el momento, los humanos ejecutan mejor. No obstante, en el contexto de Sossa al definir el término “inteligencia” , la definición proporcionada por el Grupo de Expertos de Alto Nivel de la Unión Europea sobre Inteligencia Artificial resulta ser adecuada, la IA es una ciencia que permite desarrollar sistemas artificiales capaces de desempeñar tareas complejas dentro de un ambiente impredecible, a través del análisis de datos (estructurados o no estructurados) y percepción del entorno, determina la opción más acertada para alcanzar un objetivo propuesto [24].

1.2.2. Visión artificial

En principio, lo que diferencia a un sistema con IA es la capacidad de tomar decisiones basadas en la percepción de su entorno. En ese sentido, la visión artificial (VA) se enfoca en la percepción visual. En palabras de Elgendy, la VA es un campo de la IA que permite a un sistema percibir e interpretar el mundo visual a través del análisis de datos visuales (imágenes o videos) y tomar acciones en función de esa información. Además, menciona que en su mayoría, se encuentra compuesta por 2 partes: dispositivos de percepción y dispositivos de interpretación. Si bien el procedimiento a seguir varía según la propósito del sistema con VA, Elgendy sostiene que un sistema típico de visión siempre sigue una serie de pasos (ver Fig. 1.13) para procesar y comprender imágenes [25].

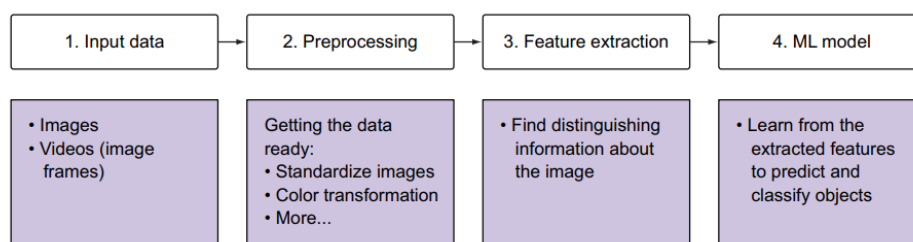


Figura 1.13: Procedimiento de un sistema de visión [25].

1.2.3. Aprendizaje automático

El aprendizaje automático o *machine learning*(ML) es un campo de la IA que permite a los ordenadores desarrollar la capacidad de resolver problemas sin intervención humana directa, incluso en situaciones inciertas que no fueron abordadas por el desarrollador. Esta capacidad es el resultado de dos aspectos complementarios: el refinamiento de la habilidad en la resolución de problemas, en donde el ordenador mejora sus habilidades con el tiempo; y, la adquisición de conocimiento a través de la experiencia, en donde el ordenador aumenta su capacidad de identificar patrones (funciones) y tomar decisiones a partir de los datos que recibe. Si bien, el modelo es reforzado por estos aspectos, durante su diseño, el ser humano es quien mediante el uso de métodos estadísticos atribuye un significado a esos datos, identificando las características más discriminatorias que sirvan como punto de partida para que el ordenador sea capaz de encontrar una relación por sí mismo [26].

1.2.4. Redes neuronales artificiales

Un término fundamental que suele aparecer dentro del ML son las conocidas redes neuronales artificiales o *artificial neural network* (ANN). Una ANN se define como un modelo computacional compuesto por múltiples nodos de neuronas artificiales que trabajan en conjunto, a fin de simular, en cierta medida, el procesamiento de información realizado por el cerebro humano. Es decir, una red neuronal es la representación de una función compleja resultante de la composición de funciones más simplificadas [27]. En un problema de clasificación de imágenes, este tipo de

arquitecturas requieren que sus datos de entrada sea necesariamente un vector con dimensiones $(1, n)$, a este proceso se lo denomina como aplanamiento de una imagen [25].

Neurona artificial

Kelleher menciona que [27], las unidades fundamentales que constituyen el cerebro humano, son las células nerviosas denominadas como neuronas. Por lo tanto, se puede afirmar que, una neurona artificial o *artificial neuron* (AN) es el elemento fundamental de una ANN que imita el funcionamiento de una neurona biológica; a su vez, representa una función que concatena varias entradas para generar un resultado. Este último, depende de las entradas y del ajuste de sus pesos asociados [26]. En la Fig. 1.14 se visualiza una analogía de los elementos que conforman una neurona biológica y una AN. Particularmente, se puede notar que este tipo de AN se compone de 2 etapas.

Una forma de expresar la etapa de integración se resume con la ecuación 1.1 que representa una transformación lineal de la entrada (resultado de la suma pondera de las entradas x_i por los pesos asociados w_{ij} y la adición de un bias b). Esta expresión, sin embargo, es contraproducente porque independientemente de la cantidad de capas que se añada a una ANN, el resultado siempre será una función lineal ineficaz en entornos reales donde existen algún tipo de no linealidad. Es fundamental que las ANN sean capaces de simular esas no linealidades. Es aquí, donde la etapa de activación aplica una transformación no lineal (ver Ec. 1.2) mediante el uso de las funciones de activación [26, 28].

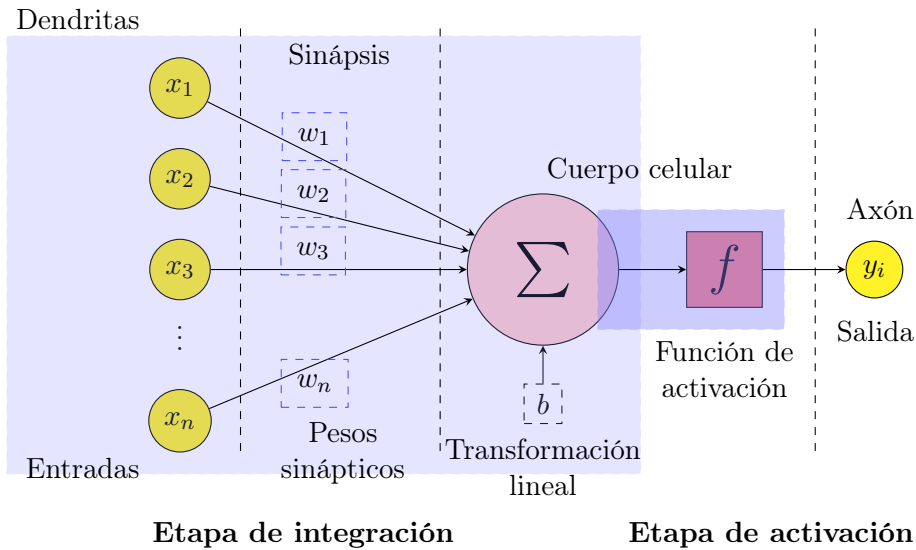


Figura 1.14: Diagrama de una AN.

$$y_j = b + \sum_{i=1}^n x_i w_{ij} \tag{1.1}$$

$$y_j = f \left(\sum_i w_{ij} x_i \right) \tag{1.2}$$

1.2.5. Funciones de activación

El propósito de una función de activación va mucho más allá de sólo introducir una no linealidad² en una ANN. En el supuesto caso de emplear el modelo resultado de la ecuación 1.1 a un problema de clasificación, el valor obtenido de y_i , a más de ser impredecible, no estaría sujeto a restricciones. Es por ello que, este tipo de funciones permiten ajustar estos valores de predicción a un umbral finito [25]. Existen una infinidad de funciones de activación, no obstante, las que comúnmente se utilizan dentro de la literatura son presentadas en la Tabla 1.1.

1.2.6. Aprendizaje profundo

El aprendizaje profundo, también conocido como *deep learning* (DL), es una rama del ML que se encarga de crear modelos de redes neuronales profundas capaces de tomar decisiones a partir del aprendizaje y reconocimiento de patrones en conjuntos masivos de datos complejos, estos patrones, en realidad son funciones representadas mediante redes neuronales [27]. De hecho, lo que destaca al DL de otras técnicas de ML, es la arquitectura que emplea para extraer las características de manera autónoma; de esa forma, son capaces no sólo de reconocer patrones dentro de un entorno específico, sino de aumentar el nivel de abstracción al trabajar con niveles de jerarquía, de esa manera, permite generar nuevas características a partir de características identificadas en el nivel anterior [26]. Un ejemplo de este tipo de tecnologías, son las conocidas redes neuronales convolucionales.

1.2.7. Redes Neuronales Convolucionales

Las redes neuronales convolucionales (CNN, por su acrónimo en ingles) [29, 30], son un tipo de arquitecturas *feedforward* que emplean métodos de convolución para el reconocimiento de patrones en conjuntos de datos visuales, representados como tensores. A diferencia de métodos convencionales [31, 32], las CNN mediante el uso de una arquitectura jerárquica de capas encargadas de aplicar convoluciones a los datos de entrada, permiten automatizar la tarea de extracción de características en varios niveles de abstracción [33].

Elgendy menciona que [25], el propósito de las capas de convolución, es extraer de manera gradual características complejas; en donde, las capas tempranas pueden aprender patrones básicos (líneas, bordes), las capas posteriores podrían extraer patrones un tanto complejos (figuras, etc.), incluso capas más profundas son capaces de detectar aspectos más sofisticados (partes del rostro, la llanta de un vehículo, etc.). Este tipo de redes (ver Fig. 1.15) están estructuradas principalmente por tres capas: capa convolucional, capa de pooling y capa totalmente conectada [34].

Capa convolucional

Los ordenadores interpretan las imágenes 2D como valores de 0 a 255 agrupados en una matriz (ver Fig. 1.16). Estos valores representan la intensidad de la luz en un punto dado, dentro del procesamiento de imágenes son denominados como píxeles [30].

Este tipo de capas son consideradas como el pilar fundamental de las CNN, debido a que, a través de una serie de filtros dinámicos, también conocidos bajo el término de *kernels*, aplican un proceso de convolución de forma horizontal y vertical sobre una imagen de entrada, de esa manera, se obtiene el resultado del producto punto entre el filtro y la entrada durante cada iteración [19]. A este procedimiento se le suma una función de activación no lineal, resultando

²La no linealidad permite que, a través de un mapeo no lineal entre la(s) entrada(s) y la salida, una ANN tenga la capacidad de aprender [27].

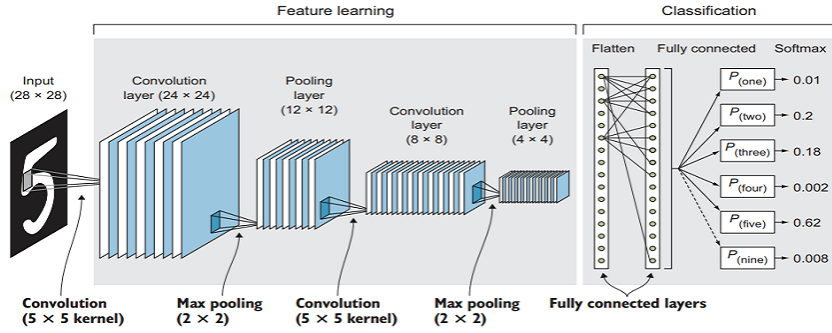


Figura 1.15: Capas básicas de una CNN [25].

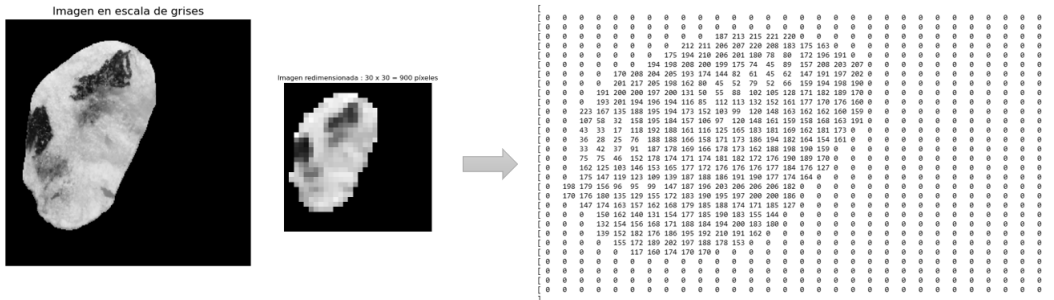


Figura 1.16: Valores de píxeles en una imagen de un solo canal de color [25].

en un conjunto de mapas de características o de activación que forman el volumen de salida [34]. Este volumen experimenta cambios proporcionales en relación con los hiper parámetros de diseño de la capa convolucional: tamaño de la imagen de entrada, dimensiones del filtro, el stride y padding. La ecuación 1.3 permite determinar el tamaño de salida; es decir, define el número de iteraciones en donde el filtro puede ser aplicado a la imagen de entrada [33].

$$\frac{W - F + 2P}{S} + 1 \quad (1.3)$$

Donde:

W = representa el alto (o ancho) de la imagen de entrada

F = es la dimensión del filtro

P = cantidad de padding

S = stride

Capa de Pooling

Permiten minimizar el coste computacional del proceso de convolución, a través de un muestreo, reduce la dimensionalidad y complejidad del volumen de salida constituido por el conjunto de mapas de características, a la vez, conservando los datos más relevantes. Dentro del contexto del procesamiento de imágenes, este procedimiento puede asemejarse a disminuir la resolución de la imagen [35]. Se encuentra configurado por dos hiperparámetros: la dimensión del filtro y el stride. Comúnmente se establece con un valor de 2, respectivamente [34]. Las capas de pooling tradicionales son: *max pooling* y *average pooling* [36].

Capa totalmente conectada (*Fully Connected*)

Esta capa permite la clasificación de imágenes de entrada, empleando como recurso las características extraídas durante el proceso de convolución. Para ello, transforma el volumen de salida en un vector unidimensional. Es similar a una ANN tradicional, en donde la última de sus capas emplea un función de activación *softmax* para generar un valor entre 0 y 1, determinando así, la probabilidad de pertenencia a una clase específica [36].

1.2.8. Métricas de rendimiento

Un paso crucial durante la elaboración de la CNN, es analizar el comportamiento del modelo. Esto implica en seccionar el conjunto de datos en tres grupos: entrenamiento, validación y prueba; además, de emplear una matriz de confusión para obtener varias métricas de rendimiento y cuantificar su eficiencia [37]. Las más utilizadas para problemas de clasificación de productos alimenticios [36, 38, 21, 39] están: *accuracy*, *precisión*, *recall*, *especificidad*, y el *F1-score*.

1.2.9. Python

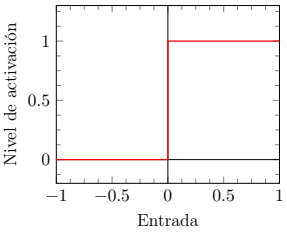
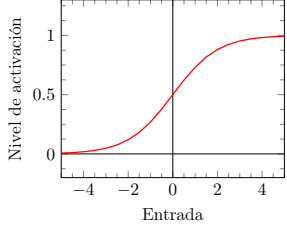
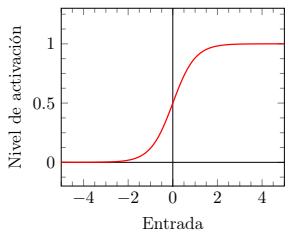
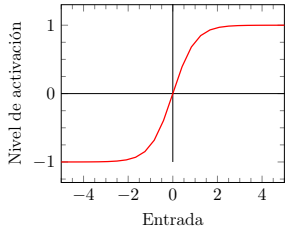
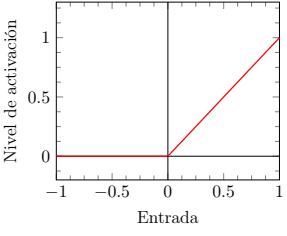
La orientación de Python hacia la legibilidad del código, lo ha convertido en uno de los lenguajes de enseñanza en cursos introductorios [40]. Debido a su sintaxis y tipado simple, Python es considerado un lenguaje de programación intuitivo, robusto y de fácil aprendizaje, y, por ende, con la opción de reciclar y modificar códigos sin introducir errores. Estas características permiten que la programación orientada a objetos (POO) sea notoriamente fácil de utilizar [41].

Al ser un lenguaje de código abierto y legible, posee una infinidad de librerías externas que permiten el desarrollo de proyectos. Además, existe un mayor soporte por parte de la comunidad de desarrolladores. Si de codificación se trata, Python emplea hasta una quinta parte de código requerido en otros lenguajes POO, posicionándolo en el segundo lugar después de Ruby. Finalmente, la versatilidad en manipular y vincular múltiples estructuras de datos; la compatibilidad de trabajar en varios entornos de desarrollo integrado (IDE), por ejemplo: Visual Studio Code, PyCharm, entre otros; poseer frameworks como Keras [42], TensorFlow [43] y PyTorch [28] se destaca como la primera opción para llevar a cabo ciencia computacional. Incluso, para desarrolladores que empiezan a adentrarse a este ámbito [40].

1.3. Recapitulación del estado del arte

En la Tabla 1.2 se puede destacar las características principales de los trabajos previos analizados, las cuales permiten tener una visión más general de los temas.

Tabla 1.1: Funciones de activación

Función de activación	Descripción	Gráfica	Ecuación
Función escalón [step function]	Produce una salida binaria de 0 a 1. En su mayoría son utilizadas en problemas de clasificación binaria, permitiendo dar un valor discreto.		$f(z) = \begin{cases} 1 & \text{si } z \geq 0 \\ 0 & \text{si } z < 0 \end{cases}$
Función logística [sigmoid]	Reduce todos los valores a una probabilidad entre 0 y 1, lo que disminuye los valores extremos o atípicos en los datos. Normalmente se utiliza para clasificar dos clases.		$f_{\text{logistic}}(z) = \frac{1}{1 + e^{-z}}$
Función Softmax	Una generalización de la función sigmoid. Se utiliza para obtener probabilidades de clasificación cuando tenemos más de dos clases.		$\sigma(z_i) = \frac{e^{z_i}}{\sum_{j=1}^K e^{z_j}}$
Tangente hiperbólica	Comprime todos los valores al rango de -1 a 1. El tangente hiperbólico casi siempre funciona mejor que la función sigmoid en capas ocultas.		$\tanh(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}}$
Unidad Rectificada Uniforme [RELU]	Activa un nodo solo si la entrada es mayor que cero. Siempre recomendado para capas ocultas. Mejor que el tangente hiperbólico (tanh).		$f_{\text{relu}}(z) = \max(0, z)$

Nota: Todos los datos obtenidos fueron extraídos de [25, 26].

Tabla 1.2: Tabla comparativa de los diferentes trabajos literarios relacionados al presente proyecto

Nro	Alimento	Propósito	Nro. img. entrena- miento	Ambiente controlado	Criterio de clasificación	Resultado	Cálculo de área defec- tuosa	Modelo usado
1	chips de papa: pálidas, ligeramente oscuras, con manchas marrones, con defectos naturales	Clasificación y modelar la preferencia del consumidor	72	Sí	Características de color y textura	95.80 % para los datos de entrenamiento y 90.00 % para los datos de validación.	No	ADL
2	chips de papa: lisas, onduladas	Clasificación mediante sistema de visión artificial	100	Sí	Características de color	error del 4.00 % para papas fritas lisas y del 7.00 % para onduladas	No	Modelo de regresión lineal
3	chips de papa	Identificación de una sustancia potencialmente carcinogénica	58	Sí	Umbral identificado	Precisión: 90.00 %	No	Procesamiento de imagen en el dominio Wavelet
4	chips de papa	Clasificación mediante visión artificial	150	Sí	Vector de características y textura	Precisión: 94.00 % y sensibilidad: 96.00 %	No	SVM
5	chips de papa	Detección automática de sustancias perjudiciales	84	Sí	Características estadísticas discriminatorias	Rendimiento del 99 %	No	ANN
6	chips de papa	Identificación automática de sustancias tóxicas	80	Sí	Características discriminatorias, desviación estandar y momento	Precisión: 95.83 %	No	Random Forest
7	chips de papa	Detección automática de sustancias perjudiciales por DL	27 378	Sí	Patrones aprendidos	F1 - score : 0.9251; 0.9644	No	CNN sin transferencia y con transferencia.
8	limón agrio	Clasificación y detección de defectos	5456	Sí	Patrones aprendidos	Precisión: 100.00 %	No	CNN
9	manzanas	Detección de residuos de pesticidas	12 288	Sí	Patrones aprendidos	Precisión: 99.09 %	No	CNN
10	zanahorias	Detección y clasificación de enfermedades	1063	Sí	Patrones aprendidos	Precisión: 99.80 %	No	CNN

Nota: Todos los datos obtenidos fueron extraídos de [14, 15, 16, 17, 18, 11, 12, 19, 20, 21]

1.4. Propuesta

La literatura sobre el procesamiento de imágenes y visión artificial ha demostrado que, aprovechar las tecnologías de aprendizaje de máquina en problemas como: la detección de defectos y la clasificación de imágenes, pueden lograr resultados satisfactorios. La Tabla 1.2 muestra que la mayoría de las investigaciones pueden alcanzar más del 90% de precisión con diversas metodologías.

Sin embargo, todavía hay vacíos identificados en la literatura que pueden ser la dirección de investigación de este estudio. Existen estudios que permiten determinar el porcentaje de área defectuosa en un alimento. No obstante, no han sido empleados a productos procesados como son las chips de papa artesanales. Es importante averiguar cómo se comporta la visión por ordenador en conjunto de tecnologías de aprendizaje de máquina con este tipo de alimentos procesados.

En base a los vacíos descritos, este estudio pretende desarrollar un algoritmo para la detección de defectos en chips de papa, para lo cual se hace uso de las ventajas de una red neuronal convolucional a fin de resolver el problema. Además, se propone un método de conteo de píxeles para calcular el porcentaje de defecto presente.

Capítulo 2

Desarrollo

En este capítulo se describe el desarrollo del modelo de detección de defectos en chips de papa a través de la clasificación entre dos clases: defectuosas y no defectuosas. En la sección 2.1 se proporciona una visión detallada del método propuesto basado en la metodología (ver Fig. 2.1) llevada a cabo en [44]. Por otra parte, para un manejo, representación visual de los datos y desarrollo del clasificador basado en CNN, se toma como referencia el script de [45] alojado en Kaggle.

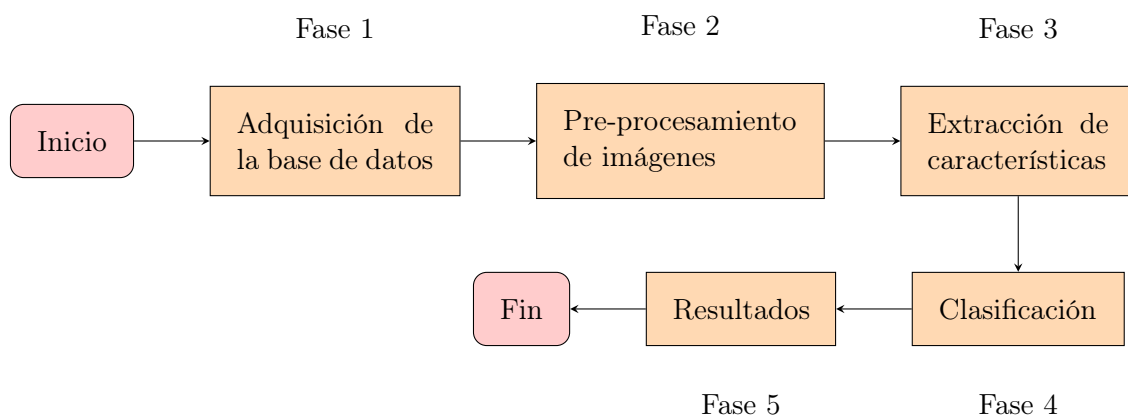


Figura 2.1: Metodología en visión artificial [44].

2.1. Arquitectura del método propuesto

Tomando en cuenta las necesidades y recursos disponibles para este estudio, se opta por desarrollar una metodología basada en la Fig. 2.1, dando como resultado el proceso que se muestra en la Fig. 2.2. El procedimiento que se lleva a cabo incluye: la adquisición de datos, el preprocesamiento de imágenes, el desarrollo de un modelo basado en la arquitectura U-Net para la segmentación automática a nivel de píxel de la región de interés (ROI), creación de la CNN para extraer características y clasificar las chips de papa (normales o defectuosas), el desarrollo de un segundo modelo U-Net para segmentar la ROI defectuosa. Luego, se introduce un método que permita compilar los tres modelos en conjuntos para la predicción de las imágenes de prueba a través de la CNN. Para el método por conteo de píxeles basado en el cálculo del porcentaje del área defectuosa, las predicciones pueden ser almacenadas en 2 directorios ("*Directorio_0*" para la clase defectuosa, "*Directorio_1*" para la clase sin defecto).

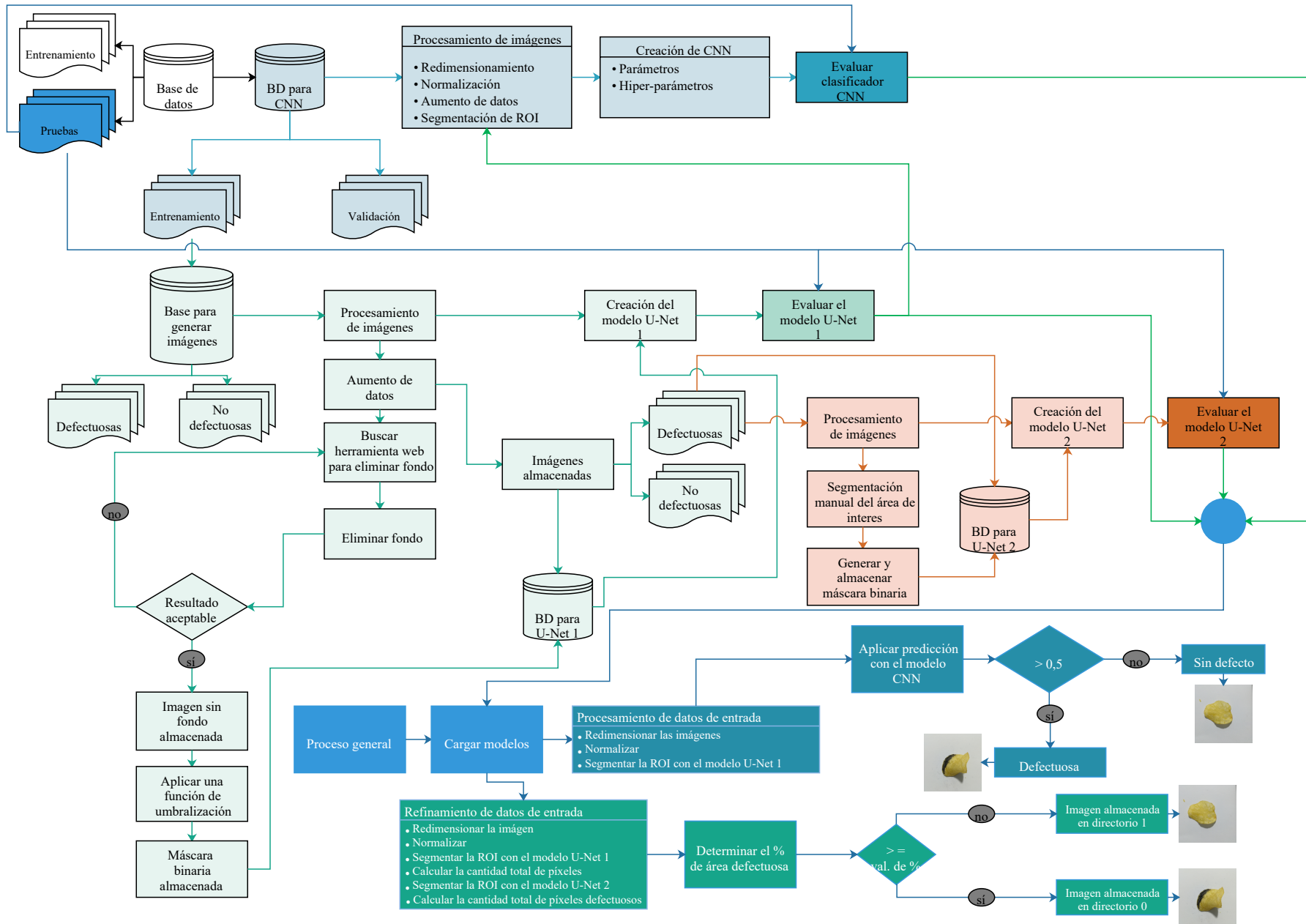


Figura 2.2: Metodología propuesta

2.2. Adquisición del conjunto de datos

El proceso de detección de defectos en chips de papa inicia con la adquisición del conjunto de imágenes. Para ello, la búsqueda se lleva a cabo en un repositorio gratuito llamado Kaggle. Hasta el momento solo se encuentran disponibles 2 conjuntos de datos. De modo que, la selección de datos se la hace tomando en cuenta el fin de este estudio (la detección de al menos un defecto presente en las chips de papa).

La base de datos seleccionada [46] posee 961 imágenes RGB capturadas con una cámara Hawey PRA-LA1 de 12 MP con una resolución de 2796 x 2796 píxeles. El formato en el que se almacenaron fue .jpg. En este caso, el autor de la base de datos, ha optado por señalar con un marcador negro una parte de la superficie de un grupo de chips de papa, con el fin de simular la presencia de un defecto. Además, comparten un mismo fondo y están divididas en 2 clases: “*Defective*” y “*Non-Defective*”. Por otra parte, se empleó como fuente de iluminación la luz del día, dando como resultado la aparición de sombras en las muestras (ver Fig. 2.3). Este ruido generado es tratado más adelante ¹.

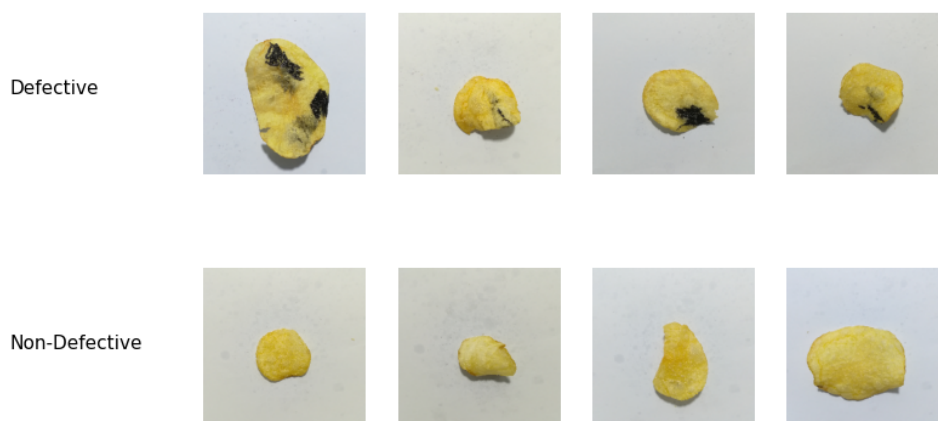


Figura 2.3: Muestras aleatorias por clase.

Las imágenes se encuentran distribuidas en dos carpetas² denominadas como “*Train*” y “*Test*”, las mismas que serán empleadas para el entrenamiento y testeo del modelo respectivamente. Antes de empezar a desarrollar el modelo, es necesario dividir los datos en rango de: 70 – 80 % para el entrenamiento y 20 – 30 % para validar o testear; de esta manera se garantiza en obtener un modelo eficiente, pero sobre todo, se evita el fenómeno de sobre-ajuste. Por lo tanto, se genera una tercera carpeta denominada “*Validation*” en la cual, mediante el uso de programación, se almacena aleatoriamente el 20 % de los datos contenidos dentro de la carpeta “*Train*”, que ahora contendrá el 80 % restante (ver Fig. 2.4).

2.3. Procesamiento y segmentación de la región de interés (ROI)

Separar el fondo de un imagen resulta ser una labor complicada, su dificultad está en gran medida condicionada en el estado de la ROI y del fondo. Como se mencionó en la sección 2.2, las muestras con imágenes de chips de papa poseen ruidos: presencia de sombras causadas por una

¹Véase en la sección 2.3.1

²Debido a que, la mayoría de la documentación y recursos utilizados dentro de este estudio están en inglés, se ha decidido mantener los nombres por defecto de los directorios.

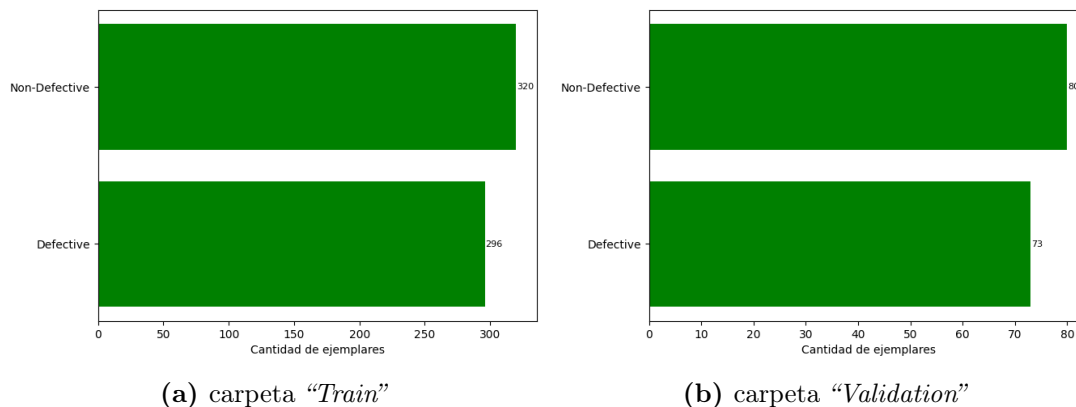


Figura 2.4: Número de muestras de imágenes distribuidas en sus respectivas carpetas y clases. a) "Train", b) "Validation".

fuentes de luz inapropiada, iluminación inconsistente (variación en el contraste y la intensidad de los píxeles) y distorsión de los colores. Esto ocasiona un retraso en la separación del fondo y la ROI. A su vez, generarían resultados no deseados, entre ellos: que la arquitectura CNN podría considerar las sombras como un defecto presente en las chips de papa de la clase normal y clasificarlas como defectuosas³. Para solventar este error, se opta por emplear la segmentación semántica, que ha arrojado resultados satisfactorios [12].

2.3.1. Creación de modelo U-Net 1

En un principio, este tipo de arquitecturas fue diseñado para abordar problemas de segmentación en imágenes médicas puesto que, obtener imágenes de este tipo, representan un gasto significativo debido al costo de los equipos que se emplean para realizar este proceso. Como consecuencia, el conjunto de datos es relativamente escaso. A pesar de ello, con un limitado grupo de datos, se ha obtenido buenos resultados [47]. Para el desarrollo del modelo de segmentación semántica, se decide tomar como referencia la arquitectura presentada en [12], en donde ha sido evaluada en productos procesados. Adicionalmente, se toma como base el script de [48] alojado en Github, en su trabajo, el autor implementa un modelo U-Net utilizando el framework PyTorch.

Creación del conjunto de datos

En [47] establece que, para entrenar este modelo se requieren dos tipos de datos de entrada: el conjunto de imágenes y sus respectivos mapas de segmentación o máscaras binarias⁴. Del directorio "Train", se elige y almacena manualmente en una carpeta denominada "Data_random" un grupo de 40 imágenes (20 de la clase normal y 20 de la clase defectuosa) en donde se evidencia significativamente la presencia de sombras y chips de papa con geometrías inusuales (ver Fig. 2.5).

Ahora bien, a través de una clase denominada como ImageDataGenerator se realiza un aumento de datos introduciendo varias transformaciones, como son: normalizado de píxeles, rotaciones y desplazamientos aleatorios, efectos espejo y un relleno de píxeles. Además, se realiza un submuestreo moderado (de 2796 x 2796 a 1280 x 1280 píxeles) para disminuir el uso de recursos computacionales y conservar detalles de las imágenes. Al finalizar el proceso se obtiene una

³Este error se evidencia en la sección 3.2.7

⁴Máscara binaria: es el resultado de convertir una imagen a color o escala de grises en una imagen binaria, donde los píxeles que superan un umbral establecido, son asignados con un valor de 1 (blanco) y los que no, con un valor de 0 [49].



Figura 2.5: Muestras con ruidos elegidas.

variedad de imágenes que permiten mejorar la capacidad del modelo para generalizar nuevos datos. El total de imágenes generadas fue de 320 (ver Fig. 2.6), es decir, por cada imagen original se obtuvo 8 imágenes más, las mismas que fueron almacenadas en un directorio denominado “*Data_Augmentation*”. En este punto, aunque se podría añadir las imágenes originales al lote generado y así aumentar el conjunto de entrenamiento, se decide descartarlas.

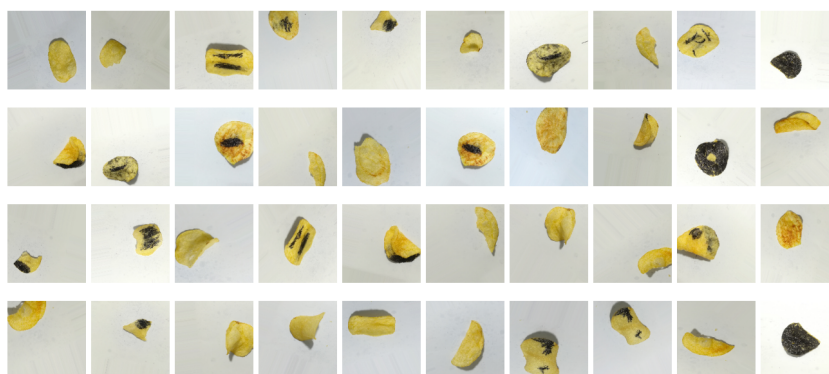


Figura 2.6: Muestras generadas.

Obtener las máscaras binarias implica la integración de varios aspectos: cantidad significativa de tiempo y el uso de software (Photoshop, Paint.NET, entre otros) o librerías como OpenCV⁵. En este caso, se decide usar la herramienta PhotoRoom⁶ disponible en la web para eliminar el fondo, no obstante, posee una restricción de resolución máxima de 1280 x 1280 píxeles en las imágenes de descarga. Antes de almacenar las imágenes, se verifica que el proceso de eliminación del fondo sea aceptable. Si bien, durante el proceso de aumento de datos se realizó una disminución del tamaño en cada imagen, aún conservan una resolución que permite a la herramienta web eliminar el fondo sin problemas. La Fig. 2.7 representa una muestra del resultado obtenido que han sido almacenados en una carpeta llamada “*Data_Nground*”.

Finalizado este procedimiento, es momento de generar las máscaras binarias aplicando las herramientas de segmentación por umbralización de OpenCV. Para lo cual, se desarrolla una función que permita recorrer todo el lote de imágenes, cargarlas, convertirlas a escala de grises, establecer un valor de umbral, aplicarlo a cada imagen y obtener las máscaras binarias. Por

⁵OpenCV: es una librería gratuita especializada en visión artificial. Ofrece herramientas de procesamiento de imágenes, entre ellas: segmentación de objetos, detección de bordes, transformaciones morfológicas [50].

⁶PhotoRoom: es un software gratuito e intuitivo de edición de imágenes desarrollado con aprendizaje profundo [51].

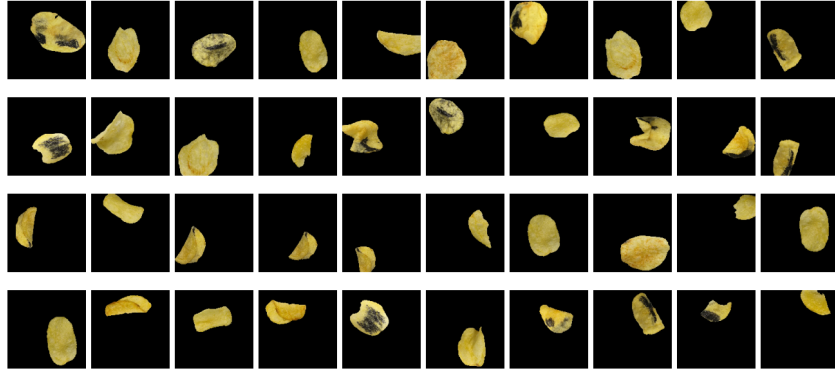


Figura 2.7: Muestras generadas sin fondo.

último, almacenarlas en una carpeta denominada como “*Data_mask*”. A diferencia de las metodologías presentadas en la literatura, la particularidad de este proceso radica en omitir el uso de operadores morfológicos para eliminar posibles ruidos que se generen durante la umbralización y en obtener de manera automática un lote de máscaras binarias con características aceptables. La Tabla 2.1 engloba el algoritmo propuesto para generar un conjunto de imágenes con sus respectivas máscaras binarias. Un muestra de los resultados obtenidos puede ser visualizada en la Fig. 2.8.

Tabla 2.1: Procedimiento para generar máscaras binarias.

Algoritmo : //Adquirir lote de imágenes con sus máscaras binarias//

Paso 1: Elección manual de un grupo de imágenes con ruidos significativos.

Paso 2: Aplicar una función de aumento de datos y almacenarlas en un directorio.

Paso 3: Buscar una herramienta gratuita de edición de imágenes disponible en la web.

Paso 4: Eliminar el fondo en cada imagen y almacenarlas en un directorio.

Paso 5: Emplear una función de umbralización para obtener las máscaras binarias.

Paso 6: Guardar las máscaras binarias en un directorio.



Figura 2.8: Resultados obtenidos del algoritmo.

Antes de empezar con el desarrollo de la arquitectura CNN, el autor [48] menciona que es necesario preparar los datos de entrenamiento. El modulo `torch.utils.data` proporciona dos clases: `Dataset` y `Dataloader`, que permiten organizar el conjunto de datos y sus etiquetas correspondientes dentro de un *dataset*, y extraer muestras o lotes (minibatches) a través de diferentes opciones de muestreo [28], respectivamente. De acuerdo con la documentación de Pytorch,

una clase *dataset* personalizada que hereda de la clase `Dataset` requiere la configuración de tres funciones: `__init__`, `__len__` y `__getitem__`, de modo que, se realizan sus respectivas configuraciones. Luego, se definen dos transformaciones: redimensionamiento (de 1280 x 1280 píxeles a 256 x 256) y conversión de las imágenes a tensores de PyTorch. Posteriormente, se genera el *dataset*. A continuación se aplica una partición del 20% con la finalidad de realizar pruebas de validación.

Para el caso del `Dataloader`, se requiere como entradas : la variable que contiene el *dataset*, el valor del minibatch y activar o no el proceso de redistribución. Según [52], para modelos basados en CNN, los valores de lotes son: 16, 32, 64, 128, 256. Aunque estos valores pueden variar debido a diversos factores, por ejemplo: la cantidad y complejidad del dato o los recursos computacionales disponibles, tal es el caso de [12] que establecen un valor de 4. En la Fig. 2.9 se puede visualizar el resultado luego de crear el *dataset* y establecer los hiperparámetros del `Dataloader`: tamaño de lote de 12 y una activación de redistribución.



Figura 2.9: Muestra extraída del *dataset* de entrenamiento creado.

Arquitectura U-Net

Este tipo de arquitecturas se basan en 2 procesos, dentro de la literatura son denominados como: *downsampling* y *upsampling*. La arquitectura U-Net presentada en [12] es el resultado de varias configuraciones que se componen principalmente de tres bloques convolucionales. En este punto, debido a que en su estudio el objeto a segmentar fue chips de papa, se decide emplear una estructura similar, manteniendo algunos hiperparámetros, con excepción del tamaño de las imágenes de entrada y del minibatch, que en esta instancia, se ha optado por imágenes de dimensiones 256 x 256 píxeles, en lotes de 12 por cada época de entrenamiento. La Fig. 2.10 es una representación de la configuración del modelo U-Net empleado.

Entrenamiento

Con la configuración de la arquitectura U-Net debidamente definida, se procede a introducir un método de entrenamiento estableciendo 3 hiperparámetros: una función de pérdida⁷ (*loss function*), un optimizador⁸ y las métricas que permitan analizar el comportamiento del modelo. Revisando la literatura [12, 53], se evidencia que la función de pérdida utilizada en problemas de segmentación semántica binaria es *binary_crossentropy*, sin embargo, en este caso se emplea *cross_entropy* utilizada por [48]. A diferencia de [12], se propone emplear el optimizador Adam para minimizar el error de la función de pérdida. Finalmente, se define como métricas de evaluación: *val. accuracy*, *val. loss*, *dice*, *IoU*.

⁷Función de pérdida o *loss function*: es una forma de cuantificar el error que existe entre el valor predicho con el real. Es decir, permite estimar la proximidad a la solución correcta [25].

⁸Optimizador: Es un algoritmo que permite ajustar los pesos y bias para minimizar el error de una función de pérdida, dentro del DL suelen emplearse variantes del descenso de gradiente estocástico, como son: Adam o RMSprop [25].

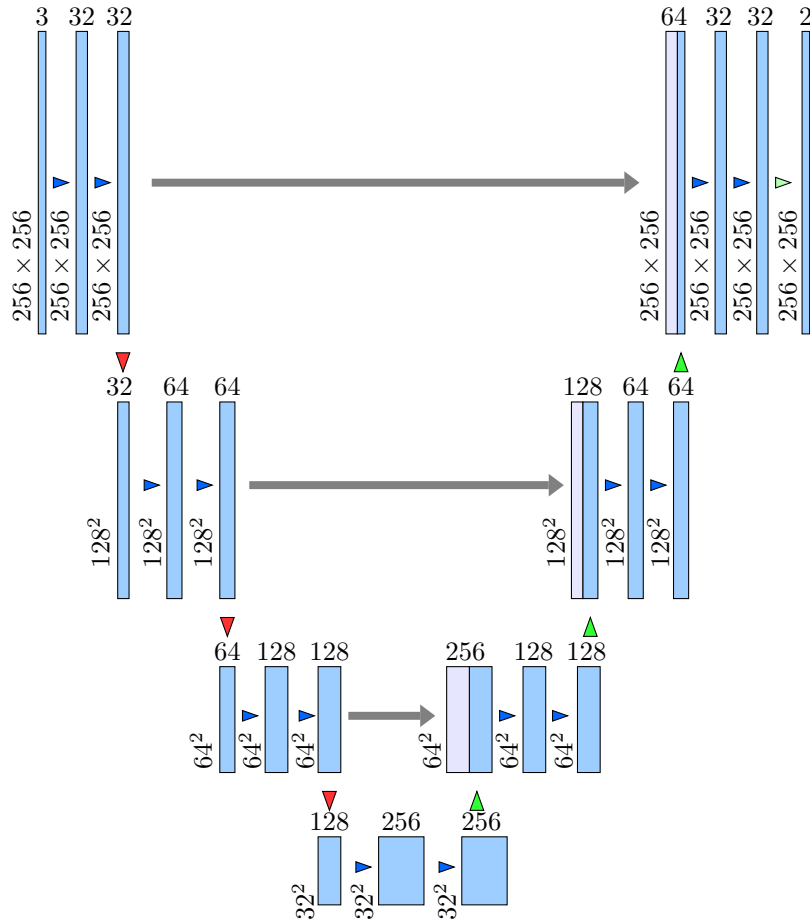


Figura 2.10: Arquitectura U-Net utilizada.

En el script de [48], presenta una función llamada `find_lr` que permite determinar un valor máximo para la tasa de aprendizaje o *learning rate* (lr) que será introducido antes de empezar el entrenamiento. Para ello se crea un modelo de prueba estableciendo un valor de semilla igual a 42, se detalla los parámetros requeridos por la arquitectura U-Net: número de canales de la imagen de entrada = 3, número de filtros iniciales = 16 y la cantidad de clases que se predicen = 2. A continuación, se establece un optimizador, en este caso se utiliza Adam con los siguientes hiperparámetros: el modelo creado, $lr = 0.0001$. Para concluir, al definir tres variables antes de llamar a la función, se habilita la posibilidad de almacenar los resultados correspondientes a la pérdida, lr y *accuracy* (acc). Tras completarse el proceso, se traza una gráfica (ver Fig. 2.11) entre los datos de pérdida y lr . Luego de evaluar el modelo con diferentes lr , max , se decide optar por un valor de 0.0024^9 .

En esta instancia, es posible observar como se comporta el modelo de prueba utilizando el `Dataloader` de validación (ver Fig. 2.12), debido a que, en cierta medida esta función simula un aprendizaje utilizando el `Dataloader` de entrenamiento, esto se debe a que faculta un aumento gradual de un lr inicial hasta un lr final predefinidos, posibilitando ver el comportamiento de la función de pérdida. Este proceso se ve reflejado en la Fig. 2.11, a medida que el lr aumenta de forma progresiva, el error de *loss function* se reduce. Sin embargo, llega un punto donde el lr es muy elevado, resultando en un aumento del valor de error, ese fenómeno es conocido como sobre-ajuste o *overfitting*.

⁹Véase en la sección 3.1

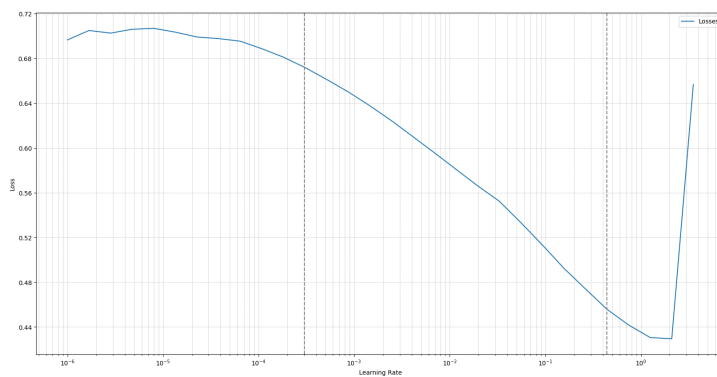


Figura 2.11: Lr vs pérdida - U-Net 1.

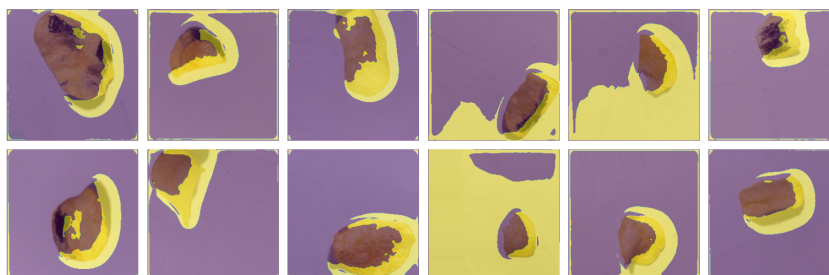


Figura 2.12: Muestras de predicción antes del entrenamiento del modelo U-Net 1.

Con los hiperparámetros establecidos, es momento de definir el método de entrenamiento que posibilite su integración y permita calcular las métricas. Las funciones: `accuracy` y `train` proporcionadas en [48], facilitan llevar a cabo este proceso. La primera función requiere como variables de entrada: una instancia del modelo U-Net entrenado y un `loader`, para poder determinar y retornar los valores de: costo promedio, *accuracy*, los coeficientes dice e IoU, siempre y cuando, se haya evaluado el modelo utilizando el `DataLoader` de validación. Es evidente que, aún cuando se disponga de la variable `loader`, no se ejecutará, debido a la ausencia del primer dato de entrada.

Para obtenerlo, es necesario introducir esta instancia a un ciclo de entrenamiento a lo largo de varias épocas. En cada época, se debe de iterar en el *dataset* y extraer un *minibatch* mediante el `DataLoader` de entrenamiento (imágenes, máscaras binarias). Las imágenes deben propagarse a lo largo de la red y obtener un *score*. No obstante, este valor no estaría sujeto a restricciones y no representaría probabilidades. Mediante una función de pérdida, se normaliza el *score* y se calcula el error existente entre las predicciones del modelo y las máscaras binarias.

Tras ello, es necesario actualizar los parámetros de la red, con la ayuda del algoritmo conocido como *backpropagation*¹⁰ y de un optimizador es posible realizar este proceso. La particularidad de la función `train` es que, proporciona un proceso que mejora la eficacia de un método de entrenamiento, al introducir un `scheduler` para tener un *lr* dinámico.

Por otra parte, se establece variables que permitan realizar un monitoreo continuo de las predicciones correctas y del costo acumulado a lo largo el ciclo de entrenamiento. Estas variables, son empleadas dentro de un proceso de evaluación que se activa luego de un cierto número

¹⁰*Backpropagation*: es un algoritmo que utiliza la regla de la cadena para calcular el gradiente de la función de pérdida con respecto a cada uno de los parámetros de la red, partiendo desde la capa de salida hasta la capa de entrada [25].

de iteraciones; este dato de entrada, es definido en la función. En vista de que, la variable de entrada anteriormente no disponible ha sido generada, es posible llamar a la primera función para valorar el desempeño del modelo con los datos de validación y cuantificar las métricas de rendimiento para el *minibatch* de validación.

El cálculo de las métricas de rendimiento para el *minibatch* de entrenamiento son obtenidos con las variables de monitoreo que se imprimen en el terminal o consola. Todo este ciclo iterativo se repite hasta finalizar las épocas establecidas. Cabe mencionar que, se añade unas líneas de código a la función para almacenar un historial de las métricas con el fin de visualizar el comportamiento del entrenamiento a través de unas gráficas.

Ahora es el turno de configura los datos de entrada que requiere la función `train`. Estas variables son: inicialización del modelo, épocas, optimizador y un programador; cada uno con las siguientes especificaciones:

- Modelo: Este depende de los parámetros que se haya establecido como entrada. Estos son: número de canales de la imagen, número de filtros iniciales, cantidad de clases a predecir. Debido a que se va introducir imágenes RGB, el número de canales será $= 3$. Luego de realizar varias iteraciones teniendo en cuenta los recursos computacionales disponibles, se establecen 16 filtros. Para asignar el número de clases a predecir, hay que tener en cuenta la función de pérdida que se haya elegido, al optar por la función `cross_entropy`, este valor será $= 2$.
- Épocas: Este valor depende de muchos factores, como son: la complejidad de la arquitectura, el tamaño del conjunto de datos, la complejidad del problema, el *lr* empleado, entre otros. Se establece un valor de 50 épocas.
- Optimizador: A diferencia de [48], se decide emplear Adam, con sus valores por defecto.
- Programador o `scheduler`: Se toma como referencia la configuración establecida en [48]; pero se cambia el valor del `max_lr` por el calculado a través de la función `find_lr`.



Figura 2.13: Muestras de predicción después del entrenamiento del modelo U-Net 1.

Una vez, finalizado el entrenamiento se realiza una verificación del modelo con un *minibatch* aleatorio del conjunto de datos de validación. A diferencia de la Fig. 2.12 evaluada antes de entrenar el modelo, se evidencia una mejoría notable en la predicción de las máscaras binarias (ver Fig. 2.13) luego de concluir el entrenamiento.

2.4. Clasificador basado en CNN

Tal como se evidenció en la revisión de la literatura, este tipo de arquitecturas arrojan resultados satisfactorios. Además, como se había mencionado al inicio del capítulo, se decide tomar como base la arquitectura presentada en el script de [45] y el modelo U-Net 1 implementado para el desarrollo del clasificador.

2.4.1. Elaboración del conjunto de datos

Antes de empezar a desarrollar el conjunto de datos para el entrenamiento, se evalúa el método establecido por [45] con el propuesto en este estudio a través de una imagen aleatoria de cada clase contenida en la carpeta “*Train*”. Al utilizar el método de extracción de la ROI establecido por el autor, se observa (ver Fig. 2.14) que los resultados no son los deseados. Puesto que, al eliminar el fondo de las imágenes, se visualizan restos de sombras presentes alrededor del objeto de interés. En cambio, al utilizar el modelo U-Net se reduce considerablemente este error (ver Fig. 2.15).



Figura 2.14: Resultado del método propuesto por el autor.



Figura 2.15: Resultado aplicando el modelo U-Net 1.

Aumento de datos

El siguiente paso, consiste en introducir este modelo dentro de un proceso de “*data augmentation*”. Tanto el framework de Pytorch como Tensorflow, poseen maneras para realizar este aumento. Sin embargo, en el caso de ImageDataGenerator perteneciente a Tensorflow, posee un parámetro denominado como `fill_mode`; al establecer un valor de ‘*nearest*’, se activa la opción de rellenado de píxeles con valores próximos a la imagen original. Es muy útil, principalmente en rotación; al aplicar esta transformación se generan espacios que carecen de información válida para rellenarlos, estos espacios a menudo toman valores predeterminados representados de color negro. Configurar el parámetro con este valor, se mitiga este error.

Una particularidad de esta clase, es que requiere que el conjunto de imágenes se encuentre dentro de una estructura jerárquica de directorios, también conocido como “árbol de directorios”. Una representación de esta estructuración se visualiza en la Fig. 2.16. El proceso de *data augmentation* aplicado está conformado por las siguientes transformaciones: normalización de píxeles, rotaciones aleatorias, desplazamientos horizontales y verticales, zoom, y efectos espejo tanto horizontales como verticales. Además, se introduce una función que permite utilizar el modelo U-Net 1 dentro de un proceso para aplicar la máscara predicha en las imágenes y extraer la ROI. Finalmente, para evitar áreas sin información en las imágenes, se activa la opción ‘*nearest*’.

A continuación, se define tres generadores que permiten: extraer un lote de imágenes aleatorias, redimensionar las imágenes, definir el tipo de clasificación, aplicar el proceso de *data augmentation*, y reorganizar aleatoriamente el lote de datos. Cabe mencionar que la técnica de *data augmentation* solo es aplicado al conjunto de datos de la carpeta “*Train*”. En cuanto a los directorios de “*Validation*” y “*Test*”, se define un proceso básico de transformaciones: normalización y aplicar la función U-Net para segmentar la ROI. Las Fig. 2.17 y 2.18 representan el

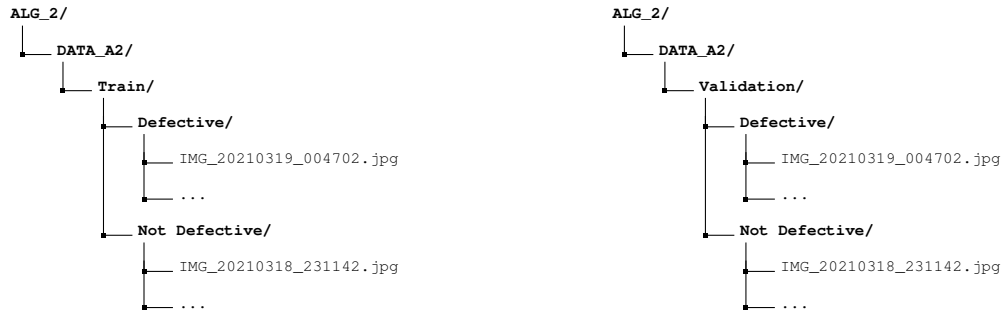


Figura 2.16: Muestra de un árbol de directorios.

resultado de este procedimiento.

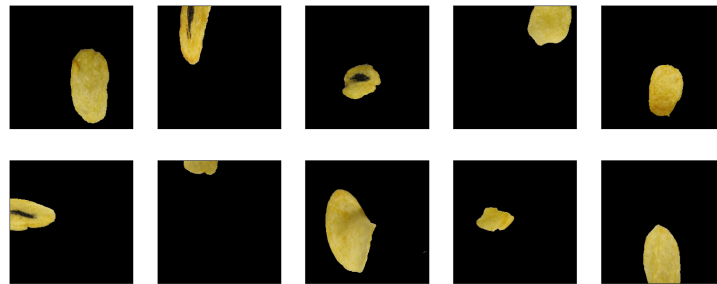


Figura 2.17: Resultado de *data augmentation* - directorio “Train”.

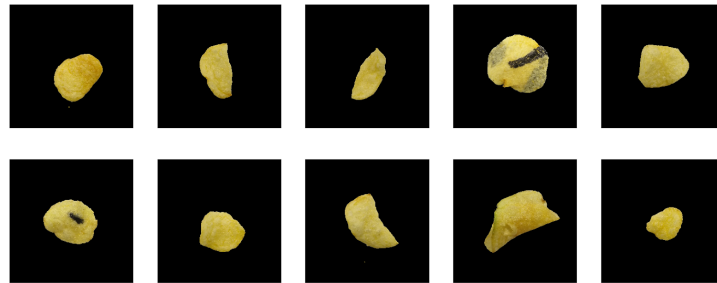


Figura 2.18: Resultados - directorio “Validation”.

Equilibrio de datos

En un problema de clasificación, el equilibrio entre la cantidad de datos contenidos en cada clase es crucial. En caso de existir una diferencia considerable, la CNN durante el entrenamiento podría priorizar la clase con mayor peso, y lo que se requiere es que el modelo se comporte imparcialmente con los datos que se presenten. Ahora, en la Fig. 2.4a se observa una diferencia en la cantidad de datos entre las clases, exactamente de 24 imágenes. Existen varias formas de abordar este error, como son: sobre-muestrear la clase con menores datos, sub muestrear la clase con mayores datos.

En este estudio, se decide elegir la alternativa empleada por [54] conocida como ponderación de clases o *class weight*, que consiste en hacer un análisis de probabilidades para todas las clases aplicando la ecuación 2.1; esto permite cuantificar el peso de cada clase. Se evidencia que la clase “Defective” tiene un peso mayor a 1 (ver Tabla 2.2), lo que significa que esta clase tiene menos

representación, es decir, menor cantidad de datos. Estos valores serán introducidos dentro de un hiperparámetro que contiene el método `fit` de Keras.

$$w_i = \frac{N}{C \cdot N_i} \quad (2.1)$$

Donde:

w_i = Es el peso de la clase.

N = Cantidad total de datos.

C = Número de clases.

N_i = Cantidad de datos en la clase i .

Tabla 2.2: Clases y pesos

Num	Clase	Etiqueta	Peso
0		Defective	1,04
1		Non-Defective	0,96

2.4.2. Arquitectura CNN

En secciones anteriores, se ha descrito que este tipo de redes neuronales, se encuentran en su mayoría compuesta por tres capas: convolucionales, de pooling y totalmente conectada. El número de capas o implementar una capa en específico se encuentra condicionado al problema que se esté abordando. En el marco de este estudio, se adopta una arquitectura presentada por [45]. Sin embargo, se ha modificado varios parámetros, que han sido seleccionados como resultado de varias iteraciones de entrenamientos¹¹. La Tabla 2.3 detalla la composición de la CNN.

Tabla 2.3: Estructura CNN empleada.

Sección	Num - capa	Tipo - capa	Dimensiones de salida
Extracción de características	1	Capa de entrada	(None, 256, 256, 3)
	2	Capa Convolutacional	(None, 254, 254, 32)
	3	MaxPooling	(None, 127, 127, 32)
	4	Dropout	(None, 127, 127, 32)
	5	Capa Convolutacional	(None, 125, 125, 64)
	6	MaxPooling	(None, 62, 62, 64)
	7	Dropout	(None, 62, 62, 64)
	8	Capa Convolutacional	(None, 60, 60, 128)
	9	MaxPooling	(None, 30, 30, 128)
	10	Dropout	(None, 30, 30, 128)
	11	Capa Convolutacional	(None, 28, 28, 256)
	12	MaxPooling	(None, 14, 14, 256)
	13	Dropout	(None, 14, 14, 256)
Clasificación	14	Flatten	(None, 50176)
	15	Dropout	(None, 50176)
	16	Dense	(None, 256)
	17	Dropout	(None, 256)
	18	Softmax	(None,2)

¹¹Véase en la sección 3.2.2

2.4.3. Entrenamiento

Una vez definido los parámetros de la CNN, es momento de configurar los hiperparámetros que contribuyen en el desempeño de entrenamiento. Al igual que en el modelo U-Net, se requiere de 3 hiperparámetros: una función de coste, un optimizador y definir las métricas. Keras ofrece herramientas que ayudan a refinar el monitoreo del entrenamiento. Como se había hecho énfasis en resolver el desbalance de datos mediante una técnica de ponderación de clases al aplicar un valor de entrada que contiene el método `fit`. Esta opción lleva el nombre `class_weight`, permite introducir los pesos de cada clase, a fin de indicar a la red que clase debe dar mayor prioridad. Esto es posible, debido a que se manipula el resultado de la función de pérdida, introduciendo estos pesos. A continuación se detalla las configuraciones principales que se toman en cuenta:

- `class_weight`: Se introduce los pesos contenidos en un diccionario.
- `steps_per_epoch`: Este valor puede ser adquirido del generador `train` calculando la longitud de minibatches que ingresarán durante todo el ciclo de entrenamiento.
- `epochs`: No existe un valor único y definido del número. Principalmente es obtenido luego de varias iteraciones, experimentaciones y observación. Por ejemplo en [19] y [37] definen unos valores de 40 y 100 épocas, respectivamente.
- `validation_steps`: Este dato puede ser encontrado al determinar la longitud del generador de validación.
- `callbacks`: Son directrices que se ejecutan en cierto puntos del proceso de entrenamiento.

Entre todas estas opciones, los `callbacks` permiten llamar a diferentes clases que contribuyen al desarrollo óptimo del entrenamiento. Tal es el caso de `ModelCheckpoint`, que, en base a una métrica elegida, se establecen condiciones de como se debe de guardar el modelo, por ejemplo: almacenar los pesos omitiendo su arquitectura o solamente cuando exista una mejoría en la métrica monitoreada. En este proceso, se decide conservar el modelo cuando se evidencie cambios en la métrica *val. accuracy*. Por último, si bien es un problema de clasificación binaria, se opta por emplear la función de pérdida `categorical_crossentropy` en conjunto del optimizador Adam.

2.5. Método por conteo de píxeles

En este punto, aún cuando el clasificador posibilita predecir si una chips de papa presenta defectos o no, una limitación crítica radica en su clasificación binaria, donde a la presencia de una mancha de color negro en la chips de papa conduce a la categorización como defectuosa, sin considerar el tamaño del área afectada. Si el modelo fuese implementado en un sistema de clasificación y las manchas de color negro representaran algún defecto (exceso de fritura, manchas naturales, etc.), existe la posibilidad que se descarte gran parte del producto.

2.5.1. Creación de modelo U-Net 2

Debido a este problema, se plantea cuantificar el porcentaje del área afectada. Para llevar a cabo este proceso, se ha optado por usar una variante del modelo U-Net que se desarrolló en secciones anteriores¹².

¹²Véase en la sección 2.3.1

Conjunto de datos

El propósito de este modelo es generar máscaras binarias que representen el área defectuosa. Para ello, se ha optado por hacer uso de las imágenes generadas correspondientes a la clase defectuosa del directorio “*Data_Nground*”. Sin embargo, debido a la complejidad del área defectuosa, el proceso para obtener las máscaras mediante las imágenes debe modificarse. A través de una herramienta denominada *Labelme* se realiza el etiquetado manual de la ROI defectuosa, los mismos que son representados por archivos tipo “*.json*”. Luego, mediante un proceso de transformación se obtiene sus máscaras binarias.

Si bien las imágenes que se usan para el etiquetado no poseen el fondo, es pertinente resaltar que, el conjunto de imágenes empleadas para el entrenamiento, proviene de la carpeta “*Data_Augmentation*”¹³. Esta elección se realiza con el propósito de permitir al modelo diferenciar entre el fondo, las sombras, las áreas sin defecto y la ROI defectuosa. En la Fig. 2.19 se visualiza el resultado luego crear el *dataset* y configurar los hiperparámetros del *Dataloader*: tamaño de lote de 8 y una activación de redistribución.

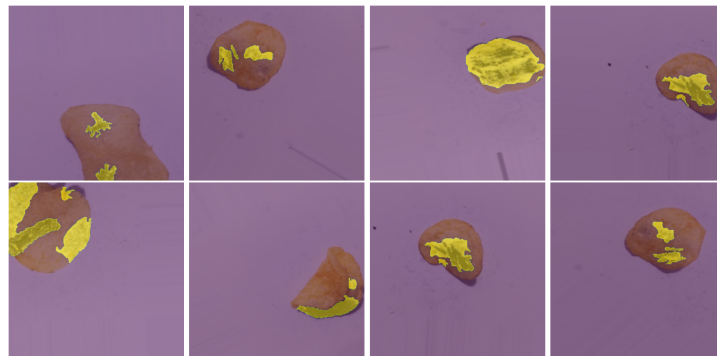


Figura 2.19: Lote de imágenes extraído del *dataset* creado.

Arquitectura U-Net 2

A diferencia de la arquitectura de la Fig. 2.10, se ha aumentado una fila de capas convolucionales para abordar la complejidad de las máscaras binarias a predecir y, compensar la disminución del conjunto de imágenes de entrenamiento. Es decir, dentro del proceso de *downsampling*, la resolución de la imagen disminuye y la profundidad de la red aumenta. Por ejemplo, al usar la arquitectura U-Net 1, las dimensiones iniciales que ingresan fuesen (8, 256, 256), correspondientes a: número de *kernels*, alto y ancho de la imagen; la salida de proceso de *downsampling* sería (64, 32, 32). Dentro de la arquitectura U-Net 2, estos valores de salida cambian a (128, 16, 16), lo que indica que se han empleado 128 *kernels* y la resolución de la imagen ha disminuido a 16 x 16 píxeles.

Entrenamiento

Aún cuando el proceso de entrenamiento que se lleva a cabo es similar al empleado para el modelo U-Net 1¹⁴, los valores que se establecen para la función `find_lr` son las siguientes:

- Parámetros requeridos por la arquitectura U-Net: número de canales = 3, número de *kernels* = 16, cantidad de clases = 3.

¹³A raíz de los resultados que se evidenciaron durante las pruebas (ver sección 3.3), se realiza un aumento de imágenes de 160 a 320.

¹⁴Véase en la sección 2.3.1.

- Optimizador: Se emplea SGD con lr (0.01), $momentum = 0.95$ y $weight_decay = 1e^{-4}$.

Al finalizar, se traza la gráfica de la función de pérdida vs el lr (ver Fig. 2.20) y, se evalúa el lr_max óptimo para el `scheduler`. La Fig. 2.21 representa el resultado del entrenamiento de la función. Es posible observar un error evidente en las máscaras predichas, pues abarcan gran parte del área de las chips.

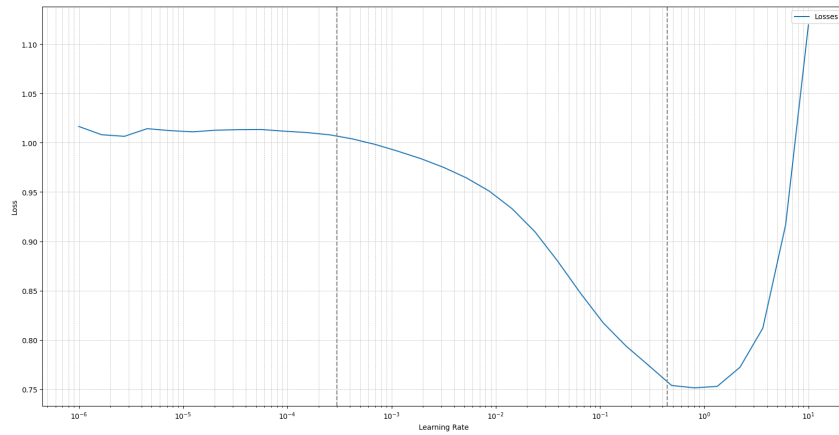


Figura 2.20: Lr vs pérdida - U-Net 2.



Figura 2.21: Muestras de predicción antes del entrenamiento del modelo U-Net 2.

Con respecto a la función `train`, se mantiene el optimizador Adam, pero las demás variables se realizan los siguientes cambios:

- Modelo: Número de canales = 3, número de filtros = 8, número de clases = 3.
- Épocas: Se establece un valor de 100 épocas.
- Programador o `scheduler`: Se introduce el valor que permita obtener los mejores resultados. Para ello, se debe realizar varias iteraciones¹⁵ y observar el comportamiento de las métricas *dice* e *IoU*.

Una vez definidos los valores de entrada requeridos por la función `train`, se procede al entrenamiento del modelo. Al monitorear las métricas (ver Fig. 2.22) se observa que, luego de la época 89 el coeficiente *dice* empieza a estabilizarse con un valor de 0.92; en el caso del valor *IoU*

¹⁵Véase en la sección 3.3.

```

epoch: 88, mb: 32, train cost: 0.0132, val cost: 0.0411, train acc: 0.9949, val acc: 0.9940, dice: 0.91897, iou: 0.85009
epoch: 89, mb: 32, train cost: 0.0128, val cost: 0.0155, train acc: 0.9950, val acc: 0.9945, dice: 0.92584, iou: 0.86192
epoch: 90, mb: 32, train cost: 0.0125, val cost: 0.0152, train acc: 0.9952, val acc: 0.9942, dice: 0.92296, iou: 0.85694
epoch: 91, mb: 32, train cost: 0.0130, val cost: 0.0146, train acc: 0.9950, val acc: 0.9945, dice: 0.92650, iou: 0.86306
epoch: 92, mb: 32, train cost: 0.0124, val cost: 0.0156, train acc: 0.9952, val acc: 0.9942, dice: 0.92235, iou: 0.85590
epoch: 93, mb: 32, train cost: 0.0127, val cost: 0.0159, train acc: 0.9951, val acc: 0.9945, dice: 0.92608, iou: 0.86233
epoch: 94, mb: 32, train cost: 0.0123, val cost: 0.0157, train acc: 0.9953, val acc: 0.9945, dice: 0.92595, iou: 0.86211
epoch: 95, mb: 32, train cost: 0.0129, val cost: 0.0147, train acc: 0.9950, val acc: 0.9944, dice: 0.92491, iou: 0.86030
epoch: 96, mb: 32, train cost: 0.0126, val cost: 0.0170, train acc: 0.9952, val acc: 0.9944, dice: 0.92449, iou: 0.85959

```

Figura 2.22: Monitoreo de métricas - U-Net 2.

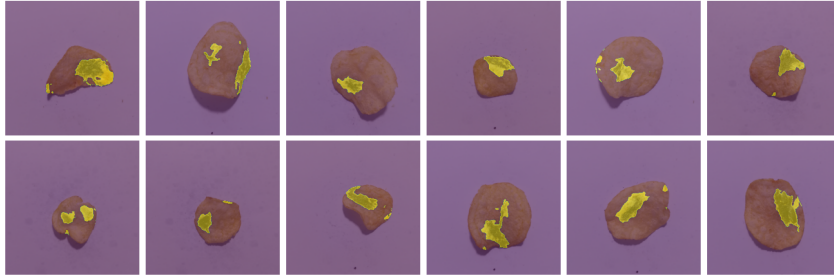


Figura 2.23: Muestras de predicción después del entrenamiento del modelo U-Net 2.

existe una oscilación mínima entre 0.85 - 0.86. Al finalizar el proceso, se evalúa el rendimiento del modelo y se visualiza que las predicciones (ver Fig. 2.23) del *dataset* de validación mejoran notoriamente.

2.5.2. Cálculo de píxeles

Pese a que se ha desarrollado un modelo U-Net para segmentar el área defectuosa, resulta imposible calcular su porcentaje. Para obtener la cantidad de píxeles, es necesario implementar un proceso que permita utilizar ambos modelos U-Net. El fin de su integración, es crear una sinergia que supere las limitaciones inherentes a cada uno.

Con ese propósito, se desarrolla una función que calcule el número de píxeles de la máscara binaria generada por cada modelo U-Net. Para hacer uso de ella, se define tres directorios: “*Entrada*”¹⁶, “*Salida_0*”¹⁷ y “*Salida_1*”¹⁸. Luego, se implementa un proceso que facilita cargar las imágenes ubicadas en la carpeta “*Entrada*”.

Finalmente, a través de una división, se determina el porcentaje de defecto, que posteriormente se compara con un valor establecido asignado en una variable, posibilitando su ajuste conforme a las necesidades. Todo este procedimiento forma parte de la sección representada en color verde en la Fig. 2.2.

¹⁶El contenido de esta ruta son las imágenes pertenecientes al conjunto de imágenes de prueba.

¹⁷Es la carpeta donde se almacenan las imágenes de las chips descartadas por tener un defecto.

¹⁸Es la ruta donde se almacenan las imágenes de las chips con ausencia de defectos.

Capítulo 3

Pruebas y resultados

En este capítulo se describe el procedimiento llevado a cabo para la validación y selección de los modelos que garanticen un rendimiento aceptable para abordar el problema planteado. Este análisis ha permitido evidenciar en como generalizan nuevos datos cada uno de los modelos desarrollados. Además, luego de su incorporación para que trabajen en conjunto, se presenta las respectivas pruebas utilizando dos conjuntos de datos: el reservado en la carpeta “*Test*” y, un grupo adicional de imágenes del objeto de estudio, las cuales fueron capturadas mediante un dispositivo móvil. Los modelos U-Net y CNN han sido entrenados en un entorno de desarrollo integrado disponible en la web, conocido como Colab. Las pruebas de validación y el proceso general se han compilado y ejecutado en un ordenador Asus ROG GL553VD con las siguientes características:

- Windows 10 Home de 64 bits
- Intel(R) Core(TM) i7-7700HQ
- 16 GB de RAM
- NVIDIA® GeForce® GTX 1050 con 4GB
- SATA III HDD 1 TB

3.1. Análisis del modelo U-Net 1

3.1.1. Criterios de evaluación

En problemas de segmentación semántica existen 2 tipos de métricas que permiten cuantificar el desempeño de predicción del modelo. Para este caso, se emplearon las siguientes métricas:

- *dice*: Permite calcular la superposición entre la máscara real y la máscara predicha.

$$\text{Dice Coefficient} = \frac{2 \times \text{Área de Intersección}}{\text{Área de la Máscara Original} + \text{Área de la Máscara Predicha}} \quad (3.1)$$

- *IoU*: Mide la proporción de la región de intersección con respecto a la región total cubierta por ambas máscaras.

$$\text{IoU} = \frac{\text{Área de Intersección}}{\text{Área de Unión}} \quad (3.2)$$

3.1.2. Resultados de entrenamiento

Los modelos comparados son el resultado de un entrenamiento con un conjunto de 320 imágenes con sus respectivas máscaras binarias a lo largo de 50 épocas. En la Fig. 3.2 se puede visualizar el comportamiento de las curvas de *val. accuracy* y *val. loss* bajo diferentes valores de *lr. max* durante el entrenamiento. Es posible constatar que, aumentar este hiperparámetro facilita una convergencia más rápida, sin embargo, también induce oscilaciones que podrían afectar el rendimiento de predicción del modelo, esto es evidenciado en las métricas obtenidas de la Tabla 3.1.

Se encontró que definiendo un *lr. max* igual a 0.0024 resultó en un buen ajuste del modelo, puesto que, los valores de *dice* e *IoU* llegaron a 0.99650 y 0.99302, respectivamente. En cambio, al introducir un valor de 3 los coeficiente *dice* y *IoU* resultaron afectados. Si bien, durante la época 8 alcanzaron unos valores máximos de: 0.97724 y 0.95549, luego de la época 13 empezaron a presentar fluctuaciones considerables (ver Fig. 3.1).

```
epoch: 13, mb: 15, train cost: 0.1179, val cost: 0.0752, train acc: 0.9782, val acc: 0.9868, dice: 0.94980, iou: 0.90440
epoch: 14, mb: 15, train cost: 0.0780, val cost: 0.0820, train acc: 0.9861, val acc: 0.9841, dice: 0.93843, iou: 0.88400
epoch: 15, mb: 15, train cost: 0.1898, val cost: 0.2478, train acc: 0.9612, val acc: 0.9364, dice: 0.69248, iou: 0.52962
epoch: 16, mb: 15, train cost: 0.2324, val cost: 0.2478, train acc: 0.9567, val acc: 0.9425, dice: 0.73006, iou: 0.57488
epoch: 17, mb: 15, train cost: 0.2243, val cost: 0.2239, train acc: 0.9502, val acc: 0.9664, dice: 0.85882, iou: 0.75257
epoch: 18, mb: 15, train cost: 0.2014, val cost: 0.1930, train acc: 0.9668, val acc: 0.9649, dice: 0.85073, iou: 0.74023
epoch: 19, mb: 15, train cost: 0.6095, val cost: 0.5211, train acc: 0.9487, val acc: 0.9229, dice: 0.60199, iou: 0.43061
epoch: 20, mb: 15, train cost: 3.4669, val cost: 2.2138, train acc: 0.7728, val acc: 0.8778, dice: 0.17578, iou: 0.09636
epoch: 21, mb: 15, train cost: 0.6965, val cost: 0.5785, train acc: 0.8804, val acc: 0.8770, dice: 0.16925, iou: 0.09245
epoch: 22, mb: 15, train cost: 0.3826, val cost: 0.3932, train acc: 0.8821, val acc: 0.8765, dice: 0.16048, iou: 0.08724
epoch: 23, mb: 15, train cost: 0.3900, val cost: 0.3870, train acc: 0.8783, val acc: 0.8755, dice: 0.14780, iou: 0.07980
epoch: 24, mb: 15, train cost: 0.3817, val cost: 0.3842, train acc: 0.8757, val acc: 0.8759, dice: 0.15199, iou: 0.08225
```

Figura 3.1: Monitoreo de métricas con un *lr. max* de 3.

Tabla 3.1: Métricas máx. de los modelos U-Net 1.

Modelo	lr	lr. max	val. accuracy	val. loss	dice	IoU
1	0.0001	0.0003	0.99849	0.05588	0.99457	0.98919
2	0.0001	0.0024	0.99903	0.00436	0.99650	0.99302
3	0.0001	0.0552	0.99860	0.00358	0.99497	0.99000
4	0.0001	0.3	0.99633	0.00953	0.98680	0.97394
5	0.01	0.3	0.99672	0.00849	0.98820	0.97668
6	0.1	3	0.99380	0.01640	0.97724	0.95549

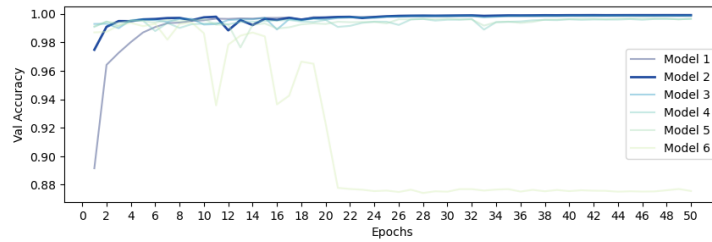
3.1.3. Selección del modelo U-Net 1

En este punto, el modelo con las mejores métricas durante el entrenamiento es el número dos, no obstante, se evalúa el desempeño de predicción de cada uno de los modelos utilizando el conjunto de imágenes contenidas en la carpeta “*Test*”. Para ello, es necesario obtener las máscaras binarias de cada imagen mediante un método similar detallado en secciones anteriores¹, con la diferencia que se omite los pasos 1 y 2; éstas máscaras son almacenadas en un directorio llamado “*Maskpruebas*”.

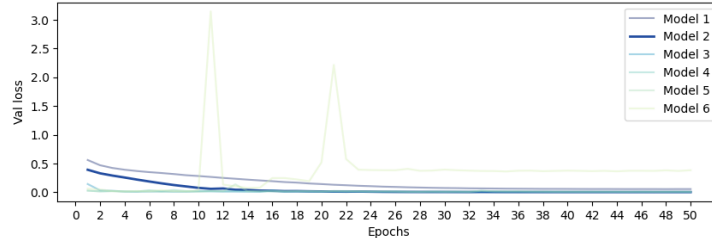
Por otra parte, por cada modelo se almacena las máscaras predichas en una ruta correspondiente. Con las imágenes y sus respectivas máscaras binarias listas, se compara los resultados a través de un análisis estadístico, a fin de identificar el modelo que garantice: robustez en la predicción, minimizar la variabilidad, valores IoU y dice mayores a 95 %.

Las medidas estadísticas obtenidas para el análisis son presentadas en las Tablas 3.2 y 3.3. Estos datos han permitido interpretar con mayor profundidad la distribución y la dispersión de

¹Véase en la Tabla 2.1.



(a) Val. accuracy



(b) Val. loss

Figura 3.2: Curvas de entrenamiento - U-Net 1.

los resultados. Además, para una representación visual de la variabilidad inherente en la calidad de predicción se emplea un diagrama de cajas y bigotes (ver Fig. 3.3). Los rangos intercuartiles calculados reflejan que la mediana de los modelos 1, 2 y 3 obtuvieron valores inferiores que los modelos 4, 5 y 6.

Las dimensiones de las cajas reflejan una variación en la longitud; estas diferencias de tamaño representan la variabilidad de predicción. Si bien, el modelo 6 presenta la mediana de mayor valor, posee una mayor dispersión de los coeficientes *dice* e *IoU* en el 50% de los datos y un número mayor de valores atípicos. En el caso de los modelos 4 y 5, también poseen indicios de una amplia variabilidad, de modo que, al no garantizar una robustez en la predicción son descartados.

Tabla 3.2: Descripción de resultados *dice* de los modelos U-Net 1.

Modelo	lr	lr. max	Mediana dice	Rango intercuartil dice	Max. dice	Min. dice	Val. atípicos dice
1	0.0001	0.0003	0.98215	0.00417	0.98708	0.97381	3
2	0.0001	0.0024	0.98187	0.00264	0.98504	0.97633	4
3	0.0001	0.0552	0.98062	0.00543	0.98754	0.96960	8
4	0.0001	0.3	0.98228	0.00720	0.99006	0.96788	14
5	0.01	0.3	0.98390	0.00784	0.99335	0.96711	14
6	0.1	3	0.98513	0.01375	0.99406	0.95661	17

De igual manera, los modelos 1, 2 y 3 también presentan estos indicios, no obstante, la variabilidad, dispersión y cantidad de valores atípicos es menor. Ahora, para la elección final del modelo, se sigue el mismo criterio de descarte: comparar la mediana, el rango intercuartil, los coeficientes *dice* e *IoU*, cantidad de valores atípicos. Adicionalmente, se calcula el rango a través de una operación de sustracción entre el máximo y el mínimo; estos datos son presentados en las Tablas 3.4 y 3.5. Al comparar el rango se observa que el modelo dos garantiza robustez en la predicción de máscaras binarias. En la Fig. 3.4 se presenta una muestra del resultado del modelo de segmentación semántica seleccionado.

Tabla 3.3: Descripción de resultados *IoU* de los modelos U-Net 1.

Modelo	lr	lr. max	Mediana IoU	Rango inter- cuartil IoU	Máx. IoU	Mín. IoU	Val. atípicos IoU
1	0.0001	0.0003	0.96492	0.00805	0.97450	0.94895	3
2	0.0001	0.0024	0.96438	0.00509	0.97053	0.95376	4
3	0.0001	0.0552	0.96198	0.01044	0.97538	0.94058	7
4	0.0001	0.3	0.96518	0.01388	0.98032	0.93776	14
5	0.01	0.3	0.96831	0.01515	0.98679	0.93632	14
6	0.1	3	0.97069	0.02656	0.98819	0.91270	16

Tabla 3.4: Resultados estadísticos de los coeficientes *dice* - U-Net 1.

Modelo	lr	lr. max	Mín. dice	Mediana dice	Máx. dice	Rango	Rango inter- cuartil dice	Val atípicos dice
1	0.0001	0.0003	0.97381	0.98215	0.98708	0.01328	0.00417	3
2	0.0001	0.0024	0.97633	0.98187	0.98504	0.00871	0.00264	4
3	0.0001	0.0552	0.96960	0.98062	0.98754	0.01793	0.00543	8

Tabla 3.5: Resultados estadísticos de los coeficientes *IoU* - U-Net 1.

Modelo	lr	lr. max	Mín. IoU	Mediana IoU	Máx. IoU	Rango	Rango inter- cuartil IoU	Val. atípicos IoU
1	0.0001	0.0003	0.94895	0.96492	0.97450	0.02555	0.00805	3
2	0.0001	0.0024	0.95376	0.96438	0.97053	0.01676	0.00509	4
3	0.0001	0.0552	0.94058	0.96198	0.97538	0.03480	0.01044	7

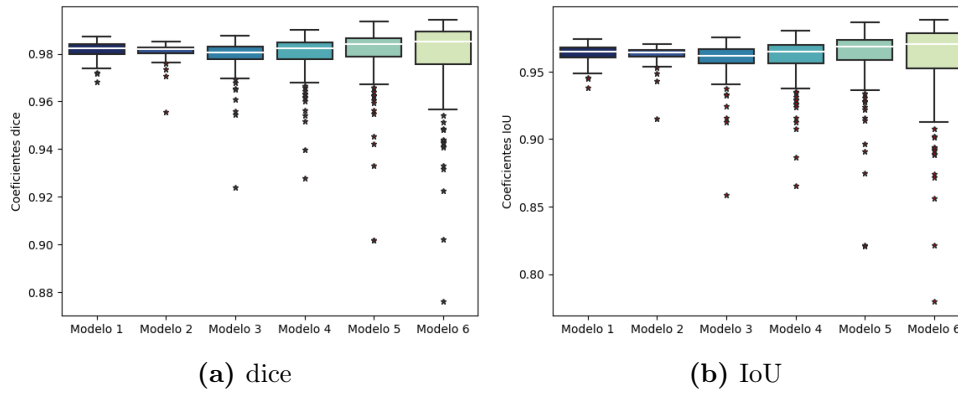


Figura 3.3: Diagrama de cajas de los modelos U-Net 1.

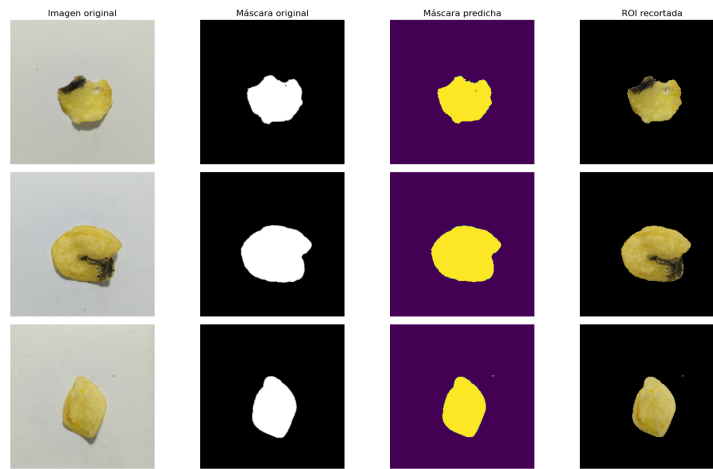


Figura 3.4: Muestra de segmentación semántica - modelo U-Net 1.

3.1.4. Comparación de técnicas

Finalmente, a través de una comparación de 3 técnicas de segmentación de la ROI, se evidencia el comportamiento frente este tipo de ruidos mencionado en secciones anteriores, pero, sobre todo, demostrar que el método seleccionado es el adecuado en imágenes con estas características. Los métodos que han sido comparados son:

- Umbralización (thresholding)
- Detección de bordes en el espacio HSV
- Segmentación semántica (modelo U-Net)

Para este caso, se utilizan varias imágenes correspondientes a la carpeta “Train”, debido a que son las que se emplean para el entrenamiento del clasificador CNN. Los coeficientes *dice* e *IoU* obtenidos de los métodos 1 y 2 demuestran variabilidad, mientras que, con el método 3, estos valores si bien en algunos casos eran inferiores, existe una similitud, demostrando la robustez en el predicción tal como se evidencia en la Tabla 3.6. La Fig. 3.5 es una representación del resultado de cada uno de los métodos utilizados.

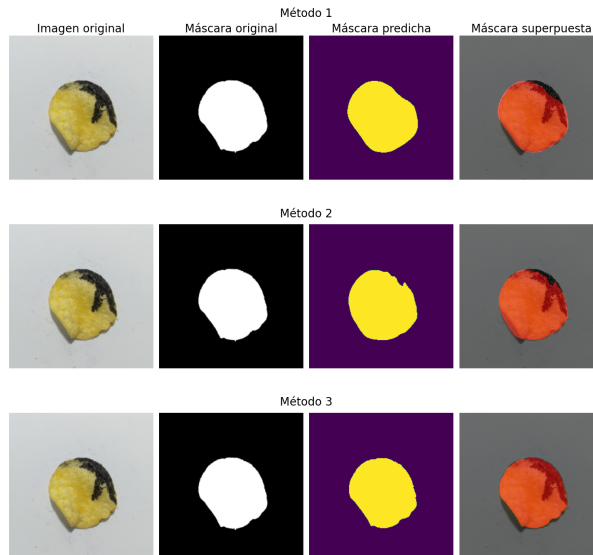


Figura 3.5: Resultado de segmentación de ROI por cada método.

3.2. Análisis del clasificador CNN

3.2.1. Criterios de evaluación

Para evaluar la calidad del clasificador, se utiliza una matriz de confusión(MC), puesto que permite relacionar las predicciones del modelo con las clases de los datos reales. La Fig. 3.6 representa la estructura de una MC para un clasificador binario. Como se puede observar, para

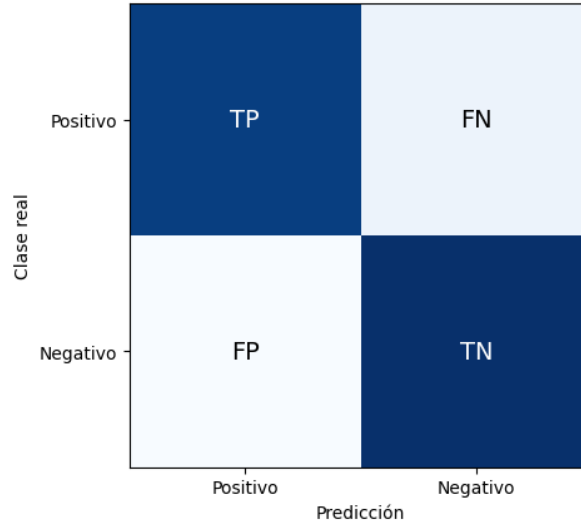


Figura 3.6: Estructura de una MC para problemas de clasificación binaria.

este caso en específico solo existen cuatro posibilidades:

- Verdaderos positivos [*TP: True Positive*]: cuando una chips de papa con defecto es clasificada en la clase defectuosa.
- Falsos positivos [*FP: False Positive*]: cuando una chips de papa sin defecto es clasificada en la clase defectuosa.
- Falsos negativos [*FN: False Negative*]: cuando una chips de papa con defecto es clasificada en la clase no defectuosa.

Tabla 3.6: Comparativa de coeficientes calculados de cada máscara generada por cada método.

Método	Muestra	Dice	Val. máx Dice	IoU	Val. máx IoU
1	1	0.97835		0.95762	
2		0.92211		0.85548	
3		0.98045	0.98045	0.96165	0.96165
1	2	0.95690		0.91737	
2		0.95398		0.91202	
3		0.98174	0.98174	0.96414	0.96414
1	3	0.96899		0.93984	
2		0.91111		0.83673	
3		0.97859	0.97859	0.95808	0.95808
1	4	0.97225		0.94600	
2		0.97664		0.95435	
3		0.98399	0.98399	0.96848	0.96848
1	5	0.98826	0.98826	0.97679	0.97679
2		0.98603		0.97244	
3		0.98673		0.97381	

- Verdaderos negativos [*TN: True Negative*]: cuando una chips de papa sin defecto es clasificada en la clase no defectuosa.

Si bien, la MC permite analizar a detalle el comportamiento del modelo en una tarea de clasificar datos de las diferentes clases, es recomendable tener a disposición métricas de rendimiento que reflejen el resultado del modelo entrenado (sensibilidad, especificidad, exactitud, precisión, medida F). Para este estudio, se emplean las siguientes ecuaciones con el fin de cuantificar las métricas del modelo:

- Sensibilidad o *Recall*: Detalla la capacidad que posee el modelo en evitar falsos negativos.

$$\text{Sensibilidad} = \frac{TP}{TP + FN} \quad (3.3)$$

- *Especificidad*: Define la capacidad que posee el modelo en evitar clasificar erróneamente los verdaderos negativos como positivos.

$$\text{Especificidad} = \frac{TN}{TN + FP} \quad (3.4)$$

- Exactitud o *accuracy*: Representa la capacidad que posee el modelo en clasificar correctamente en relación con el total de casos, por lo general es representada en tanto por ciento.

$$\text{Exactitud} = \frac{TN + TP}{TN + TP + FN + FP} \quad (3.5)$$

- *Precisión*: Representa la calidad de predicciones positivas de cada clase.

$$\text{Precisión} = \frac{TP}{TP + FP} \quad (3.6)$$

- Medida F o *F1-score*: Proporciona una medida equilibrada del rendimiento del modelo, combinando la precisión y la sensibilidad en un solo valor.

$$F1 - score = \frac{2 * Recall * Precisión}{Recall + Precisión} \quad (3.7)$$

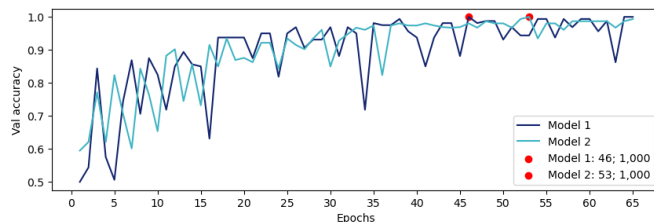
3.2.2. Resultados de entrenamiento

El modelo clasificador obtenido fue el resultado de varios procesos iterativos durante el entrenamiento, como son: configuración de parámetros e hiperparámetros, evaluación de métricas y tiempo de predicción. El total de modelos evaluados es de seis, sin embargo, para abordar el desequilibrio del conjunto de datos de entrenamiento se ha implementado 2 métodos denominados como: ponderación de clases y un sobre muestreo de la clase que presenta un desbalance. Debido a estas configuraciones, se ha dividido en 3 grupos, cada uno compuesto por dos modelos con su respectiva técnica de mencionada.

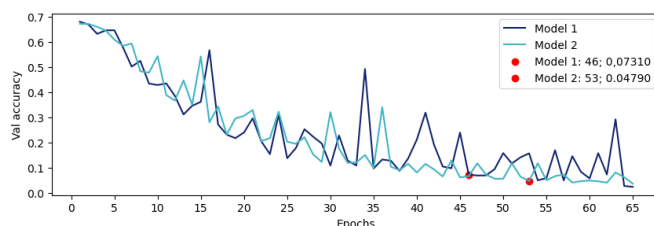
Primera configuración

La etapa de extracción de características se ha estructurado de 2 capas convolucionales (32,64 filtros) con sus respectivas capas de *maxpooling* de 2 x 2 sin *paddin*. Posteriormente, se introduce una capa de aplanamiento, seguida de una capa *dropout* con un valor de 0.05. La etapa de clasificación, se estructura de una capa densa con 128 neuronas y activación ReLU, seguida de otra capa *dropout* con una tasa del 5%. Finalmente, la predicción se produce a través de una capa densa con dos neuronas y una activación *softmax*².

Las gráficas 3.7a y 3.7b ilustran el progreso de entrenamiento del grupo 1. Si bien, el clasificador 1 requiere de menos épocas de entrenamiento para converger, obtuvo mayor inestabilidad, alcanzando un error mínimo de 0.07310 en la época 46. En el caso del clasificador 2, a pesar de requerir un mayor número de épocas para entrenarse, presenta menos fluctuaciones, alcanzando un error mínimo de 0.04790 después de la época 53.



(a) Val accuracy modelo 1 y 2.



(b) Val loss modelo 1 y 2.

Figura 3.7: Curvas de entrenamiento del grupo 1.

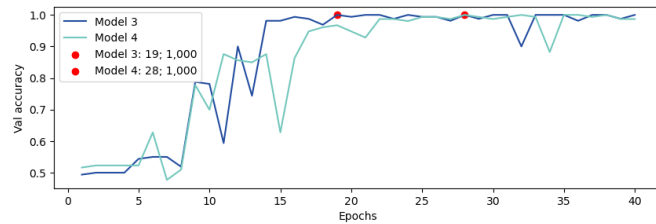
Segunda configuración

Se ha modificado la arquitectura de extracción de características mediante la incorporación de dos grupos capas similares a las existentes. Este proceso ha conllevado en un incremento

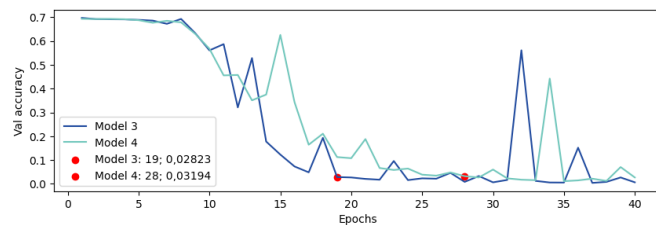
²Si bien, para problemas binarios se recomienda emplear la función de activación *sigmoid*, se observó que, la métrica de *val_acc* requirió de más épocas de entrenamiento para converger. Por otro lado, la función *softmax* proporcionó una convergencia más rápida.

del número de filtros por capa convolucional (32, 64, 128, 256) y, una reducción gradual en las dimensiones espaciales de las representaciones generadas, resultando en la obtención de un conjunto de mapas de características. Para la etapa de clasificación se aumentó el número de neuronas de 128 a 256 a la estructura de la primera configuración.

En contraste con el resultado de la configuración del modelo secuencial previo, se observa una reducción en la variabilidad de las métricas de *val. accuracy* y *val. loss* durante el transcurso del entrenamiento, así como una disminución en el número de épocas requeridas para la convergencia, alcanzando un error mínimo de 0.02823 y 0.03194 en las épocas 19 y 28 respectivamente.



(a) Val accuracy modelo 3 y 4.



(b) Val loss modelo 3 y 4.

Figura 3.8: Curvas de entrenamiento del grupo 2.

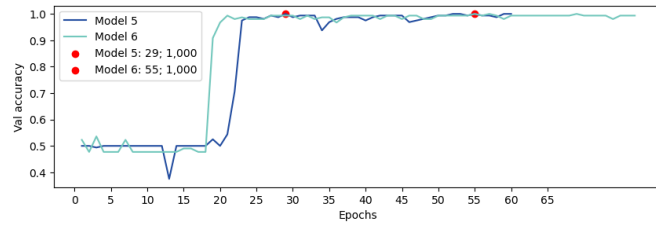
Tercera configuración

Como última instancia, se plantea complementar la estructura del modelo secuencial pre-existente mediante la adición de una capa de regularización con una tasa de abandono del 50 % en cada salida de las capas de *pooling* de modo que permitan prevenir posibles fenómenos de sobre-ajuste. Adicionalmente, se ajustaron las tasas de abandono de las capas pertenecientes a la etapa de clasificación, a 40 % y 25 %, respectivamente.

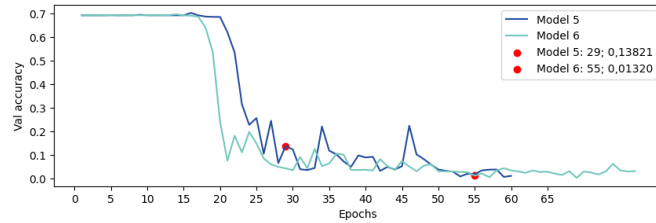
La incorporación de estos hiperparámetros ha generado cambios notables a lo largo del proceso de entrenamiento. En las primeras 18 épocas, el valor de *accuracy* de validación permaneció estático con un valor de 0.52288, no obstante, se observa una mejora progresiva en las épocas subsiguientes. Aún cuando se evidencia inestabilidad, la configuración de los *callbacks* que se ha detallado, permite almacenar el modelo sólo cuando se haya observado una mejoría en esta métrica. Por otra parte, contrario a los modelos previos, se detecta una prolongación en el número de épocas requeridas para lograr reducir el error mínimo, alcanzando valores de 0.13821 y 0.01320 en las épocas 29 y 55, respectivamente para los clasificadores 5 y 6.

3.2.3. Resultados de evaluación

Para evaluar el rendimiento de los clasificadores, se ha utilizado el mismo conjunto de imágenes de la carpeta "*Test*" (100 normales, 92 defectuosas), puesto que comparten el mismo fondo



(a) Val accuracy modelo 5 y 6.



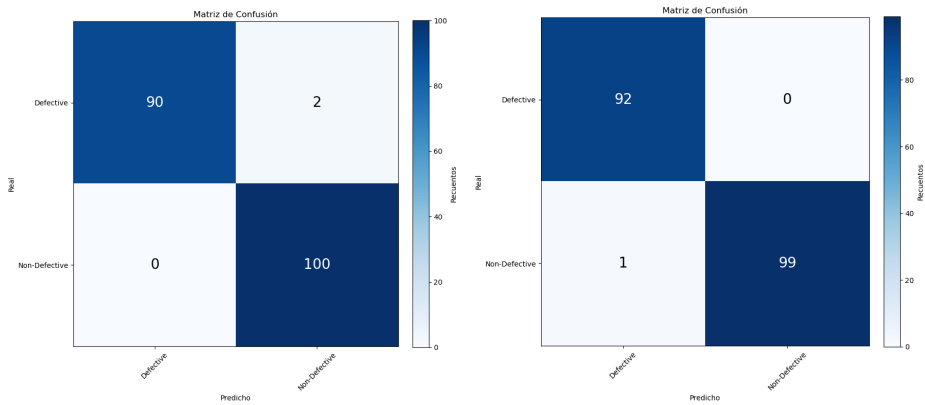
(b) Val loss modelo 5 y 6.

Figura 3.9: Curvas de entrenamiento del grupo 3.

que las que se emplean durante el entrenamiento. El procedimiento inicia con un previo procesamiento, como son: cargar las imágenes³, disminuir la resolución, normalizarlas y extraer la ROI a través del modelo U-Net 1 seleccionado, a fin de que los modelo pueda interpretarlas, evaluarlas y clasificarlas.

3.2.4. Primer conjunto de prueba

Luego de evaluar el conjunto de imágenes de prueba de la carpeta “*Test*” dentro de los clasificadores, las predicciones son registradas manualmente en una hoja de cálculo con el fin de realizar una comparación con los valores reales y generar las correspondientes matrices de confusión de los 6 clasificadores, presentadas en las Fig. 3.10 - 3.12.



(a) Clasificador 1.

(b) Clasificador 2.

Figura 3.10: Matrices de confusión del grupo 1 - Primer conjunto de imágenes de prueba.

³Se observa que, al momento de cargar las imágenes a través de la librería OpenCV, los resultados de predicción varían; no obstante, al emplear el módulo `Utils` de la biblioteca `Keras`, este error fue solventado. La causa más probable es que, durante el entrenamiento se utiliza la función `flow_from_directory` para cargar y procesar lotes de imágenes desde un directorio.

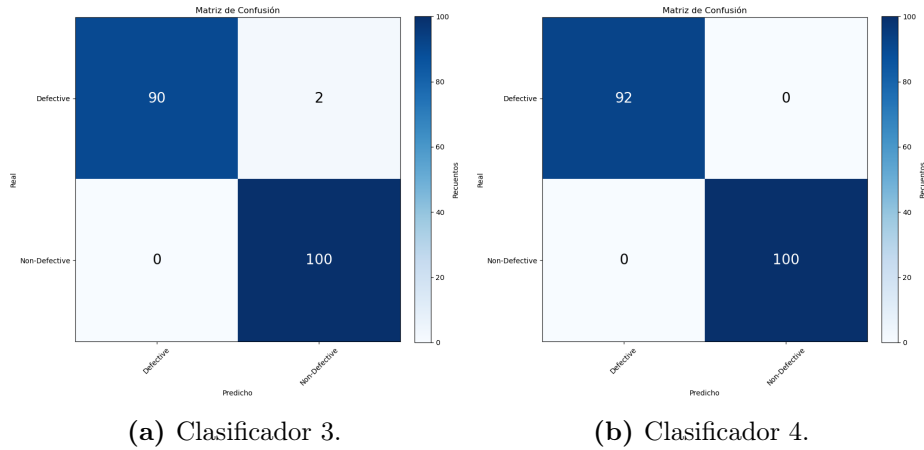


Figura 3.11: Matrices de confusión del grupo 2 - Primer conjunto de imágenes de prueba.

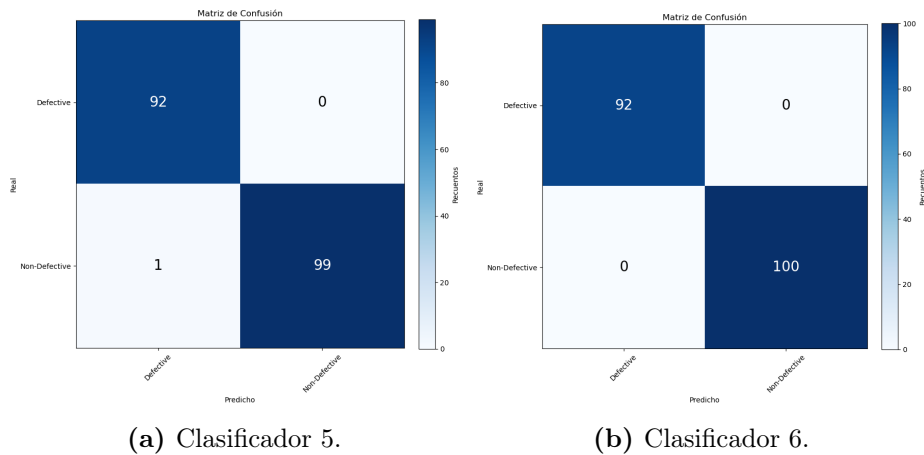


Figura 3.12: Matrices de confusión del grupo 3 - Primer conjunto de imágenes de prueba.

Con base en los datos contenidos en las matrices y las ecuaciones 3.3 - 3.7, se llevó a cabo la cuantificación de las métricas de rendimiento, las cuales se presentan en la Tabla 3.7. Al obtener las métricas, se observan diferencias mínimas entre los clasificadores 1, 2, 3 y 5. Por otra parte, los tiempos de ejecución exhibieron variaciones en cada grupo, con un aproximado de: 1 segundo entre los modelos del grupo uno, 2 segundos entre los modelos del grupo dos, 1 segundo entre los modelos del grupo 3; siendo el clasificador 6 el que registra un tiempo de ejecución menor que los demás.

3.2.5. Segundo conjunto de prueba

Hasta este punto, los modelos 4, 5 y 6 presentan una capacidad óptima para interpretar y clasificar el primer conjunto de imágenes de prueba. Sin embargo, se lleva a cabo una evaluación adicional haciendo uso de un segundo conjunto de imágenes adquiridas por un dispositivo móvil. El segundo conjunto de pruebas constó de 189 imágenes, distribuidas en 85 pertenecientes a la clase defectuosa y 104 a la clase sin defecto. En vista de los recursos computacionales disponibles, las imágenes fueron ingresadas en un lote de 8 imágenes a la vez. Tras obtener las matrices de confusión (ver Fig. 3.13 - 3.15) y posteriormente realizar el cálculo de las métricas de rendimiento (ver Tabla 3.8) se observaron varias diferencias considerables.

Los resultados revelan disparidades significativas en todas las métricas. En cuanto a la mé-

Tabla 3.7: Métricas de rendimiento de los clasificadores - Primer conjunto de imágenes de prueba.

Modelo	Precisión	Recall	Esp.	F-score	Accuracy	Tiempo de ejecución (s)	T. prom. por img. (ms)
1	1.0000	0.9783	1.0000	0.9890	0.9896	63	326
2	0.9892	1.0000	0.9900	0.9946	0.9948	64	333
3	1.0000	0.9783	1.0000	0.9890	0.9896	65	337
4	1.0000	1.0000	1.0000	1.0000	1.0000	63	329
5	0.9892	1.0000	0.9900	0.9946	0.9948	63	330
6	1.0000	1.0000	1.0000	1.0000	1.0000	62	320

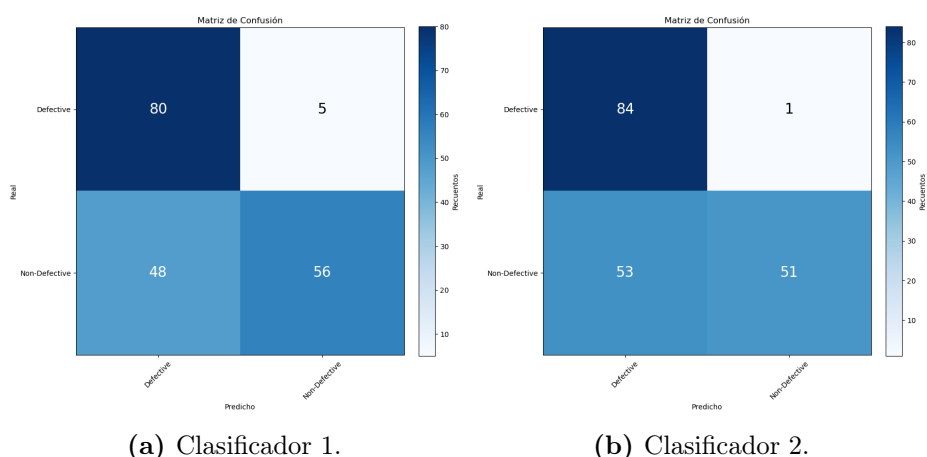


Figura 3.13: Matrices de confusión del grupo 1 - Segundo conjunto imágenes de prueba.

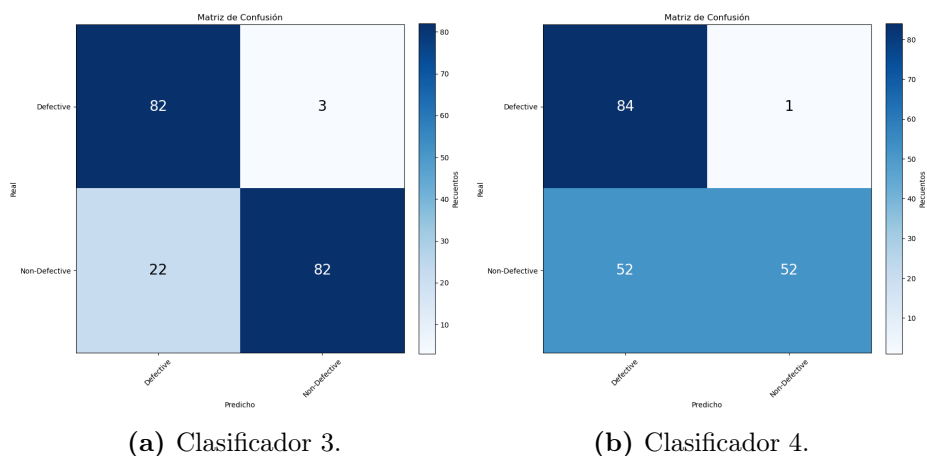


Figura 3.14: Matrices de confusión del grupo 2 - Segundo conjunto imágenes de prueba.

trica de *precisión*, se evidenció una disminución significativa en la capacidad de evitar falsos positivos, especialmente en los grupos 1 y 2. Al examinar el *recall*, se constató una leve reducción en la eficacia de identificar correctamente las chips de papa defectuosas en los tres grupos.

Sin embargo, al calcular la *especificidad*, se observó que la habilidad de evitar falsos negativos resultó considerablemente reducida en los grupos 1 y 2. En cuanto a la capacidad general de predicción entre clases, se constató que los grupos 1 y 2 no superaron el 86%. Con respecto al grupo 3, los valores demostraron una predicción favorable.

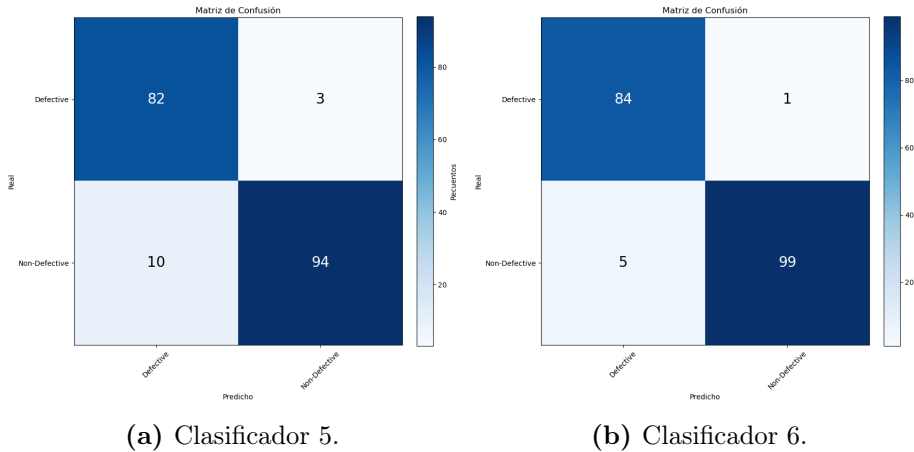


Figura 3.15: Matrices de confusión del grupo 3 - Segundo conjunto imágenes de prueba.

Tabla 3.8: Métricas de rendimiento de los clasificadores - Segundo conjunto imágenes de prueba.

Modelo	Precisión	Recall	Esp.	F1-score	Accuracy	Tiempo de ejecución (s)	T. prom. por img. (ms)
1	0.6250	0.9412	0.5385	0.7512	0.7196	86	454
2	0.6131	0.9882	0.4904	0.7568	0.7143	77	408
3	0.7885	0.9647	0.7885	0.8677	0.8677	78	411
4	0.6176	0.9882	0.5000	0.7602	0.7196	86	455
5	0.8913	0.9647	0.9038	0.9266	0.9312	72	382
6	0.9438	0.9882	0.9519	0.9655	0.9683	75	395

Dada la importancia de minimizar tanto la pérdida de productos en buen estado como la presencia de productos defectuosos, resulta fundamental controlar los resultados de falsos negativos (FN) y positivos (FP) de manera que se reduzca al mínimo su presencia. Siguiendo este criterio de evaluación, los grupos 1 y 2 son descartados. Aunque los modelos del grupo 3 presentan resultados satisfactorios, es necesario considerar el tiempo de ejecución y el tiempo de procesamiento por imagen para obtener una comprensión más completa del rendimiento de los modelos e identificar el clasificador más robusto.

3.2.6. Selección de modelo

Después de evaluar los modelos binarios, se determina que el modelo que demuestra un rendimiento y equilibrio óptimo basado en las métricas *F1-score*, *accuracy* y tiempo de ejecución es el número 6 (ver Tabla 3.9). De un conjunto de 189 imágenes, divididas en 2 clases: 104 normales y 85 defectuosas; el modelo logró predecir correctamente 84 imágenes de chips de papa como defectuosas (TP) y 99 como normales (TN). Sin embargo, se produjeron clasificaciones erróneas, una imagen de chips de papa de la categoría defectuosa fue clasificada como normal (FN) y, cinco imágenes de chips de papa pertenecientes a la clase normal fueron interpretadas como defectuosas (FP).

El modelo demuestra una *precisión* del 94.38%; es decir, que de un total de 89 imágenes predichas como defectuosas, sólo 5 no pertenecían a esa clase, demostrando una disminución en la proporción de falsos positivos. Además, al registrar una tasa de *recall* del 98.82%, de las 85

imágenes categorizadas como chips defectuosas, 84 fueron predichas correctamente, lo que indica una habilidad excepcional para detectar chips defectuosas. Al presentar una *especificidad* de 95.19 %, indica que 99 de las 104 imágenes predichas como no defectuosas realmente pertenecían a esa clase, lo que se traduce en una incidencia reducida de falsos negativos.

Con un valor de 0.9655 del *F1-score* cercano a 1, el modelo 6 posee un rendimiento equilibrado entre la *precisión* y *recall*, aún cuando se evidencia un desequilibrio entre los datos de las clases. Al comparar el tiempo de ejecución se observa una variación de 3 segundos, mientras que en el tiempo de procesamiento por imagen la diferencia fue de 13 milisegundos. Finalmente, el clasificador seleccionado registra un *accuracy* del 96.83 %, es decir, de un total de 189 imágenes el modelo clasificó erróneamente 6 (ver Fig. 3.16), lo que sugiere un desempeño general aceptable para la tarea de clasificación de chips de papa normales y defectuosas.

Tabla 3.9: Métricas de rendimiento del grupo 3 - Segundo conjunto imágenes de prueba.

Modelo	Precisión	Recall	Esp.	F1-score	Accuracy	Tiempo de ejecución (s)	T. prom. por img. (ms)
5	0.8913	0.9647	0.9038	0.9266	0.9312	72	382
6	0.9438	0.9882	0.9519	0.9655	0.9683	75	395

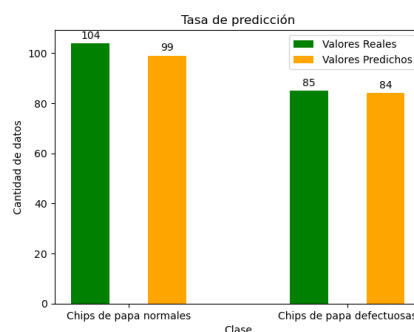


Figura 3.16: Número de acierto del modelo 6 con el segundo conjunto de pruebas.

3.2.7. Comparación de técnicas

Para demostrar la capacidad de clasificación que posee la CNN en combinación del modelo U-Net frente a imágenes que poseen ruidos⁴, se realiza una comparación con una CNN cuyos datos de entrenamiento fueron procesados por el método detallado en [45]. Luego de evaluar el segundo conjunto de imágenes, las matrices (ver Fig. 3.17) obtenidas reflejan una diferencia notable de evitar falsos positivos y falsos negativos.

Con respecto a las métricas de rendimiento y eficiencia computacional, los resultados reflejan diferencias considerables, como se indica en la Tabla 3.10. Se observa diferencias del 11.00 %, 12.50 % y 10.58 % en el *F1-score*, *especificidad* y *accuracy*, respectivamente. Si bien, el modelo CNN - HSV presenta tiempos de procesamiento menor, el consumo de memoria resulta ser mayor. Aunque estos valores reflejan ser mejores al modelo desarrollado, su capacidad de clasificación muestra un nivel inferior.

⁴Este tipo de ruidos son descritos en la sección 2.3

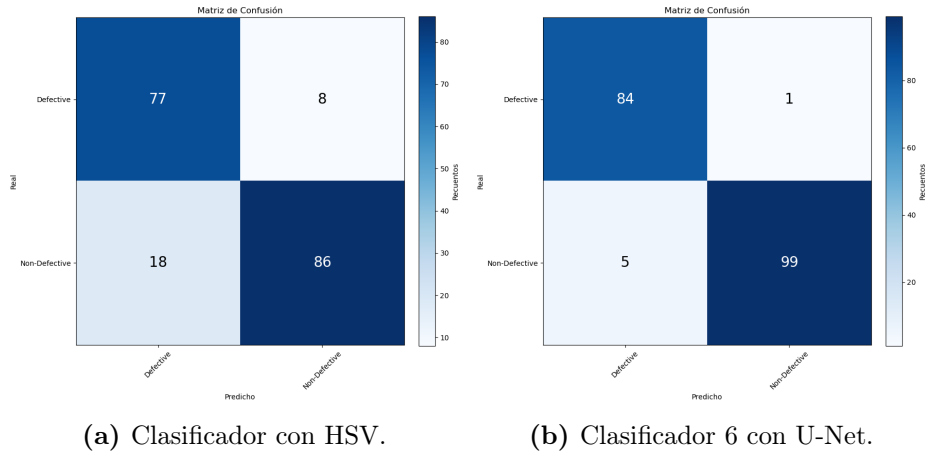


Figura 3.17: Comparación de matrices de confusión - Segundo conjunto de imágenes de prueba.

Tabla 3.10: Métricas de rendimiento - Segundo conjunto imágenes de prueba.

Modelo	Precisión	Recall	Esp.	F1-score	Accuracy	Tiempo de ejecución (s)	T. prom. por img. (ms)	Consumo de memoria (MiB)
CNN - HSV	0.8105	0.9059	0.8269	0.8556	0.8624	40	213	95.53
CNN - UNet	0.9438	0.9882	0.9519	0.9655	0.9683	73	385	1.07

3.3. Análisis del modelo U-Net 2

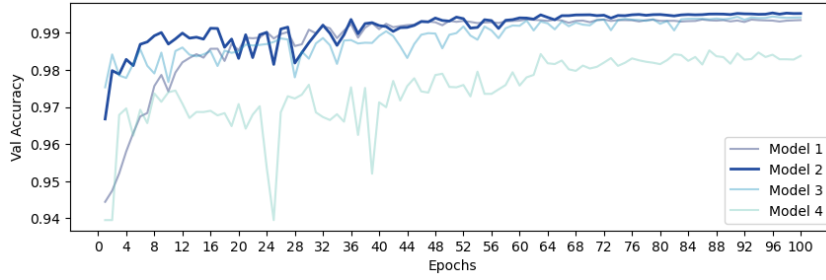
3.3.1. Resultados de entrenamiento

Para evaluar el rendimiento y seleccionar el modelo adecuado, se emplea un procedimiento similar al descrito en la sección 3.1. En esta etapa del estudio, se realiza una comparación de cuatro modelos U-Net analizando el comportamiento de las curvas obtenidas (ver Fig. 3.18) tras la finalización del entrenamiento de cada modelo, así como los coeficientes de similitud *dice* y el *IoU*.

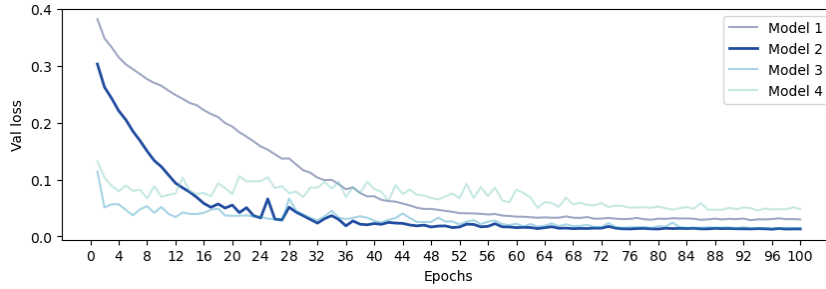
Si bien, los resultados demostraron que con un *lr. max* de 0.0024 el modelo 2 obtuvo las mejores métricas (ver Tabla 3.11), para validar su desempeño, fue imprescindible evaluar la capacidad de predicción utilizando el conjunto de imágenes de la clase defectuosa contenidas en la carpeta “*Test*”.

Tabla 3.11: Resultados del entrenamiento de los modelos U-Net 2.

Modelo	lr	lr. max	val. accuracy	val. loss	dice	IoU
1	0.0001	0.0003	0.99363	0.02865	0.94147	0.88941
2	0.0001	0.0024	0.99538	0.01276	0.95768	0.91879
3	0.0001	0.0552	0.99442	0.01412	0.94899	0.90294
4	0.0001	0.4417	0.98529	0.04544	0.87665	0.78039



(a) Val. accuracy U-Net 2.



(b) Val. loss U-Net 2.

Figura 3.18: Curvas de entrenamiento - U-Net 2.

3.3.2. Selección del modelo U-Net 2

En contraste con el procedimiento de la sección 3.1, las máscaras binarias correspondientes a la clase defectuosa se obtienen a través de un método que convierte las anotaciones de segmentación almacenadas en archivos JSON a un formato visual estructurado. Esta conversión ha permitido disponer de datos que sirvieron de referencia para evaluar las máscaras binarias generadas por cada modelo. En cuanto a las máscaras predichas, por cada modelo se almacenan en rutas específicas.

Tras comparar las predicciones mediante un análisis estadístico (ver Tablas 3.12 y 3.13), se observó que las medianas de los coeficientes dice e IoU no superaron a 0.90. Adicionalmente, los diagramas de cajas y bigotes (ver Fig. 3.19) revelaron indicios de variabilidad inherente en la calidad de predicción, con una cantidad significativa de valores atípicos en el rango de 7 – 10 y 6 – 8 de entre 92 máscaras binarias evaluadas.

Tabla 3.12: Resultados estadísticos de los coeficientes *dice* - U-Net 2.

Modelo	lr	lr. max	Mediana dice	Rango inter-cuartil dice	Máx. Dice	Mín. dice	Val. atípicos dice
1	0.0001	0.0003	0.86210	0.07829	0.97212	0.51855	10
2	0.0001	0.0024	0.90613	0.04847	0.97243	0.60313	8
3	0.0001	0.0552	0.88351	0.08303	0.97587	0.43267	9
4	0.0001	0.4417	0.88169	0.07706	0.96281	0.54280	7

A pesar de que los diagrama de cajas y bigotes del modelo 2 reflejaron la presencia de valores atípicos en los conjuntos de datos obtenidos, los rangos intercuartiles asociados resultaron ser menores. Además, la parte inferior resultó tener una longitud mayor que la superior; esto de-

Tabla 3.13: Resultados estadísticos de los coeficientes IoU - U-Net 2.

Modelo	lr	lr. max	Mediana IoU	Rango inter-cuartil IoU	Max. IoU	Min. IoU	Val atípicos IoU
1	0.0001	0.0003	0.75762	0.11955	0.94575	0.35003	8
2	0.0001	0.0024	0.82838	0.08000	0.94635	0.43177	7
3	0.0001	0.0552	0.79133	0.13211	0.95287	0.27606	7
4	0.0001	0.4417	0.78842	0.12057	0.92829	0.37249	6

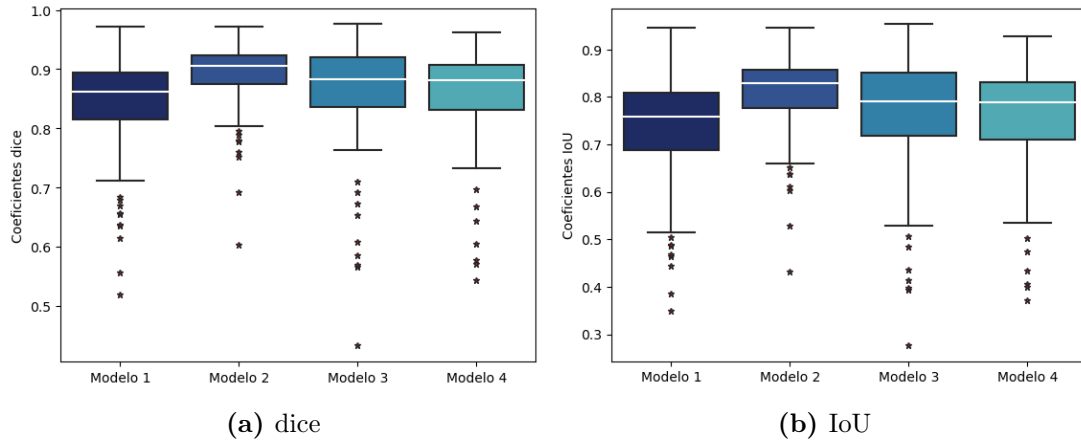


Figura 3.19: Diagramas de cajas de los modelos - U-net 2.

muestra que la mayoría de los casos poseen coeficientes altos, sugiriendo una menor distribución de datos, es decir, una mayor consistencia en las predicciones.

3.3.3. Optimización del modelo U-Net 2 elegido

En este punto, el modelo seleccionado fue el segundo. Para optimizar aún más la precisión de las máscaras y reducir la cantidad de valores atípicos, se decidió incrementar el tamaño del conjunto de datos de entrenamiento de 160 a 320 imágenes con sus respectivas máscaras binarias. Posteriormente, se llevó a cabo dos entrenamientos: el primero con el mismo número de épocas mientras que el segundo con 200 épocas. Al finalizar los entrenamientos, los resultados presentados en las Tablas 3.14 y 3.15 demostraron una mejoría en la mayoría de las medidas estadísticas evaluadas.

Tabla 3.14: Resultados estadísticos de los coeficientes IoU con aumento de épocas y datos de entrenamiento.

Modelo	lr	lr. max	Mín. IoU	Mediana IoU	Máx. IoU	Rango	Rango inter-cuartil IoU	Val. atípicos IoU
2	0.0001	0.0024	0.43177	0.82838	0.94635	0.51457	0.08000	7
2_1	0.0001	0.0024	0.64478	0.85066	0.94312	0.29834	0.07235	2
2_2	0.0001	0.0024	0.49510	0.85856	0.95380	0.45870	0.08394	6

Tabla 3.15: Resultados estadísticos de los coeficientes dice con aumento de épocas y datos de entrenamiento.

Modelo	lr	lr. max	Mín. dice	Mediana dice	Máx. Dice	Rango	Rango intercuartil dice	Val. atípicos dice
2	0.0001	0.0024	0.60313	0.90613	0.97243	0.36930	0.04847	8
2_1	0.0001	0.0024	0.78403	0.91930	0.97073	0.18670	0.04255	3
2_2	0.0001	0.0024	0.66230	0.92390	0.97635	0.31406	0.04867	8

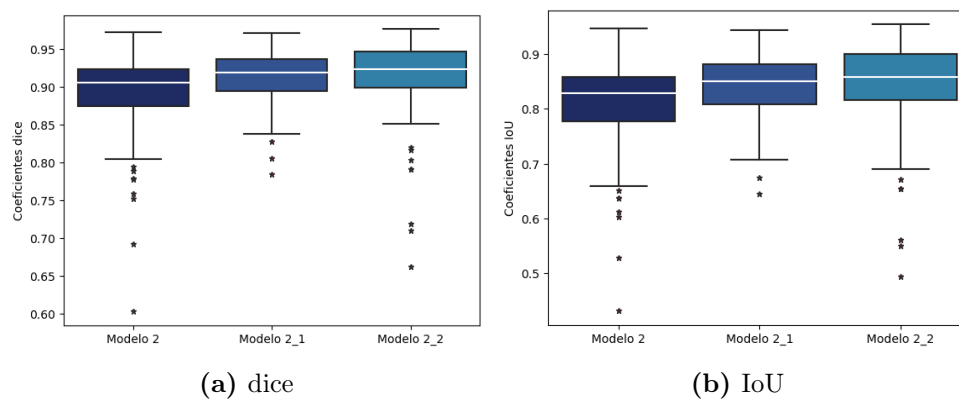


Figura 3.20: Diagrama de cajas de los modelos - U-Net 2 con aumento de datos y épocas.

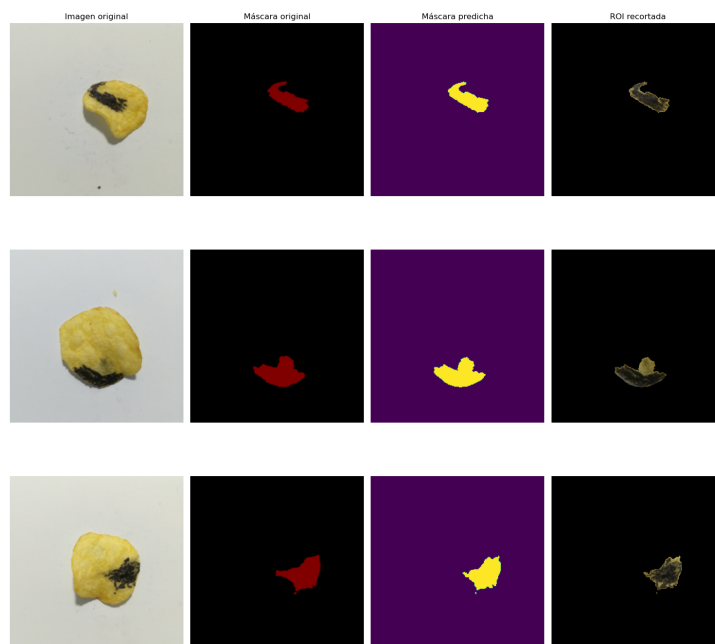


Figura 3.21: Muestra de segmentación semántica - modelo U-Net 2 seleccionado.

En particular, la cantidad de valores atípicos que poseía el modelo 2, se redujeron notoriamente en el modelo 2_1. En el caso del modelo 2_2, aun cuando el tiempo de entrenamiento fue mayor, la reducción fue mínima; esto puede ser visualizado en los diagramas de cajas y bigotes de la Fig. 3.20. Una muestra de la capacidad de predicción del método de segmentación seleccionado es presentado en la Fig. 3.21.

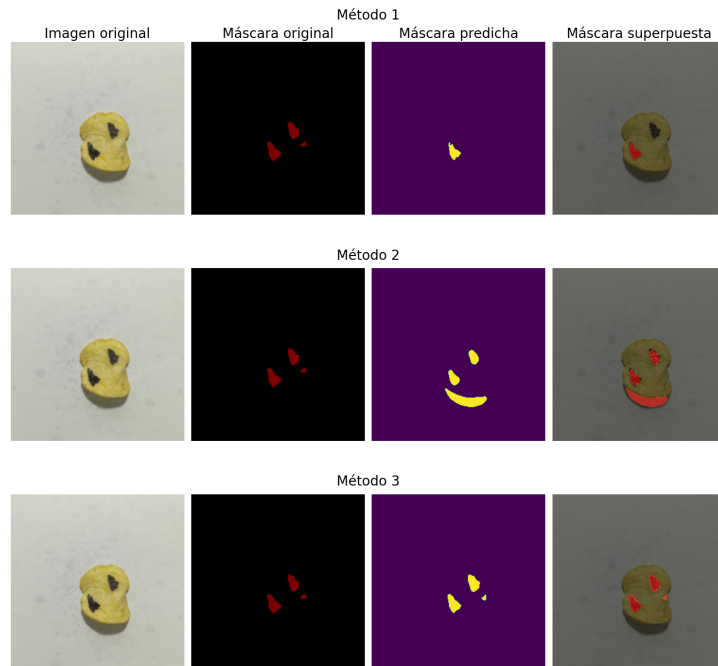


Figura 3.22: Resultado de segmentación de ROI por cada método - U-net2.

3.3.4. Comparación de técnicas

Para comparar la calidad de predicción de cada técnica de segmentación, se utilizaron varias imágenes de la carpeta “Test”. La Fig. 3.22 ilustra los resultados por cada uno de los métodos que fueron comparados, en donde, visualmente, las máscaras generadas reflejaron diferencias significativas.

En el caso del método por umbralización, en la mayoría de los casos las máscaras no segmentaban toda el área defectuosa debido a que el valor de umbral es estático. Al emplear el espacio HSV, se observó que el rango de valores establecido no solo segmentaba el área defectuosa, sino que también afectaba a la región con sombra.

Por otra lado, el método U-Net desarrollado carecía de los defectos que se presentaron en los otros métodos. Para validar estas diferencias, los coeficientes calculados (ver Tabla 3.16) demostraron que, en todas las máscaras binarias generadas, el método desarrollado obtuvo las mejores métricas, posicionándolo como el candidato ideal para la tarea de segmentación de la ROI con defecto.

3.4. Análisis del método por conteo de píxeles

3.4.1. Criterios de evaluación

Al ser un método sujeto a interpretación debido al ajuste del umbral realizado por el usuario y, a la forma en como se han obtenido los porcentajes reales a través de las máscaras binarias correspondientes a la clase defectuosa del primer conjunto de prueba, se ha excluido del análisis. Sin embargo, para validar el rendimiento del método a detalle, se toma como referencia los resultados de las matrices obtenidas (ver Fig. 3.12b y 3.17b) por el modelo CNN 6 con el primer y segundo conjunto de pruebas. Adicionalmente, se considera utilizar métricas de rendimiento del apartado 3.2.1.

Tabla 3.16: Resultados obtenidos por cada técnica de segmentación

Método	Muestra	dice	Val. máx dice	IoU	Val. máx IoU
1	1	0.62987		0.45971	
2		0.58521		0.41364	
3		0.83821	0.83821	0.72148	0.72148
1	2	0.95717		0.91785	
2		0.63397		0.46409	
3		0.96437	0.96437	0.93120	0.93120
1	3	0.68403		0.51979	
2		0.78589		0.64729	
3		0.88996	0.88996	0.80174	0.80174
1	4	0.90115		0.82008	
2		0.58976		0.41820	
3		0.95072	0.95072	0.90607	0.90607
1	5	0.87379		0.77586	
2		0.64949		0.48092	
3		0.92853	0.92853	0.86659	0.86659

Las pruebas se realizan con cuatro umbrales diferentes: 5, 10, 25 y 40%. Estos valores han permitido identificar que imágenes fueron categorizadas como defectuosas y cuales no; además de compararlas con la clasificación predicha por el modelo CNN seleccionado.

3.4.2. Resultados de evaluación - primer conjunto de prueba

Umbral del 5%

La matriz de la Fig 3.23 indica que el método ha clasificado correctamente todas las imágenes, puesto que, carece de falsos positivos como negativos. Esto es evidenciado en el *F1-score*, *especificidad* y el *accuracy* presentados en la Tabla 3.17. Por otra parte, en cuanto a la eficiencia computacional, el tiempo de ejecución requerido para procesar todas las imágenes fue de 93.073 segundos.

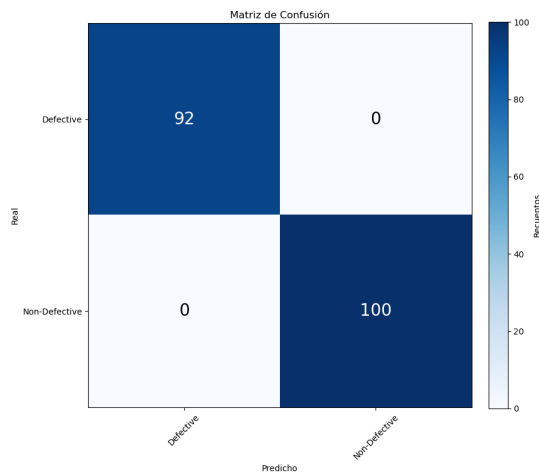


Figura 3.23: Matriz de confusión con umbral 5% - primer conjunto de prueba.

Tabla 3.17: Resultados de las métricas de rendimiento y tiempo de ejecución - método con umbral de 5%.

Métrica	Valor
<i>F1-score</i>	1.00
<i>Especificidad</i>	1.00
<i>Accuracy</i>	1.00
Tiempo de ejecución (s)	93.073

Umbral del 10%

La matriz de la Fig 3.24 evidencia que el método ha clasificado correctamente 84 imágenes de la clase defectuosa y 100 de la clase sin defecto. En este caso, al presentar 8 falsos negativos, el *F1-score* y el *accuracy* se han visto afectados; no obstante, al carecer de falsos positivos, no hubo alteraciones en la *especificidad*, tal como se presenta en la Tabla 3.18. Con respecto a la eficiencia computacional, el tiempo necesario para la ejecución fue de 77.44 segundos.

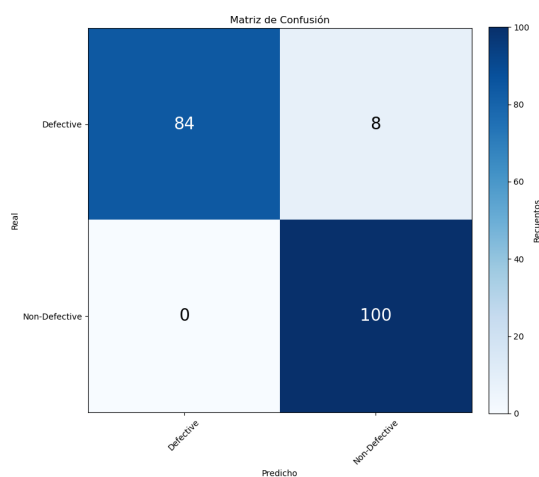


Figura 3.24: Matriz de confusión con umbral 10% - primer conjunto de prueba.

Tabla 3.18: Resultados de las métricas de rendimiento y tiempo de ejecución - método con umbral de 10%.

Métrica	Valor
<i>F1-score</i>	0.9545
<i>Especificidad</i>	1.00
<i>Accuracy</i>	0.9583
Tiempo de ejecución (s)	77.444

Umbral del 25%

La matriz de la Fig 3.25 muestra que el método ha clasificado correctamente 10 imágenes de la clase defectuosa y 100 de la clase sin defecto. Al presentar 82 falsos negativos, el *F1-score* y el *accuracy* se han visto considerablemente afectados. Sin embargo, la *especificidad* permanece igual, tal como se presenta en la Tabla 3.19. En lo que respecta a la eficiencia computacional, se requirieron de 83.031 segundos.

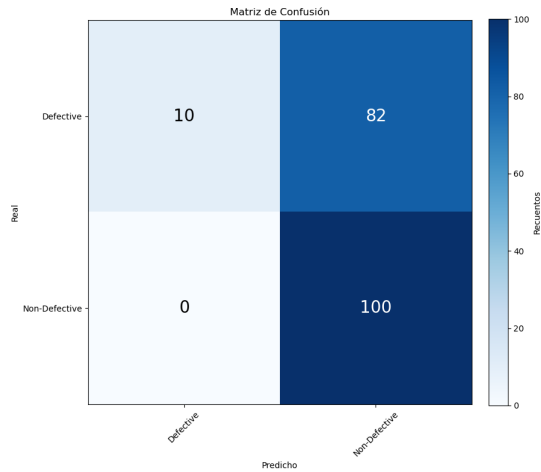


Figura 3.25: Matriz de confusión con umbral 25 % - primer conjunto de prueba.

Tabla 3.19: Resultados de las métricas de rendimiento y tiempo de ejecución - método con umbral de 25 %.

Métrica	Valor
<i>F1-score</i>	0.1961
<i>Especificidad</i>	1.00
<i>Accuracy</i>	0.5729
Tiempo de ejecución (s)	83.031

Umbral de 40 %

La matriz de la Fig 3.26 revela que el método al clasificar correctamente 100 de la clase sin defecto y carecer de falsos positivos, la *especificidad* permanece igual. En cambio, debido a la ausencia de verdaderos positivos y presentar 92 falsos negativos, el *F1-score* y el *accuracy* han sido severamente afectados, tal como se presenta en la Tabla 3.20. El tiempo de ejecución requerido fue de 76.994 segundos.

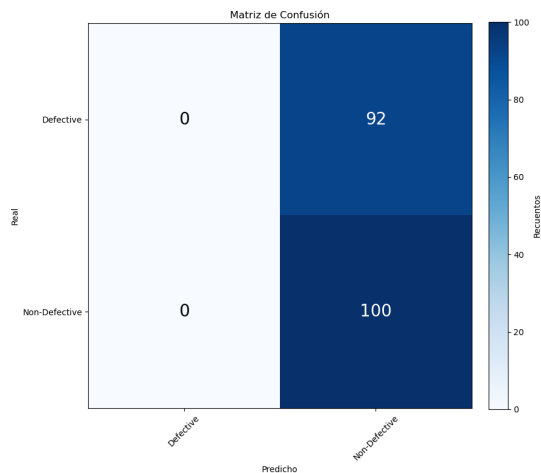


Figura 3.26: Matriz de confusión con umbral 40 % - primer conjunto de prueba.

Tabla 3.20: Resultados de las métricas de rendimiento y tiempo de ejecución - método con umbral de 40 %.

Métrica	Valor
<i>F1-score</i>	0.00
<i>Especificidad</i>	1.00
<i>Accuracy</i>	0.5208
Tiempo de ejecución (s)	76.994

3.4.3. Resultados de evaluación - segundo conjunto de prueba

Umbral del 5 %

Los resultados que se presentan en la matriz de la Fig. 3.27 revelan que de las 189 imágenes, 83 se han clasificado correctamente como defectuosas y, 101 sin defecto. Además, debido a la presencia de falsos positivos y negativos, las métricas de rendimiento resultaron afectadas, tal como se evidencia en la Tabla 3.21. A diferencia de los resultados previos, se ha requerido un tiempo mayor de ejecución, con un valor de 124.347 segundos.

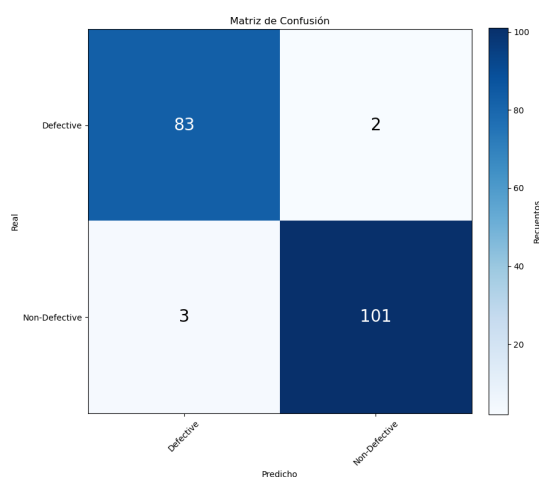


Figura 3.27: Matriz de confusión con umbral 5 % - segundo conjunto de prueba.

Tabla 3.21: Resultados de las métricas de rendimiento y tiempo de ejecución - método con umbral de 5 %.

Métrica	Valor
<i>F1-score</i>	0.9708
<i>Especificidad</i>	0.97120
<i>Accuracy</i>	0.9735
Tiempo de ejecución (s)	124.347

Umbral del 10 %

Los resultados que se muestran en la matriz de la Fig. 3.28 indican que: 101 imágenes han sido clasificadas correctamente en la clase sin defecto y 73 en la clase defectuosa. Asimismo, debido a la presencia de falsos positivos y negativos, las métricas de rendimiento (ver Tabla 3.22) se vieron comprometidas. En este caso, el tiempo de ejecución fue menor, alcanzando los 107.632 segundos.

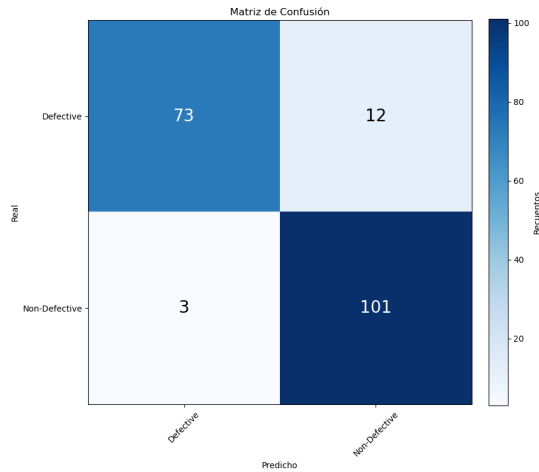


Figura 3.28: Matriz de confusión con umbral 10% - segundo conjunto de prueba.

Tabla 3.22: Resultados de las métricas de rendimiento y tiempo de ejecución - método con umbral de 10%.

Métrica	Valor
<i>F1-score</i>	0.9068
<i>Especificidad</i>	0.9712
<i>Accuracy</i>	0.9206
Tiempo de ejecución (s)	107.632

Umbral del 25 %

En la matriz de la Fig. 3.29 se observa una disminución de verdaderos positivos y un aumento de falsos negativos. De un total de 189 imágenes, 54 han sido identificadas como defectuosas y 103 como no defectuosas. La presencia de estos errores han comprometido las métricas de rendimiento, como se puede ver en la Tabla 3.23. Con respecto al tiempo requerido para procesar todas las imágenes del umbral anterior, la diferencia fue mínima, alcanzando los 107.535 segundos.

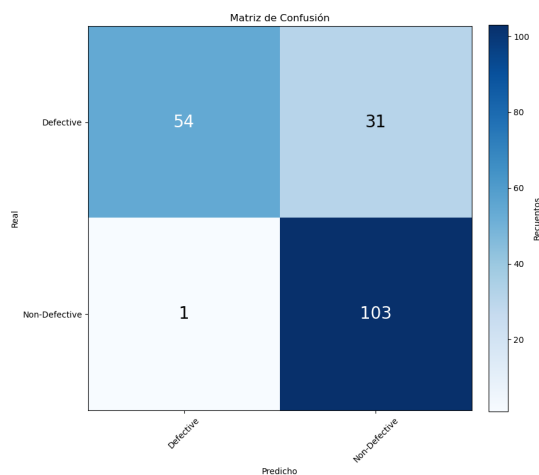


Figura 3.29: Matriz de confusión con umbral 25% - segundo conjunto de prueba.

Tabla 3.23: Resultados de las métricas de rendimiento y tiempo de ejecución - método con umbral de 25 %.

Métrica	Valor
<i>F1-score</i>	0.7068
<i>Especificidad</i>	0.9904
<i>Accuracy</i>	0.7937
Tiempo de ejecución (s)	107.535

Umbral de 40 %

Como última instancia, los resultados que se presentan en la matriz de confusión (ver Fig. 3.30) muestran que, de 150 imágenes que han sido clasificadas correctamente, 47 pertenecen a la clase defectuosa y 103 a la clase sin defecto. Del mismo modo, la presencia de falsos negativos y positivos, refleja una disminución en la capacidad de predicción, como se puede observar en las métricas de la Tabla 3.24. Si bien, las métricas presentan una mejoría con respecto a los resultados del umbral previo, el tiempo de ejecución aumentó, ascendiendo a 118.540 segundos.

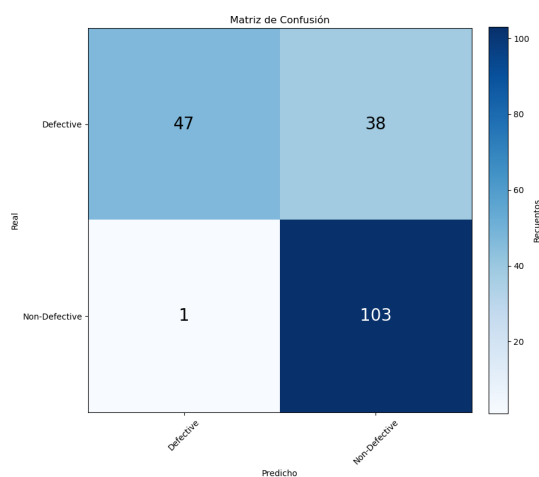


Figura 3.30: Matriz de confusión con umbral 40 % - segundo conjunto de prueba.

Tabla 3.24: Resultados de las métricas de rendimiento y tiempo de ejecución - método con umbral de 40 %.

Métrica	Valor
<i>F1-score</i>	0.7714
<i>Especificidad</i>	0.9904
<i>Accuracy</i>	0.8307
Tiempo de ejecución (s)	118.540

3.4.4. Comparación con el clasificador CNN

Las Tablas 3.26 y 3.27 presentan una comparación detallada de los resultados de clasificación obtenidos mediante los dos métodos de detección de defectos de chips de papa: el método por conteo de píxeles a diferentes umbrales y el clasificador CNN. La evaluación se llevó a cabo utilizando una muestra de 8 imágenes aleatorias correspondientes a los dos conjuntos de pruebas.

Al analizar los resultados del primer conjunto de pruebas, las predicciones proporcionadas por el clasificador CNN reflejan una precisión ideal. Por otro lado, los resultados derivados del

método por conteo presentan errores; a pesar de esto, a medida que el valor del porcentaje disminuye, estos errores se reducen. Estas observaciones se validan con las métricas de rendimiento que se presentan en la Tabla 3.25.

Algo similar sucede con los resultados del segundo conjunto de pruebas, en donde las imágenes en cierta medida, representan datos en un ambiente no ideal. Debido a esto, todas las métricas se han visto afectadas, no obstante, esta tendencia permanece: a medida que el umbral del porcentaje disminuye, las métricas mejoran, tal como se muestra en la Tabla 3.25.

Tabla 3.25: Comparación de métricas del método por conteo de píxeles.

Conjunto de prueba	Umbral (%)	F1-score	Esp.	Accuracy (%)
Primero conjunto	40	0.00	1.00	52.08
	25	0.1961	1.00	57.29
	10	0.9545	1.00	95.83
	5	1.00	1.00	100.00
Segundo conjunto	40	0.7714	0.9904	83.07
	25	0.7068	0.9904	79.37
	10	0.9068	0.9712	92.06
	5	0.9708	0.97120	97.35

Tabla 3.26: Resultados de predicción con muestras aleatorias y umbrales de defecto - primer conjunto de imágenes de prueba.

















Muestra	Clase real	Pred. CNN	Pred. met. Pixel (5 %)	Pred. met. Pixel (10 %)	Pred. met. Pixel (25 %)	Pred. met. Pixel (40 %)
	Defectuosa	Defectuosa	Defectuosa	Defectuosa	Sin defecto	Sin defecto
	Sin defecto	Sin defecto	Sin defecto	Sin defecto	Sin defecto	Sin defecto
	Defectuosa	Defectuosa	Defectuosa	Sin defecto	Sin defecto	Sin defecto
	Defectuosa	Defectuosa	Defectuosa	Defectuosa	Sin defecto	Sin defecto
	Sin defecto	Sin defecto	Sin defecto	Sin defecto	Sin defecto	Sin defecto
	Defectuosa	Defectuosa	Defectuosa	Defectuosa	Sin defecto	Sin defecto
	Defectuosa	Defectuosa	Defectuosa	Defectuosa	Sin defecto	Sin defecto
	Sin defecto	Sin defecto	Sin defecto	Sin defecto	Sin defecto	Sin defecto

Tabla 3.27: Resultados de predicción con muestras aleatorias y umbrales de defecto - segundo conjunto de imágenes de prueba.

Muestra	Clase real	Pred. CNN	Pred. met. Pixel (5 %)	Pred. met. Pixel (10 %)	Pred. met. Pixel (25 %)	Pred. met. Pixel (40 %)
	Sin defecto	Sin defecto	Sin defecto	Sin defecto	Sin defecto	Sin defecto
	Sin defecto	Sin defecto	Sin defecto	Sin defecto	Sin defecto	Sin defecto
	Defectuosa	Defectuosa	Defectuosa	Defectuosa	Sin defecto	Sin defecto
	Sin defecto	Sin defecto	Sin defecto	Sin defecto	Sin defecto	Sin defecto
	Defectuosa	Defectuosa	Defectuosa	Defectuosa	Sin defecto	Sin defecto
	Defectuosa	Defectuosa	Defectuosa	Defectuosa	Sin defecto	Sin defecto
	Defectuosa	Defectuosa	Defectuosa	Defectuosa	Sin defecto	Sin defecto
	Defectuosa	Defectuosa	Defectuosa	Defectuosa	Defectuosa	Defectuosa

Capítulo 4

Conclusiones, recomendaciones y trabajos futuros

4.1. Conclusiones

Luego de una serie de pruebas exhaustivas, es evidente que, los modelos basados en arquitecturas CNN han demostrado ser herramientas eficaces para analizar, identificar y extraer información relevante de productos procesados, pero sobre todo, automatizar este proceso. En el caso de las chips de papa, una vez definido los parámetros e hiperparámetros de la red, fue posible visualizar como la CNN interpretaba y procesaba la información a través de las capas convoluciones; estas características discriminatorias fueron representadas a través de mapas de activación. Dependiendo de la complejidad del red, cantidad y variedad en los datos, estos tensores resultaron contener información más relevante, siendo de utilidad para visualizar cómo la CNN está interpretando las imágenes.

Uno de los puntos más relevantes de este estudio fue demostrar que tres modelos CNN desarrollados en dos frameworks diferentes (PyTorch y Keras) pueden interactuar entre sí, a fin de resolver el problema planteado. Al utilizar una arquitectura U-Net fue posible no solo de preparar los datos de entrenamiento sino de disminuir considerablemente la presencia de falsos positivos y negativos, mientras que con una CNN se entrenó un modelo que permitió evaluar el estado de las chips de papa y clasificarlas en dos clases. Por otra parte, con la obtención de un segundo modelo U-Net, se abrió la oportunidad de desarrollar un método de clasificación basado en el conteo de píxeles.

Al introducir diferentes hiperparámetros durante el entrenamiento, fue posible realizar una comparación cuantitativa del rendimiento. En el caso de los modelos U-Net, se observó que introducir un método de aceleración, en este caso un *lr. max* dinámico, permitió encontrar un equilibrio entre la convergencia rápida y estabilizar el proceso reduciendo gradualmente el *lr* que se define dentro del optimizador, obteniendo modelos robustos capaces de segmentar la ROI de manera automática. Por otra parte, al aumentar el número de capas convolucionales e introducir varias combinaciones de dropout dentro de la configuración de la CNN, el rendimiento de generalización mejoró.

El clasificador CNN obtenido logró un excelente rendimiento de predicción, alcanzando un *F1-score* de 1.00, un *accuracy* del 100.00% y un tiempo promedio de procesamiento por imagen de 320 ms con el primer conjunto de pruebas. En cambio, aunque con el segundo conjunto de prueba, estas métricas disminuyeron, alcanzaron una puntuación aceptable, con un *F1-score* de 0.9655, un *accuracy* del 96.83 y un tiempo promedio por imagen de 395 ms.

Finalmente, se observó que disminuir el umbral de porcentaje mejoró significativamente el resultado de clasificación del método por conteo de píxeles, alcanzando métricas similares al clasificador CNN, con un *F1-score* de 1.00, un *accuracy* del 100.00 % y 485 ms por imagen con el primer conjunto de prueba y, un *F1-score* de 0.9708, un *accuracy* del 97.35 % y 658 ms por imagen en el segundo conjunto.

4.2. Recomendaciones

Es evidente que, para obtener un modelo robusto capaz de generalizar nuevos datos, es indispensable poseer una gran variedad de imágenes. Sobre todo, es recomendable trabajar con imágenes que compartan similitud con el escenario donde va a ser implementado el modelo. Debido a que, las CNN se caracterizan por automatizar el proceso de extracción de información relevante del objeto de estudio. Si estas imágenes no comparten alguna similitud, la predicción del modelo se vera comprometido, como se evidenció con el segundo conjunto de imágenes de pruebas, que si bien, pertenecían al objeto de estudio, el entorno en el que se obtuvo los datos no era el adecuado, generando falsos positivos y negativos. Al poseer un entorno controlado, se evitaría realizar procesamientos de imágenes que requieren de una gran cantidad de tiempo, más aún, cuando se trabaja con conjunto de imágenes de terceros.

Aunque los modelos basados en CNN permiten abordar problemas de detección de defectos con resultados satisfactorios; las principales desventajas que se presentan, son los fenómenos de *overfitting* y *underfitting*. Si bien, los resultados obtenidos del modelo con el conjunto de datos de prueba fueron satisfactorios, las gráficas obtenidas durante el entrenamiento no presentaron un crecimiento progresivo, especialmente en el grupo 3, en donde las primeras 20 épocas no se produjeron mejorías.

Por ello, se recomienda aplicar varios métodos que eviten estos problemas, como son: *data augmentation*, *dropout*, aumentar o reducir la complejidad del modelo, experimentar con diferentes hiperparámetros, arquitecturas o *transfer learning*; es decir, encontrar un ajuste que permita a la CNN superar los mínimos locales, converger y alcanzar el mínimo global, mejorando así, el tiempo de entrenamiento y el rendimiento del modelo.

4.3. Trabajos futuros

En vista que, para el desarrollo de todos los modelos se tomó como referencia un conjunto de imágenes en donde el defecto fue simulado a través de la pigmentación con un marcador negro de una parte de chips de papa, resulta imposible ver el comportamiento del modelo frente a defectos reales presentes en las fundas de chips de papa sin marca que se encuentran en el mercado, de modo que, se sugiere crear un dataset de entrenamiento que contenga varias clases: normales, defectos naturales, defectos por cocción, objetos extraños; realizar el entrenamiento y observar la capacidad de predicción de la CNN propuesta.

Ahora bien, para mitigar el tiempo de procesamiento por imagen y consumo computacional, se sugiere disminuir el tamaño de las imágenes que son ingresadas a la CNN para el entrenamiento. Para compensar la pérdida de información por este proceso se debería aumentar el número de imágenes de entrenamiento; eso sí, se debe de tomar en cuenta que la configuración de la CNN no genere algún conflicto. Adicionalmente, ya que los modelos U-Net fueron entrenados con imágenes de 256 x 256, resultarían obsoletos, por lo tanto, también se debe de volver a realizar el entrenamiento o, buscar un proceso que permita disminuir el tamaño de las imágenes que se

generaron luego al eliminar el fondo y aplicar transformaciones, antes de ser ingresadas a la CNN.

A pesar de que se puede simular el proceso de clasificación con solo definir tres directorios: Entrada, defectuosas, sin defecto, se desconoce el comportamiento que tendría en la vida real. Sería prudente que, con la ayuda de una banda transportadora y un detector de objetos de chips de papa, se podría capturar las imágenes de entrada y enviar al clasificador para ser evaluadas en tiempo real.

Por último, aunque el método por conteo de píxeles podría ser una herramienta que ayudaría a compensar las limitaciones de clasificación del modelo CNN debido a que solo posee dos posibles predicciones: defectuosas y sin defecto, sin tener en cuenta la gravedad del área afectada; la información que se empleó para el entrenamiento y validación fueron generadas de manera manual, carece de objetividad y justificación. Para solucionarlo, se sugiere varias alternativas: la participación de personal experto en el área para el etiquetado de las imágenes y evaluadores de alimentos o buscar una herramienta de segmentación que minimice la influencia de interpretación del usuario.

Bibliografía

- [1] M. P. Recalde Sánchez and P. M. Aguirre Mejía, *Plan de marketing con enfoque de sustentabilidad para la producción y comercialización de snacks saludables en base de productos andinos en la provincia de Imbabura*, vol. 6. Cuvillier Gotinga, 2021.
- [2] E. Villacrés-Poveda, G. Zurita-Sorrosora, I. Samaniego-Maigua, and J. Angós-Iturgaiz, “Evaluación del contenido de acrilamida en chips de papa (*solanum tuberosum* l.) elaborados por fritura convencional y al vacío,” *Revista Latinoamericana de la Papa*, vol. 24, no. 1, pp. 34–49, 2020.
- [3] D. de Nutrición, *Análisis de riesgos relativos a la inocuidad de los alimentos. Guía para las autoridades nacionales de inocuidad de los alimentos*. OMS y FAO, 2007.
- [4] M. F. Vivas Maticurena and W. A. Vivar Encalada, “Diseño e implementación de un sistema de control automático con visión artificial y redes neuronales destinado al control de calidad de alimentos,” B.S. thesis, Universidad Politécnica Salesiana, 2022.
- [5] N. A. L. Ibarra, “Proyecto montaje de una empresa procesadora de papa en el municipio de pupiales-nariño,” *Universidad Industrial de Santander, Santander, Trabajo de Grado*, 2012.
- [6] C. Alimentarius, “Código de prácticas para reducir el contenido de acrilamida en los alimentos,” *CAC/RCP*, pp. 67–2009, 2009.
- [7] ARCSA, *Normativa técnica para alimentos procesados*, 2023.
- [8] T. Varzakas and C. Tzia, *Handbook of food processing: food safety, quality, and manufacturing processes*, vol. 35. CRC Press, 2015.
- [9] K. Jürkenbeck and A. Spiller, “Importance of sensory quality signals in consumers’ food choice,” *Food Quality and Preference*, vol. 90, p. 104155, 2021.
- [10] S. Bhattacharya, *Snack Foods: Processing and Technology*. Academic Press, 2022.
- [11] A. Yadav, M. K. Dutta, and R. Burget, “Automated visible range imaging scheme to identify toxic substance from common starchy food,” in *2018 4th International Conference on Computational Intelligence & Communication Technology (CICT)*, pp. 1–6, IEEE, 2018.
- [12] R. Maurya, S. Singh, V. K. Pathak, and M. K. Dutta, “Computer-aided automatic detection of acrylamide in deep-fried carbohydrate-rich food items using deep learning,” *Machine Vision and Applications*, vol. 32, no. 4, p. 79, 2021.
- [13] A. R. Paguay Paguay and P. R. Urgilés Ortíz, “Recuperación de imágenes mediante extracción de blobs aplicando el operador laplaciano de gauss y el kernel gaussiano y desarrollo de un prototipo,” B.S. thesis, Universidad Politécnica Salesiana, 2012.
- [14] F. Mendoza, P. Dejmek, and J. M. Aguilera, “Colour and image texture analysis in classification of commercial potato chips,” *Food Research International*, vol. 40, no. 9, pp. 1146–1154, 2007.

- [15] F. Pedreschi, D. Mery, A. Bunger, and V. Yanez, “Computer vision classification of potato chips by color,” *Journal of Food Process Engineering*, vol. 34, no. 5, pp. 1714–1728, 2011.
- [16] A. Singh, M. K. Dutta, R. Burget, and J. Masek, “Identification of acrylamide in fried potato crisps using image processing in wavelet domain,” in *2015 38th International Conference on Telecommunications and Signal Processing (TSP)*, pp. 545–549, IEEE, 2015.
- [17] M. K. Dutta, A. Singh, and S. Ghosal, “A computer vision based technique for identification of acrylamide in potato chips,” *Computers and Electronics in Agriculture*, vol. 119, pp. 40–50, 2015.
- [18] A. Singh, M. Mishra, M. K. Dutta, and R. Burget, “An imaging method for automated detection of acrylamide in potato chips,” in *2017 4th IEEE Uttar Pradesh Section International Conference on Electrical, Computer and Electronics (UPCON)*, pp. 487–490, IEEE, 2017.
- [19] A. Jahanbakhshi, M. Momeny, M. Mahmoudi, and Y.-D. Zhang, “Classification of sour lemons based on apparent defects using stochastic pooling mechanism in deep convolutional neural networks,” *Scientia Horticulturae*, vol. 263, pp. 109–133, 2020.
- [20] B. Jiang, J. He, S. Yang, H. Fu, T. Li, H. Song, and D. He, “Fusion of machine vision technology and alexnet-cnns deep learning network for the detection of postharvest apple pesticide residues,” *Artificial Intelligence in Agriculture*, vol. 1, pp. 1–8, 2019.
- [21] S. D. Ray, M. K. T. K. Natasha, M. A. Hakim, and F. Nur, “Carrot cure: A cnn based application to detect carrot disease,” in *2022 6th International Conference on Trends in Electronics and Informatics (ICOEI)*, pp. 01–07, IEEE, 2022.
- [22] J. H. Sossa Azuela, “El papel de la inteligencia artificial en la industria 4.0,” in *Inteligencia artificial y datos masivos en archivos digitales sonoros y audiovisuales* (P. O. R. Reséndiz, ed.), pp. 21–58, México D.F., primera ed., 2020.
- [23] E. Rich, K. Knight, and S. Nair, *Artificial Intelligence*. Tata McGraw Hill, third ed., 2009.
- [24] J. D. Pedraza Caro, “La inteligencia artificial en la sociedad: explorando su impacto actual y los desafíos futuros.” No Publicado, Mayo 2023.
- [25] M. Elgendy, *Deep learning for vision systems*. Simon and Schuster, 2020.
- [26] F. Berzal, *Redes neuronales & deep learning*. Granada, 2019.
- [27] J. D. Kelleher, *Deep learning*. MIT press, 2019.
- [28] E. Stevens, L. Antiga, and T. Viehmann, *Deep learning with PyTorch*. Manning Publications, 2020.
- [29] Z. Li, F. Liu, W. Yang, S. Peng, and J. Zhou, “A survey of convolutional neural networks: analysis, applications, and prospects,” *IEEE Transactions on Neural Networks and Learning Systems*, vol. 33, pp. 6999–7019, 2021.
- [30] P. Purwono, A. Ma’arif, W. Rahmiani, H. I. K. Fathurrahman, A. Z. K. Frisky, and Q. M. ul Haq, “Understanding of convolutional neural network (cnn): A review,” *International Journal of Robotics and Control Systems*, vol. 2, no. 4, pp. 739–748, 2022.
- [31] P. A. Torrione, K. D. Morton, R. Sakaguchi, and L. M. Collins, “Histograms of oriented gradients for landmine detection in ground-penetrating radar data,” *IEEE Transactions on Geoscience and Remote Sensing*, vol. 52, no. 3, pp. 1539–1550, 2014.

- [32] H. Erfankhah, M. Yazdi, M. Babaie, and H. R. Tizhoosh, "Heterogeneity-aware local binary patterns for retrieval of histopathology images," *IEEE Access*, vol. 7, pp. 18354–18367, 2019.
- [33] M. Krichen, "Convolutional neural networks: A survey," *Computers*, vol. 12, no. 8, p. 151, 2023.
- [34] N. Bačanin Džakula *et al.*, "Convolutional neural network layers and architectures," in *Sinteza 2019-International Scientific Conference on Information Technology and Data Related Research*, pp. 445–451, Singidunum University, 2019.
- [35] M. M. Taye, "Theoretical understanding of convolutional neural network: concepts, architectures, applications, future directions," *Computation*, vol. 11, no. 3, p. 52, 2023.
- [36] R. Nithya, B. Santhi, R. Manikandan, M. Rahimi, and A. H. Gandomi, "Computer vision system for mango fruit defect detection using deep convolutional neural network," *Foods*, vol. 11, no. 21, 2022.
- [37] Y. Li, J. Xue, K. Wang, M. Zhang, and Z. Li, "Surface defect detection of fresh-cut cauliflowers based on convolutional neural network with transfer learning," *Foods*, vol. 11, no. 18, 2022.
- [38] C. Wang and Z. Xiao, "Potato surface defect detection based on deep transfer learning," *Agriculture*, vol. 11, no. 9, 2021.
- [39] M. Arora, P. Mangipudi, and M. K. Dutta, "Deep learning neural networks for acrylamide identification in potato chips using transfer learning approach," *Journal of Ambient Intelligence and Humanized Computing*, pp. 1–14, 2021.
- [40] A. Nagpal and G. Gabrani, "Python for data analytics, scientific and technical applications," in *2019 Amity International Conference on Artificial Intelligence (AICAI)*, pp. 140–145, 2019.
- [41] M. Lutz, *Learning python: Powerful object-oriented programming*. "O'Reilly Media, Inc.", 2013.
- [42] D. S. Ahmed, R. shaker ibrahim Al-badri, F. A. Hashim, and N. M. Hussien, "Keras deep learning package in python: A review," *European Journal of Interdisciplinary Research and Development*, vol. 19, pp. 24–29, 2023.
- [43] A. Kapoor, A. Gulli, S. Pal, and F. Chollet, *Deep Learning with TensorFlow and Keras: Build and deploy supervised, unsupervised, deep, and reinforcement learning models*. Packt Publishing Ltd, 2022.
- [44] Y. Adhitya, S. W. Prakosa, M. Köppen, and J.-S. Leu, "Feature extraction for cocoa bean digital image classification prediction for smart farming application," *Agronomy*, vol. 10, no. 11, p. 1642, 2020.
- [45] R. Reed, "Seedling image classification using keras." <https://www.kaggle.com/code/reedr1208/seedling-image-classification-using-keras/notebook>, February 2020. Accedido en Enero de 2024.
- [46] U. Navid, "Pepsico lab potato chips quality control." <https://www.kaggle.com/datasets/concaption/pepsico-lab-potato-quality-control>, January 2022. Accedido en Septiembre de 2023.

- [47] O. Ronneberger, P. Fischer, and T. Brox, “U-net: Convolutional networks for biomedical image segmentation,” in *Medical Image Computing and Computer-Assisted Intervention—MICCAI 2015: 18th International Conference, Munich, Germany, October 5-9, 2015, Proceedings, Part III 18*, pp. 234–241, Springer, 2015.
- [48] J. Cantoral, “Dl fundamentals.” https://github.com/JACantoral/DL_fundamentals/blob/main/Fundamentals_DL_UNET_4_video_v2.ipynb, July 2022. Accedido en Enero de 2024.
- [49] K. Mohi-Alden, M. Omid, M. Soltani Firouz, and A. Nasiri, “A machine vision-intelligent modelling based technique for in-line bell pepper sorting,” *Information Processing in Agriculture*, vol. 10, no. 4, pp. 491–503, 2023.
- [50] OpenCV, “OpenCV Documentation,” 2023. Accedido: 4 de enero de 2024.
- [51] PhotoRoom, “Photoroom,” 2020.
- [52] I. Kandel and M. Castelli, “The effect of batch size on the generalizability of the convolutional neural networks on a histopathology dataset,” *ICT Express*, vol. 6, no. 4, pp. 312–315, 2020.
- [53] X. Huang, R. Azzam, S. Javed, D. Gan, L. Seneviratne, A. Abusafieh, and Y. Zweiri, “Cm-unet: Convmixer unet for segmentation of unknown objects in cluttered scenes,” *IEEE Access*, vol. 10, pp. 123622–123633, 2022.
- [54] O. V. Putra, N. Trisnaningrum, N. S. Puspitasari, A. T. Wibowo, and E. Rachmawaty, “An optimized rice leaf disease classification using transfer learning and balanced class weight distribution based on bandit approach,” in *2022 5th International Conference on Information and Communications Technology (ICOIACT)*, pp. 417–422, 2022.

Anexos

Todos los códigos fueron desarrollados en dos entornos IDE: Jupyter notebook y Colab; se encuentran disponibles en un repositorio de GitHub: <https://github.com/Sniper202/Deteccion-de-defectos-en-chips-de-papa.git>