Universidad Autónoma de Baja California

Facultad de Ingeniería, Arquitectura y Diseño

Maestría y Doctorado en Ciencias e Ingeniería

Análisis de color y clasificación de uvas a través de algoritmos de aprendizaje de máquina

TESIS

que para cubrir parcialmente los requisitos necesarios para obtener el grado de

DOCTOR EN CIENCIAS

Presenta

Miguel Ricardo González Márquez

Ensenada, Baja California, Enero 2023.

Universidad Autónoma de Baja California

Facultad de Ingeniería, Arquitectura y Diseño

Análisis de color y clasificación de uvas a través de algoritmos de aprendizaje de máquina

TESIS

que para cubrir parcialmente los requisitos necesarios para obtener el grado de

DOCTOR EN CIENCIAS

Presenta

Miguel Ricardo González Márquez

Aprobada por:

Dr. Miguel Enrique Martínez Rosas
Director de Tesis

Dr. Carlos Alberto Brizuela Rodríguez
Co-director de Tesis

Dr. Alejandro Cabello Pasini
Miembro del Comité

Dr. José Ángel González Fraga
Miembro del Comité

Dr. Humberto Cervantes de Ávila
Miembro del Comité

Ensenada, Baja California, Enero 2023.

Resumen de la tesis de **Miguel Ricardo González Márquez**, presentada como requisito parcial para la obtención del grado de DOCTOR EN CIENCIAS del programa de Maestría y Doctorado en Ciencias e Ingeniería (MYDCI) de la UABC. Ensenada Baja California, México, Enero 2023.

Análisis de color y clasificación de uvas a través de algoritmos de aprendizaje de máquina

La detección de frutos individuales en imágenes de racimos de uvas ha recibido un gran impulso gracias al poder computacional actual, a la gran cantidad de datos disponibles y al auge del campo de inteligencia artificial. Para este caso particular, los frutos se aglomeran dentro del racimo, por lo que su detección es un reto importante desde una perspectiva computacional. El cálculo del área que ocupa cada fruto dentro de una imagen, puede ser usado en problemas específicos de viticultura, como por ejemplo, la estimación del rendimiento. A pesar de que múltiples trabajos han abordado el problema de separar frutos individuales con el fin de estimar la cantidad de uvas contenidos en las imágenes, la estimación del área correspondiente para cada fruto y la relación entre el peso con el número de frutos y/o su tamaño, siguen siendo problemas sin resolver. Para abordar dichos problemas, en este trabajo se exploraron tres enfoques de procesamiento de imágenes: el uso de características de color para distinguir entre el fondo y regiones de interés; la detección de círculos para relacionar la región obtenida a un fruto; y aprendizaje profundo para distinguir de manera más fina regiones correspondientes a frutos. El enfoque de aprendizaje profundo presentó el desempeño más alto para la clasificación de frutos, alcanzando un 92 % de exactitud para localizar y determinar el área que corresponde a cada uva dentro de una imagen evaluada; se logró un 95 % de eficiencia de conteo de frutos con respecto al conteo manual y se estimó el peso del racimo con un error de 5.65 % con respecto al peso real. Además, se desarrollaron dos aplicaciones web que usan como motor de procesamiento los algoritmos desarrollados para procesar imágenes de racimos de uvas.

Palabras Clave: *segmentación, aprendizaje de máquina, aprendizaje profundo, racimos de uvas, características de color, detección de círculos.*

Resumen Aprobado por:

Dr. Miguel Enrique Martínez Rosas
Director de Tesis

Dr. Carlos Alberto Brizuela Rodríguez
Co-director de Tesis

Dedicatoria

A Dios por guiarme en este camino y haber llenado de bendiciones mi vida.

A mi esposa **Daniela**, por ser el pilar de apoyo más grande en esta aventura, sin ella nada de esto hubiera sido posible. Por sus palabras de aliento cuando más lo necesitaba y por compartir conmigo este logro. A mis niñas **Dayla**, **Michelle** y **Melina** por llenar de alegría mi vida y por ser tan comprensivas cuando papá tenía que dedicar más tiempo a su trabajo que a ellas.

A mis padres **Rogelia** y **Ricardo**, por sus acertados consejos en los momentos más retadores, y por ser un ejemplo de vida para mi.

A mi hermana **Margarita** por siempre encontrar las palabras para levantarme el ánimo.

Los amo con todo mi corazón.

Agradecimientos

Quiero expresar mi agradecimiento:

A mis directores de tesis, los Doctores **Miguel Martínez** y **Carlos Brizuela**, por confiar en mí para llevar a cabo este proyecto bajo su guía; por sus grandes consejos, enseñanzas y paciencia, pero sobre todo por su disposición de querer compartir su conocimiento conmigo.

A mis sinodales los Doctores Alejandro Cabello, José A. González Fraga y Humberto Cervantes por sus consejos, observaciones y apoyo durante el desarrollo de esta trabajo.

A mis compañeros y amigos, por su amistad y los buenos momentos que hicieron más amena esta aventura.

Al Consejo Nacional de Ciencia y Tecnología (CONACyT) por el apoyo económico brindado y a la Facultad de Ingeniería, Arquitectura y Diseño de la Universidad Autónoma de Baja California (UABC) por las facilidades otorgadas para la realización de éste trabajo.

Índice general

Índice de figuras

Índice de tablas

Capítulo 1

Introducción

Desde principios del siglo XXI, se han incrementado la cantidad de métodos que abordan tareas computacionales en el área de aprendizaje de máquina (*machine learning*, ML) [1]. Esto debido principalmente al poder computacional de la época y al acceso a grandes volúmenes de datos, lo cual ha dotado de la capacidad para dar soluciones más robustas a problemas más complejos. Con el auge de las redes profundas entrenadas con millones de imágenes [2], el paradigma de ML sufrió un cambio radical sin precedentes. La aplicación de este tipo de soluciones ha presentado un incremento en la última década, y se pueden destacar algunos sectores: la industria [3], la educación [4], la medicina [5], la astronomía [6] y la agricultura[7]. Siendo el último sector de principal interés para esta investigación.

La agricultura moderna se adapta a la actualidad y precisa de técnicas más complejas para optimizar los recursos empleados en los cultivos, a fin de mejorar su calidad y rendimiento [8], y en esta adaptación la viticultura no es la excepción. Usando algoritmos de visión por computadora y aprendizaje de máquina la viticultura es llevada a otro nivel de precisión.

En la actualidad, es evidente los esfuerzos para incluir nuevas tecnologías en viticultura, especialmente si son soluciones de visión por computadora [9], ya que son comúnmente no-invasivas. Esta área presenta un desarrollo creciente y hay tres campos de la viticultura donde se hace presente: detección de enfermedades [10], evaluación de la calidad [11] y estimación del rendimiento [12], siendo el último el de mayor relevancia para esta investigación.

Este trabajo trata acerca del uso de técnicas de inteligencia artificial aplicadas en datos que provienen de viticultura. Se exploran algunos esquemas de captura de imágenes en campo y se emplean algoritmos de ML para extraer información de imágenes de racimos de uvas. Estos algoritmos, son el motor de procesamiento en las soluciones de visión por computadora, por lo que también serán el núcleo de esta investigación.

1.1. Planteamiento del problema

Este trabajo se centra en la siguiente pregunta de investigación ¿cuál es el esquema de visión por computadora más adecuado que permita detectar y localizar frutos individuales en imágenes de racimos de uvas? Esto con la intención de ser la plataforma para una potencial solución de algunos de los problemas en viticultura como el de definir la variabilidad del tamaño o el color de los frutos dentro de un racimo. El uso de imágenes digitales permite emplear algunos métodos para localizar regiones correspondientes a frutos individuales. Por ejemplo, Millan et al. [13] propusieron un método que estima la cantidad de frutos. Así mismo, Aquino et al. [14, 15] introdujeron un método de detección que calcula el número de frutos en una etapa temprana de crecimiento; mientras que Perez-Zavala et al. [12] propusieron un esquema que localiza centros candidatos y determina si son uvas con un algoritmo de ML. Por su parte, Zabawa et al. [16, 9], usaron un modelo de aprendizaje profundo (*deep learning*, DL) no solo para localizar, sino además para segmentar áreas que corresponden a frutos individuales.

Nuske et al. [17] mencionan que, la variación del rendimiento en un viñedo es una combinación de tres elementos: el número de racimos por planta, la cantidad de frutos por racimo y el tamaño de las uvas. Por lo tanto, reconocer frutos individuales dentro de imágenes digitales, resulta relevante para describir las variaciones en el rendimiento de un cultivo, por esta razón, el problema computacional abordado en esta investigación es: *localizar los frutos individuales dentro de una imagen digital de racimos de uvas.*

1.2. Motivación

Aunque los trabajos mencionados previamente, abordaron de manera novedosa la problemática de esta investigación, la mayoría se enfocaron en localizar racimos o estimar el número de frutos contenidos en la imagen, y no incluyeron una manera de localizar las regiones correspondientes a frutos individuales o su relación con el peso de los racimos. Por ejemplo, Millan et al. [13] resolvieron el conteo de frutos pero no su localización dentro de las imágenes; Aquino et al. [14, 15] localizaron los frutos individuales en una etapa temprana de crecimiento, cuando el nivel de aglomeración en los frutos es menor en comparación al tiempo cercano a la cosecha. Por otro lado, Perez-Zavala et al. [12] propusieron un esquema que localiza y clasifica uvas, pero requiere saber de antemano el tamaño promedio de las uvas a localizar. Por último, Zabawa et al. [16, 9] propusieron un esquema de aprendizaje profundo donde localizaron y estimaron la cantidad de frutos sin depender de ningún parámetro adicional, pero no hicieron pruebas cuantitativas fuera de su conjunto de entrenamiento.

Por lo antes mencionado, este trabajo se centra en dar respuesta a la pregunta de investigación antes formulada y para ello se plantean los siguientes objetivos.

1.3. Objetivo General

Desarrollar una metodología que permita clasificar, contar de manera eficiente, y extraer parámetros para estimar: la cantidad y la talla de las uvas en un racimo, usando técnicas de procesamiento digital de imágenes y aprendizaje automatizado.

De manera puntual, se proponen los siguientes objetivos particulares con el fin de lograr el objetivo general.

1.4. Objetivos específicos

1. *Crear una base de datos de imágenes para evaluar los algoritmos propuestos.*
2. *Proponer un vector de características basado en el color para utilizarlo en el proceso de segmentación de imágenes de racimos de uvas.*
3. *Determinar la técnica más adecuada para el problema de detección de círculos y evaluar su aplicabilidad en imágenes de racimos de uvas.*
4. *Determinar el enfoque de aprendizaje de máquina más adecuado para identificar los frutos de manera individual.*
5. *Proponer una etapa a post-proceso para determinar el área oculta de frutos parcialmente ocluidos.*
6. *Determinar, por variedad, la relación entre el número de frutos individuales localizados en una imagen digital, con peso real del racimo.*

Para abordar cada objetivo se siguió la siguiente metodología.

1.5. Metodología propuesta

Para llevar a cabo esta investigación se proponen las siguientes etapas:

1. Recolección de datos: Se propone capturar imágenes de racimos de uvas, a fin de incrementar el número de muestras de las bases de datos ya existentes que incluyen este tipo de imágenes. Así mismo, revisar la disponibilidad de estas bases de datos públicas, y etiquetar manualmente las imágenes adquiridas para que puedan usarse en la evaluación de los algoritmos propuestos.

2. Espacios de color: Explorar diferentes representaciones de imágenes para extraer características de color, a fin de localizar regiones de interés.

3. **Detección de círculos:** Se propone evaluar el desempeño de estos detectores en imágenes genéricas. Además, explorar su utilización para identificar regiones que correspondan a frutos individuales dentro de imágenes de racimos de uvas.

4. **Algoritmos de ML:** Explorar diversos enfoques de aprendizaje de máquina, tales como, el de clasificar píxel-a-píxel, localizar regiones de interés y el de aprendizaje profundo. El primero tiene como objetivo catalogar píxeles que correspondan a cierta categoría dada; el segundo consiste en evaluar si las regiones detectadas corresponden a alguna clase y el último consiste en categorizar y revisar si hay grupos de píxeles que corresponden a cierta región de interés. Un ejemplo de estas regiones pueden ser frutos individuales.

5. **Predicción de área ocluida:** Proponer una alternativa para estimar el área oculta de los frutos captados en una imagen, ya que de manera natural se traslapan unos con otros.

6. **Predicción de peso:** Explorar la relación tamaño-peso de los frutos ó racimos y proponer una alternativa para estimar el peso de un racimo capturado en una imagen digital.

1.6. Secuencia de la tesis

Este trabajo consta de cinco capítulos y cada uno contiene lo siguiente:

Capítulo 1: Una introducción al trabajo de investigación, se indica la motivación, se formula la pregunta de investigación y se describen los objetivos, tanto el general como los específicos; estos para abordar la problemática de detectar frutos individuales en imágenes de racimos de uvas.

Capítulo 2: Los antecedentes al problema, los trabajos en viticultura abordados por visión por computadora y el enfoque de aprendizaje profundo.

Capítulo 3: La metodología propuesta para segmentar racimos, frutos individuales y estimar el área ocluida de cada fruto detectado en imágenes digitales.

Capítulo 4: Los resultados obtenidos con la metodología propuesta, junto con una discusión de las limitaciones de esta investigación.

Capítulo 5: Las conclusiones, un resumen de las aportaciones y algunas ideas de trabajo a futuro.

Capítulo 2

Marco teórico

Este capítulo presenta los fundamentos de visión por computadora y los antecedentes alrededor del enfoque de aprendizaje de máquina utilizado. Se analizan algunos trabajos que utilizaron visión por computadora en el área de viticultura, proporcionando la información más relevante para esta investigación. Se explican los esquemas de captura propuestos en la literatura y se da un resumen de los conjuntos de datos de imágenes disponibles públicamente. El capítulo finaliza con la explicación de los conceptos básicos del enfoque de aprendizaje profundo ya que este fue primordial para llevar a cabo esta investigación.

2.1. Antecedentes

Un sistema de visión por computadora corresponde a una serie de técnicas y dispositivos que imitan al sistema de visión humano [18]. Regularmente, se compone de dos pilares principales [7]: el esquema de captura de imágenes y el motor de procesamiento. El primero corresponde a un dispositivo que captura una escena 2D de un mundo real en 3D, y la almacena en un archivo digital llamado imagen. Formalmente, sea $\mathbf{A}$ una matriz de imagen formada por unidades básicas llamadas píxeles e indexadas por (i, j) con M renglones y N columnas y cuantificada en k valores, es decir, los valores de $\mathbf{A}(i, j) \in \{0, k-1\}$. El valor de k depende de la sensibilidad del sensor de captura (Figura 2.1).

Por otra parte, el segundo pilar corresponde a un algoritmo que analiza la imagen y la convierte en información relevante para el usuario final; al área que estudia estos procedimientos se le conoce como **procesamiento de imágenes digitales** [20]. Las aplicaciones industriales de visión por computadora son diversas [18], desde inspeccionar circuitos impresos, hasta verificar el nivel correcto de líquido en botellas. Estas soluciones suelen ser prácticas, ya que realizan tareas que normalmente son tediosas para una persona luego de cierto tiempo de desempeñarlas. Sin importar lo simple de una tarea, extraer información relevante de una imagen digital no es tarea sencilla.

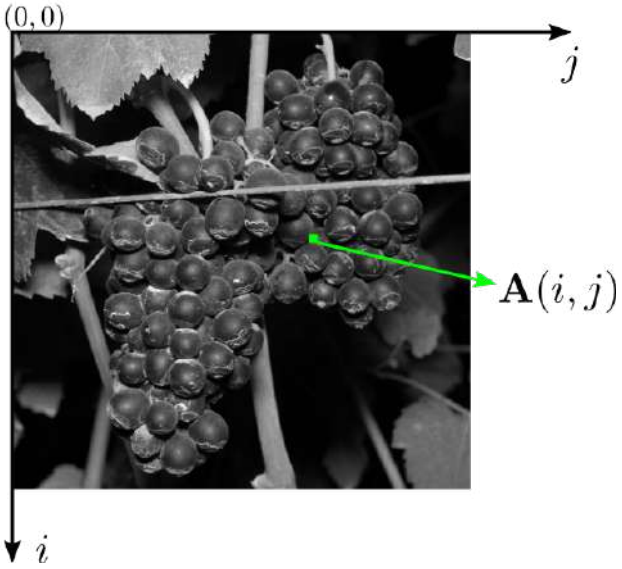


Figura 2.1: Representación de los píxeles dentro de una imagen digital; idea tomada de [19].

Con el avance de la tecnología, en los últimos años han surgido diferentes esquemas de procesamiento y el que más se ha destacado es el de aprendizaje de máquina (ML, *machine learning*).

El aprendizaje de máquina es un área en desarrollo dentro de inteligencia artificial. Este corresponde a aplicar una serie de técnicas, las cuales buscan similitudes y diferencias dentro de registros previamente almacenados (datos), y al mismo tiempo aprender de ellos [21]. El origen del aprendizaje de máquina se remonta a 1957, cuando el psicólogo Frank Rosenblatt y un grupo de personas crearon una máquina que reconocía letras del alfabeto [1]. Esta máquina fue llamada el "perceptrón" ya que su diseño se basó en el comportamiento de una neurona del sistema nervioso humano. Sin embargo, por la carencia del poder computacional de la época, este desarrollo no tuvo impacto hasta décadas más tarde. Los algoritmos que hoy conocemos como redes neuronales artificiales (ANN, *Artifitial Neural Network*) son basados en el "perceptrón" de Rosenblatt.

Fue a partir del año 2000 cuando ML tuvo un parteaguas en su historia, al darse tres condiciones que hicieron posible su aplicación como lo conocemos hoy [1]. Primero, gracias al incremento en la infraestructura de internet, el acceso a **grandes volúmenes de datos** fue posible. Segundo, **la inclusión de unidades gráficas de procesamiento** (GPU, *graphic process units*) a ML, posibilitó un incrementó en el poder computacional para entrenar algoritmos más complejos. Tercero, el desarrollo de lo que hoy conocemos como **aprendizaje profundo** (DL, *deep learning*); este consiste en aprender de los datos crudos e incluir un número mayor de procedimientos en comparación a las redes tradicionales. El enfoque de DL ha probado ser el mejor en ciertas tareas computaciona-

les, como lo es **clasificación** de objetos [2]. Esta tarea consiste en catalogar cierto dato, por ejemplo, texto, audio, videos o imágenes, dentro de una categoría dada, *e.g.* "fruta", "perro", "vehículo", etc. Esta tarea parece trivial para un ser humano, lo complicado radica en resolverla para millones de datos en un tiempo razonable. Las soluciones de ML tienen el objetivo principal de resolver tareas repetitivas y que son propensas a errores si una persona las lleva a cabo de manera repetida. Hay una tarea más compleja que la de clasificación de datos, y es específica para imágenes y videos: **segmentación**. Esta consiste en identificar si los píxeles de la imagen corresponden a alguna clase dada, por ejemplo, "fondo", "bordes", "objetos", etc. El proceso general para desarrollar una solución de ML comprende seis etapas secuenciales [7]:

1. **Preparación/adquisición de los datos:** esta corresponde a adquirir los datos adecuados para dar solución al problema en cuestión.

2. **Selección de las características de interés:** corresponde al uso de una representación más simple, en comparación a los datos crudos adquiridos en la etapa anterior, por ejemplo, textura o color. Cabe destacar que esta etapa no aplica para el enfoque de aprendizaje profundo.

3. **Selección del algoritmo:** en la actualidad existe un abanico de opciones con respecto al algoritmo que se puede utilizar. La elección de este dependerá del problema específico a resolver.

4. **Sintonización de parámetros:** cada algoritmo incluye una cantidad finita de variables, para las cuales se deben calcular los valores que optimicen algún criterio de sintonización pre-especificado.

5. **Entrenamiento:** este es el proceso medular donde se toman los datos y se aprenden de ellos. Cosiste en determinar los pesos de la red que minimizan alguna función de error entre la predicción y el valor real.

6. **Validación:** este proceso consiste en evaluar al algoritmo entrenado, con una serie de datos no antes vistos y calcular el porcentaje de error del mismo.

Las soluciones basadas en sistemas de visión por computadora que incluyen inteligencia artificial, han tenido una tendencia ascendente en la última década [7]. Hay diversas áreas en desarrollo donde este tipo de soluciones se ven presentes y algunas pueden ser: medicina [22], astronomía [23] y agricultura [9], siendo la última de gran interés para esta investigación. Las soluciones de visión por computadora suelen ser atractivas en el área de viticultura[1], ya que son una herramienta adicional en la toma de decisiones para el desarrollo de un viñedo [8]. Estas soluciones permiten extraer

[1]Sub-área de la agricultura que estudia el cultivo de la vid.

información más fina de las escenas capturadas, dar respuesta rápida a algún problema específico, y son generalmente soluciones no-invasivas. Por un lado, si un viticultor necesita saber si alguna enfermedad persiste en el viñedo, la solución propuesta por Adeel et al. [24] pudiera ser una alternativa. Su solución consistió en un algoritmo que cataloga imágenes de hojas de vid en tres enfermedades establecidas, o define si es una hoja sana. Por otro lado, si se necesita saber dónde se localizan racimos dentro de una imagen, la solución que propusieron Liu et al. [25] también puede ser una alternativa; ellos propusieron usar características de color para reconocer racimos de uvas tintas. Hay diversas problemáticas en viticultura, que pueden ser abordadas con un enfoque de visión por computadora y en lo siguiente se explican algunas de ellas.

2.2. Visión por computadora en viticultura

En esta sección se da un recorrido por los trabajos que emplearon visión por computadora en viticultura. Se describe de manera general cada uno, y se resaltan aspectos relevantes para esta investigación. Los tres problemas de viticultura donde la visión por computadora se hace presente son: detección de enfermedades, evaluación de la calidad y la predicción del rendimiento.

2.2.1. Detección de enfermedades

Este problema consiste en detectar anomalías dentro de imágenes que incluyen frutos u hojas [26, 27, 28, 29, 30, 31, 32], y es uno de los problemas comunes en agricultura. Durante esta investigación se encontraron algunas alternativas que intentan dar solución a esta problemática.

Primero, Dutta et al. [33] determinaron si un racimo de uvas presenta residuos de pesticidas. Utilizaron un enfoque de ML basado en las máquina de vectores de soporte (SVM, *support vector machine*, Apéndice A.2.2) para distinguir entre dos categorías: uvas tratadas y sin tratamiento. Usaron un dispositivo óptico (espectrómetro de masas) para medir si los racimos contenían residuos de pesticidas y con esta información armaron sus datos para el proceso de aprendizaje. Para este usaron 18 variables estadísticas extraídas de las imágenes en escala de grises, *i.e.* media, varianza, desviación estándar, entre otras, por lo que no incluyeron características de color.

Después, Adeel et al. [24] clasificaron enfermedades en imágenes de hojas de vid. Ellos usaron el conjunto llamado PlantVilage[2] para entrenar una SVM e identificar las siguientes enfermedades: *black rot*, *black measle*, *leaf blight* y las hojas sanas. Escogieron como características textura, geometría y color de las imágenes. Para la última característica usaron la representación del espacio CIELab (Apéndice A.1.2).

[2]https://www.kaggle.com/datasets/emmarex/plantdisease, accesado el 15 de julio de 2022

Por último, Oberti et al. [10] propusieron identificar zonas afectadas en hojas de vid dentro de una imagen, para luego, rociar pesticida en la zona localizada. Ellos usaron un brazo robótico que incluía una cámara multiespectral, la cual capturaba imágenes en el infrarrojo cercano y el espectro visible; y localizaron zonas dañadas con un esquema tradicional de procesamiento de imágenes. El brazo incluye un rociador que colocó pesticida en las zonas dañadas, y reportan que su mayor aportación es la reducción de aplicación de pesticidas hasta en 90 %.

Detectar enfermedades en cultivos de manera temprana es crucial, ya que impacta en la calidad de los frutos. La evaluación de esta, es otra área donde se puede implementar una solución de visión por computadora, y a continuación se describen algunos de los trabajos encontrados.

2.2.2. Evaluación de la calidad

Este problema consiste en determinar cuál es el estado de los frutos en un cultivo, es decir, analizar sus propiedades físicas y químicas a fin de asignarle un valor o categoría. Las propiedades químicas pueden ser la concentración de azúcar y la acidez [34]; mientras que las físicas pueden ser el color o el tamaño. Esta investigación solo se centró en las propiedades físicas y visibles de las uvas. La realidad es que no hay a la fecha, un enfoque de visión por computadora que resuelva del todo esta problemática. Esto se debe a que, para determinar la calidad en los frutos es necesario hacer un análisis químico de su contenido, y estas características no son captadas por un sensor de una cámara digital. Sin embargo, hay algunas propuestas que intentaron hacerlo y se pueden destacar dos.

Primero, Nogales-Bueno et al. [11] capturaron uvas tintas individuales colocadas sobre una superficie blanca. Ellos usaron una cámara especializada que captura toda la riqueza del color proporcionado por una fuente de iluminación; cada captura corresponde a n imágenes, y cada imagen incluye la información al separar la luz en sus n componentes esenciales; formando un hipercubo de cada escena. Este tipo de cámaras llevan por nombre hiperespectrales[3]. Para reducir la dimensión del hipercubo y tener una representación más simple, usaron un análisis por componentes principales (PCA, ver Apéndice A.1.5), y al lado de cuatro variables químicas intentaron predecir la concentración de azúcar y la acidez de los frutos evaluados. Sin embargo, en su trabajo no presentaron resultados concluyentes y mencionan que solo fue un reporte preliminar.

Luego, Rodríguez-Pulido et al. [35] intentaron determinar el estado de madurez de uvas individuales. Capturaron imágenes a color de uvas y de sus semillas. Su trabajo lo centraron en relacionar características de color (espacio CIELab, Apéndice A.1.2) con un estado de madurez previamente definido. Su conclusión fue que las semillas se tornaban de un tono café oscuro conforme la maduración avanzaba.

[3]Estos dispositivos capturan una imagen por cada longitud de onda en el espectro electromagnético, generan un hipercubo por escena y su resolución la define el intervalo de longitudes de onda captadas.

A continuación se describen los trabajos que propusieron una solución a la estimación del rendimiento en viticultura.

2.2.3. Estimación del rendimiento

Como se mencionó en la sección 1.1, la variación del rendimiento es una combinación de tres factores: el número de frutos por racimo, la cantidad de racimos por planta y el tamaño de cada fruto. Aunque hay varias propuestas que abordaron esta problemática, la mayoría se centra en localizar racimos o estimar el número de frutos contenidos en la imagen, y solo dos trabajos incluyen una relación con el peso del cultivo.

Por un lado, Nuske et al. [17] propusieron un esquema para predecir el rendimiento en imágenes de campo sobre tres variedades de uva: Gerwurztraminer, Traminette, Riesling. Ellos relacionaron linealmente el número de frutos con el peso de todo un surco. Además estimaron el número de racimos capturados en una imagen, sin embargo, no hicieron una evaluación cuantitativa sobre esta última estimación. Para contabilizar los frutos tomaron ventaja de su forma esférica, y encontrando las zonas con mayor reflexión de luz dentro de la imagen, obtuvieron un error de 9.8 % promedio en ocho surcos evaluados.

Por otro lado, Millan et al. [13] usaron un esquema basado en características de color para identificar las áreas correspondientes a racimos; y para estimar el número de frutos, contaron cuántas veces cabía un área circular promedio en el área correspondiente a racimo. Obtuvieron una $R^2 = 0.81$ con un RMSE de 310.2 gramos, al relacionar linealmente el número de frutos con el peso. El valor de R^2 se interpreta como un porcentaje de desempeño para el conteo, sin embargo, esto realmente indica qué tan bien se ajusta una recta al conteo estimado.

La tarea principal de ambos trabajos, fue estimar automáticamente la cantidad de frutos contenidos en una imagen digital. Luego relacionarlo linealmente con el peso real de cada racimo y obtener un coeficiente de relación. Ya que localizar frutos individuales es una de las tareas para determinar la variación del rendimiento [17], ahora se describen los trabajos centrados en esta sub-problemática.

Detección de frutos individuales

Detectar frutos individuales no es una tarea sencilla, la forma natural de los frutos en cada racimo la hace una tarea retadora en visión por computadora. Los frutos se aglomeran de forma natural, y los bordes que separan cada fruto se difuminan al capturarlos en una imagen digital. En esta investigación se encontraron algunas propuestas y a continuación se describen los aspectos relevantes de cada una.

Primero, un método para localizar frutos individuales es la transformada de simetría radial (RST, por su sigla en inglés *radial symmetry transform*). Esta consiste en aprovechar la forma esférica de los frutos y localizar algo que es llamado “pico de reflexión”

(Figura 2.2). Esta región de interés, es la zona dentro de una uva que aparece más iluminada en una imagen, al usar fuente de iluminación frontal al momento de capturarla, y es llamada en la literatura como puntos clave (*key points*) [12, 14, 17].

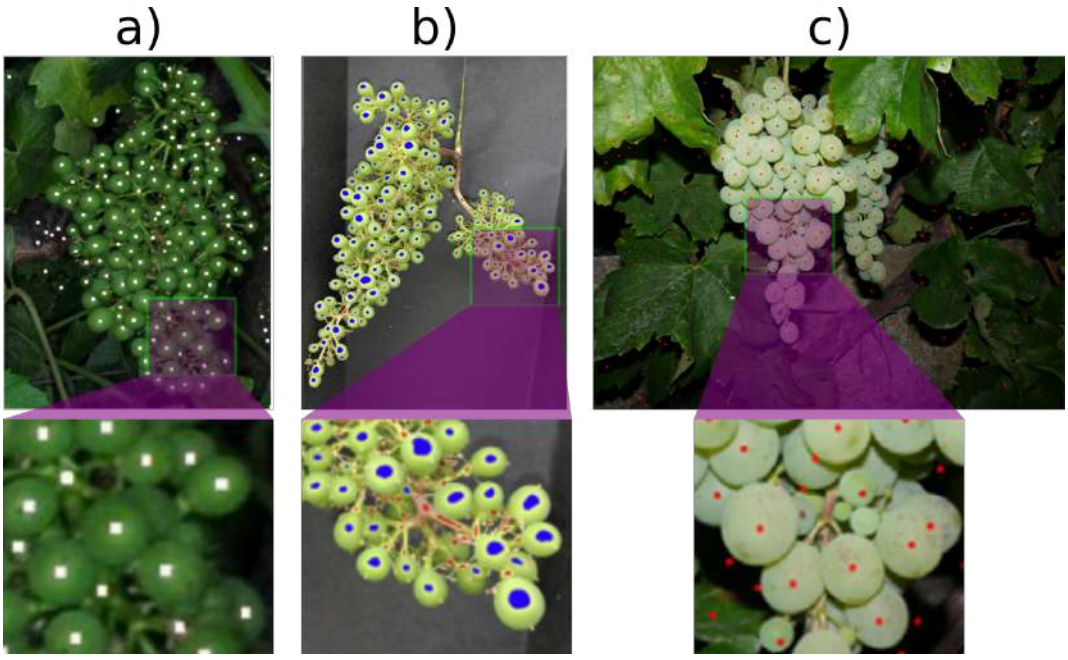


Figura 2.2: Transformada de simetría radial. Muestra sobre imágenes de racimos de uvas: (a) Nuske et al. [17], (b) Aquitno et. al. [14] y (c) Pérez-Zavala et. al. [12].

Nuske et al. [17] determinaron si los puntos detectados correspondían a uvas, a través de revisar si en la vecindad había otros puntos clave cercanos. Tanto Aquino et. al. [14] como Perez-Zavala [12] usaron un esquema de ML para determinar si estos puntos clave correspondían a uvas. Después, Millan et al. [13] propusieron un método que, en vez de localizar uvas, ellos estimaron la cantidad de frutos dentro de un racimo, basados en el área segmentada de la imagen y la geometría de las uvas (Figura 2.3-a). Por último, Murillo et al. [36] (Figura 2.3-b) y Rudolph [37] (Figura 2.3-c) utilizaron la transformada de Hough (CHT) para detectar uvas como círculos y estimar la cantidad de frutos contenidos en la imagen.

Segundo, las propuestas que usaron un enfoque de ML requirieron seleccionar características de interés. Por un lado, Škrabánek et al. [38] y Perez-Zavala et al. [12] categorizaron sub-imágenes con una SVM a fin de determinar si era una uva o no. El primero usó características de textura usando gradientes orientados (HOG, detalles en Apéndice A.1.4), mientras que el segundo incluyó además patrones binarios (LBP, ver Apéndice A.1.4) para enriquecer las características elegidas y por consecuencia, mejorar el desempeño de clasificación. Por otro lado, Aquino et al. [14, 15] usaron en ambas propuestas una red neuronal artificial (ANN, Apéndice A.2.2), incluyendo característi-

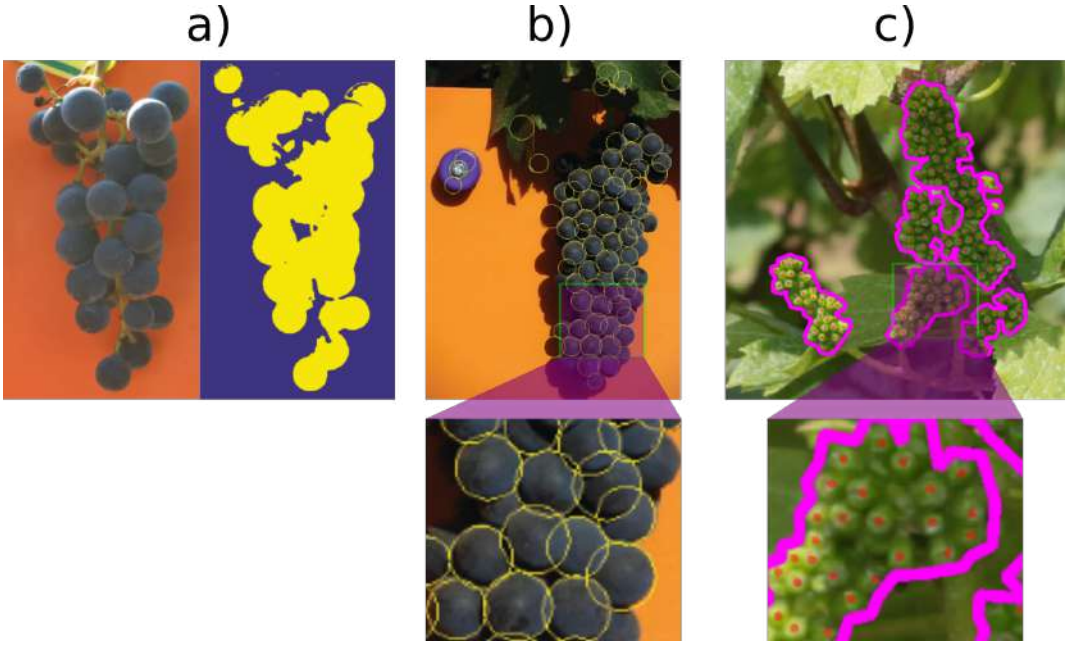


Figura 2.3: Estimando el número de frutos. Muestra sobre imágenes de racimos de uvas: (a) Millan et al. [13], (b) Murillo et al. [36] y Rudolph [37].

cas de color y estimaron el número de frutos en un racimo en una etapa temprana de crecimiento.

Tercero, las propuestas que usaron DL aprendieron las características de las imágenes durante el entrenamiento. Por un lado, Škrabánek et al. [39] abordaron la tarea de clasificación usando una red neuronal convolucional (CNN, *convolutional neural network*) y evaluaron las mismas imágenes que en la propuesta original [38]. Por otro lado, Rudolph et al. [37] y Zabawa et al. [9] utilizaron también una red convolucional, pero abordando la tarea de segmentación. Rudolph et al. [37] segmentaron zonas que corresponden a racimos y usando un detector de círculos estimaron la cantidad de frutos y su posición, mientras que Zabawa et al. [9] segmentaron cada fruto individual en un solo paso.

La selección de características resulta de gran interés, ya que los racimos comparten aspectos similares, tales como, la forma y el color. Por tal motivo ahora se explica cómo usar diversas representaciones de color para abordar la problemática en esta investigación.

2.2.4. El color en imágenes de viticultura

El color es una característica a resaltar en esta investigación, ya que este se comparte entre frutos de la misma variedad, y además puede extraerse de imágenes digitales.

Un algoritmo computacional requiere representaciones matemáticas para procesar las características de color, estos son los llamados índices o espacios de color.

Por un lado, los índices de color son usados para separar el suelo del follaje en imágenes de agricultura [40]. Hamuda et al. [40] explicaron cómo extraer 10 índices para realizar esta tarea. A la fecha no se encontró el uso de estos índices para abordar la tarea de localizar racimos en imágenes. Estos índices tienen como objetivo, procesar una imagen a color y transformarla a un espacio donde sea más sencillo separarla en regiones de interés. Este procedimiento se hace con técnicas tradicionales de segmentación usando un valor de umbral (Apéndice A.2.1), esto corresponde a crear una imagen donde solo se conservan los píxeles a partir de un valor establecido. Los detalles del cálculo de cada índice se pueden encontrar en el Apéndice A.1.3.

Por otro lado, un espacio de color corresponde a una representación 3D de una imagen digital. Aquí se separa el color en dos componentes: cromaticidad e intensidad (Apéndice A.1.2). El primero describe el color puro de cada píxel, por ejemplo, azul, rojo o verde; mientras que el segundo describe la intensidad del mismo, por ejemplo, azul-claro, azul-oscuro, etc. En los trabajos previamente mencionados, los que se usaron de manera recurrente fueron los espacios CIELab y HSV.

Hasta este punto se revisaron los temas alrededor de visión por computadora y aprendizaje de máquina aplicados en la viticultura. Se puede resaltar que la calidad de las imágenes se relacionan con el desempeño de este tipo de soluciones. Las imágenes usadas deben tener una resolución de al menos 2592×2048 píxeles [9] y que sean capturadas bajo condiciones de iluminación controladas [43]. En lo siguiente se explica algunas plataformas encontradas en la literatura para capturar imágenes en un viñedo.

2.3. Esquemas de captura

La selección del esquema de captura es el arranque de cualquier proyecto de visión por computadora –si no se cuenta con imágenes–. Sin embargo, antes de iniciar esta etapa se necesita considerar tres criterios básicos:

- **La resolución:** este criterio lo define el dispositivo de captura, depende del tamaño del sensor y corresponde a la cantidad de píxeles en la imagen.
- **La iluminación:** se refiere a elegir el tipo de fuente de iluminación, ya sea natural o artificial. Capturar una imagen con una fuente de iluminación controlada, ha probado ser la mejor alternativa en la tarea de localizar regiones de interés en imágenes de racimos de uvas [43].
- **La configuración de captura:** este punto lo definen los parámetros de entrada del dispositivo de captura, la posición del mismo y las características de la fuente de iluminación a utilizar.

La literatura provee algunos esquemas para capturar imágenes de racimos de uvas. Primero, Millan et al. [13] utilizaron una cuatrimoto como vehículo para desplazarse entre los surcos del cultivo. Emplearon una cámara Sony alpha 7-II *mirrorless*[4] para adquirir imágenes a color con una resolución de 6000×3376 (24 megapíxeles). Realizaron capturas nocturnas y usaron un panel LED para controlar la iluminación; todo fue fijado a una estructura de aluminio y a su vez al vehículo. Las imágenes fueron capturadas cada cierta revolución de las ruedas mientras este se desplazaba (Figura 2.4a).

Figura 2.4: Esquema de captura de Millan et al.[13]. (a) imagen del montaje real, (b) imagen muestra.

Segundo, Kicherer et al. [43] adaptaron la plataforma PHENObot [44] para capturar imágenes en campo. Ellos diseñaron un vehículo que soporta tres cámaras mono-tono, una cámara a color y una infrarroja con resoluciones a 2448×2050, 2448×2050 y 1388×1038, respectivamente. Incluyeron ocho paneles LED y realizaron las capturas también de noche. Además, todo fue colocado en una estructura de aluminio sobre un vehículo motorizado que navega entre los surcos; con la diferencia que añadieron un dispositivo GPS (*global position system*, por su sigla en inglés) que permitió incluir en los meta-datos de cada imagen su posición geográfica (Figura 2.5a).

Por último, también Kicherer et al. [46] propusieron una versión actualizada que llamaron: **PHENOLiner**. En ella incluyeron cuatro cámaras RGB, una infrarroja y dos hiperespectrales con resolución de 5.1 megapíxeles; usaron 6 lámparas de halógeno de 300 W para iluminación y esta plataforma fue colocada debajo de un tractor que navega sobre los cultivos (Figura 2.6a).

Kicherer et al. [43] hicieron una comparación de las tomas diurnas contra las nocturnas. Ellos concluyeron que para abordar la tarea de segmentación, es mejor realizarlo con capturas de imágenes con iluminación controlada.

Una vez que se tienen los datos recolectados, es necesario hacer una depuración y

[4]No utilizan espejo como las cámaras DSLR: *digital single lens reflex* por su sigla en inglés.

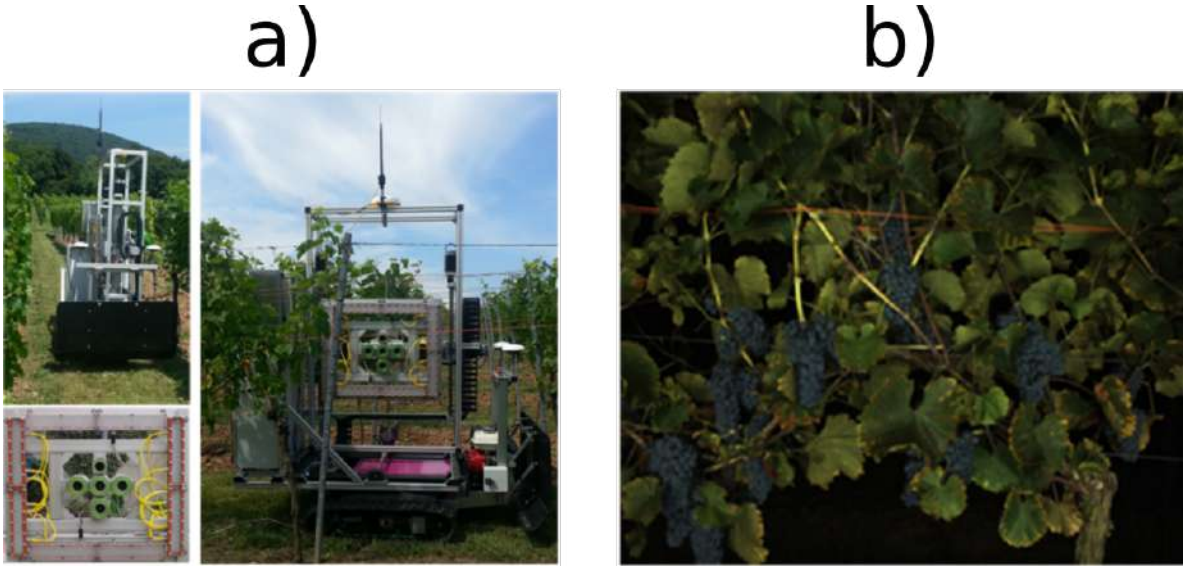


Figura 2.5: La plataforma PHENObot [44]. Implementación de Kicherer et al.[43]: (a) imagen del montaje real, (b) imagen muestra tomada de BIVcolor [45].

Figura 2.6: La plataforma PHENOliner. Propuesta de Kicherer et al.[46]: (a) imagen del montaje real, (b) imagen muestra tomada de [47].

agregar información adicional a cada imagen. En lo siguiente, se explican las características de los conjuntos de datos disponibles, en el dominio de imágenes de racimos de uva.

2.4. La relevancia de los datos

La calidad de los datos de entrada, afecta el desempeño en algoritmos de ML. En este contexto, la calidad se refiere a cumplir ciertas características [48]:

- **Variabilidad**: los datos deben tener suficiente diversidad para representar todo el problema. Para esta investigación, las imágenes deberán contener racimos de diferentes variedades, tamaños y colores.
- **Escala**: la cantidad de muestras debe ser grande, la cuestión es ¿cuánto es grande?. En ocasiones, este dato es incluido dentro de los parámetros a sintonizar, sin embargo, Andrew Ng[5] menciona que a partir de 2000 muestras un conjunto de datos tiene un tamaño aceptable[6]. Tal como lo hicieron Zabawa et al. [9] y Rudolph et al. [37] al incluir más de 5,000 imágenes para entrenamiento.
- **Precisión**: las etiquetas de cada muestra no deben contener errores. La forma de la etiqueta –también llamada referencia u objetivo– dependerá de la tarea (Figura 2.7) y cómo se modeló la solución del problema; esta podrá ser una imagen, una cadena de caracteres o un conjunto de coordenadas.
- **Adecuado**: las muestras deben ser relevantes al problema. Si se agregan imágenes con pocas o ausencia de uvas u objetos fuera del contexto de viñedos, pudiera impactar negativamente el desempeño de los algoritmos de ML.
- **Representativo**: los datos deben representar el problema de manera justa y sin sesgo, y de tenerlo deberá incluir una manera de manejarlo. Este punto es uno de los más complicados de lograr, ya que de manera implícita se añade sesgo al conjunto de imágenes para abordar el problema específico.

Un reto de la actualidad –a pesar del incremento de contenidos multimedia en línea y su acceso a ellos– es la falta de conjunto de datos para problemas específicos. En la literatura se encontraron disponibles algunos conjuntos de imágenes que pueden adecuarse al problema abordado en esta investigación. Estos trabajos se listan a continuación:

[5]https://twitter.com/AndrewYNg, accesado el 06/Junio/2022

[6]https://www.coursera.org/lecture/deep-neural-network/understanding-mini-batch-gradient-descent-lBXu8, , accesado el 06/Junio/2022 - Understanding Mini-batch Gradient Descent

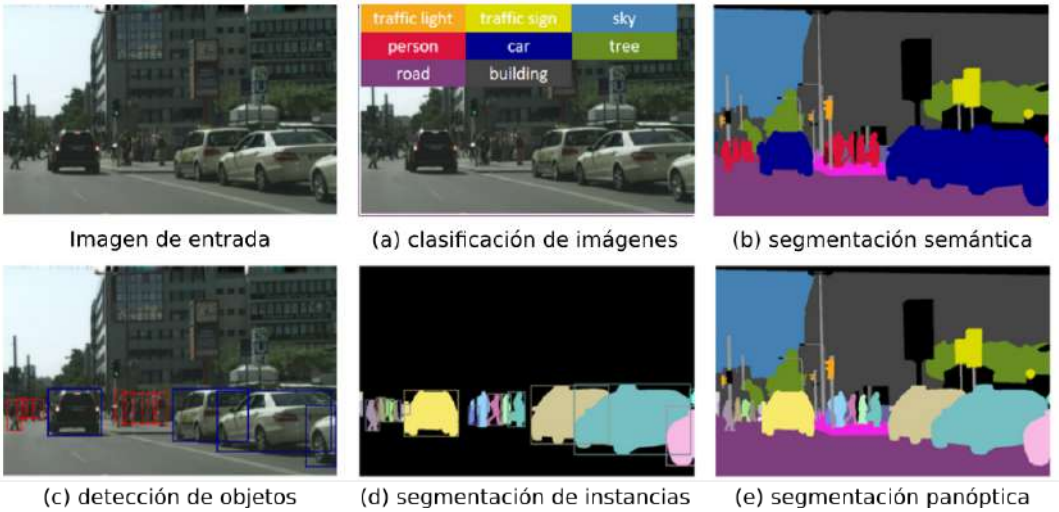


Figura 2.7: Tareas de visión por computadora [49]. La tarea (a) requiere por imagen, solo una lista de palabras (clases) que incluyan los contenidos de la imagen; (b) requiere una imagen donde cada píxel tenga asignado una clase; (c) requiere las coordenadas por cada objeto contenido en una caja (rectángulo) y su clase correspondiente; (d) es una combinación de (b) y (c), y (e) es combinación de todas las tareas y se diferencia a nivel de píxel por objeto individual.

- Škrabánek [38]: contiene 2000 sub-imágenes para entrenamiento y 4000 para validación. Son imágenes cuadradas que contienen uvas individuales (Figura 2.8). Este conjunto de datos no contiene información adicional y la tarea abordada es una clasificación binaria: “uva” o “fondo”.

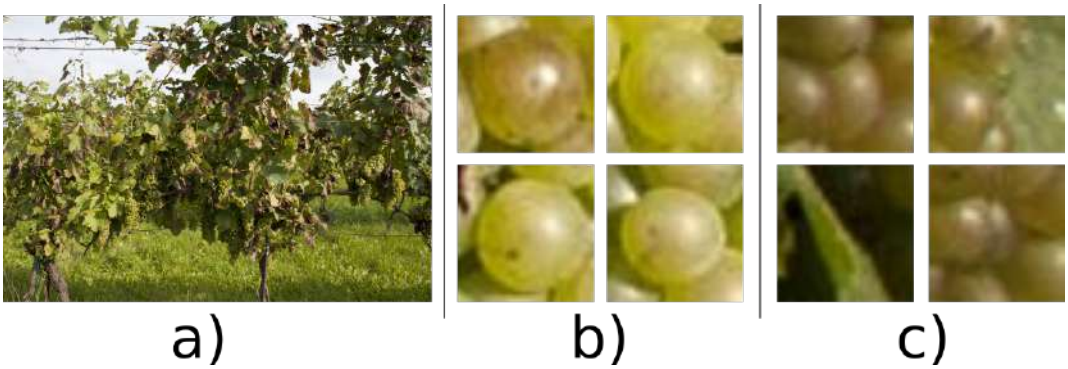


Figura 2.8: Muestra base de datos [38]. a) Imagen fuente, b) imágenes para entrenamiento, c) imágenes para pruebas.

- Seng et al. [50]: este conjunto de imágenes es llamado GrapeCS-ML y contiene 2078 imágenes de racimos individuales en diferentes etapas de crecimiento; la

información adicional a cada imagen es una imagen de referencia, donde cada racimo es separado del fondo y dentro de cada captura, incluyen un patrón estándar de color. La tarea abordada para este conjunto es segmentación semántica (Figura 2.9). El obstáculo con este conjunto de datos es que, a pesar de que se dice público, no se encontró el repositorio y al enviar correo a los investigadores no se recibió respuesta.

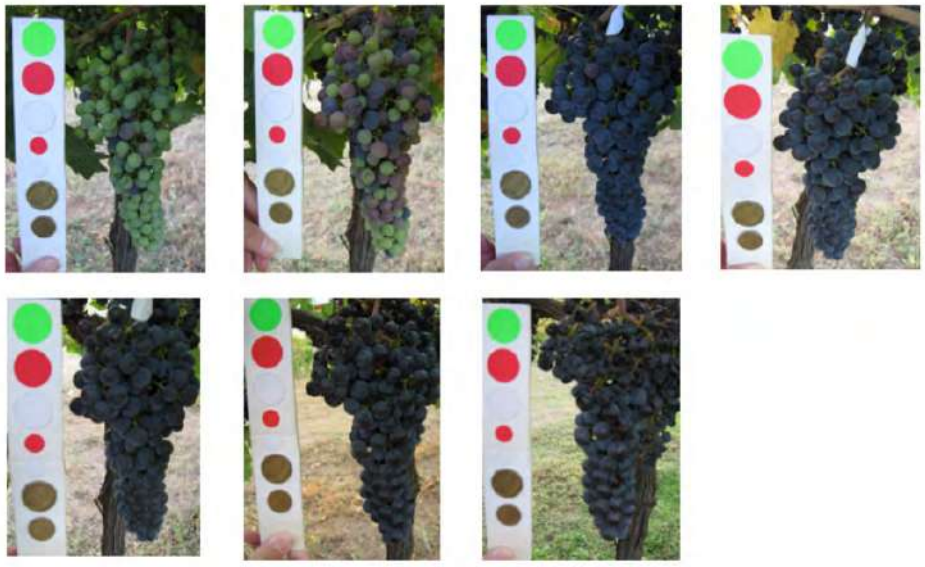


Figura 2.9: Muestra de GrapeCS-ML. Desarrollado por K. Seng et al. [50].

- BIVcolor [45]: contiene 100 imágenes con racimos de distintos colores y etapas de crecimiento (Figura 2.10); tienen identificados los resultados de usar la trasformada Hough para círculos (CHT, por su sigla en inglés *circular Hough transform*) sin embargo, cada uva no está etiquetada de manera individual. Estás imágenes fueron capturadas con la plataforma PHENObot [43].

Figura 2.10: Muestras de BIVcolor [45].

- Kicherer [51]: este conjunto de datos es llamado "*Riesling grapevine canopy images for 3d reconstruction*"; contiene 184 imágenes en secuencia del recorrido de PHENObot [43] y no se encuentran etiquetadas, ya que la tarea para la que fueron capturadas, fue reconstrucción 3D (Figura 2.11).

Figura 2.11: Muestras de Kicherer [51].

- Zabawa [47]: este conjunto de datos contiene 42 imágenes capturadas por la plataforma PHENOliner [46]. Cada muestra incluye una imagen de referencia dónde se identifica los píxeles que corresponden a las clases: "uvas", "bordes" de las uvas y el resto como "fondo" (Figura 2.12). Este conjunto es llamado "*Segmentation of wine berries*", y es el único hasta esta investigación, que incluye imágenes de referencia para abordar la tarea de segmentación.

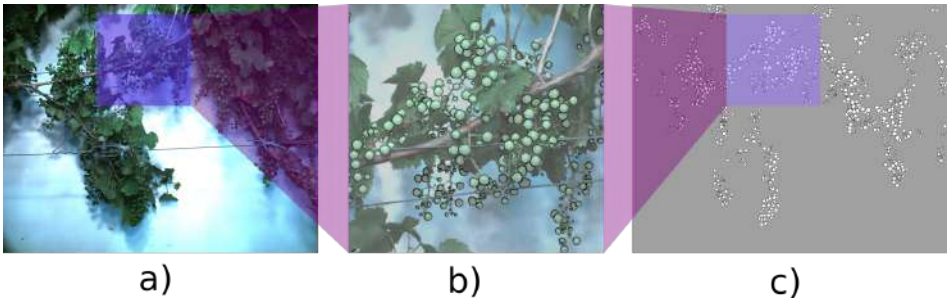


Figura 2.12: Muestras de Zabawa [47]. Imagen fuente (a), porción de la imagen fuente con la etiqueta traslapada (b) y la etiqueta original (c).

El conjunto de Zabawa [47] es el único que incluye imágenes de referencia adecuadas para la tarea de segmentar frutos individuales. Sin embargo, no cumple con las características de representatividad, al solo incluir tres variedades de uvas: Riesling, Felicia y Regent (Tabla 2.1). Ningún de los conjunto explorados en este trabajo, reúne las cinco características de calidad (sección 2.4). En esta investigación se propuso incluir la característica de disponibilidad (Tabla 2.1), ya que el conjunto de datos GrapesCS-ML [50] no está públicamente disponible. Este es el conjunto de datos más completo con respecto a la característica de representatividad, ya que contiene 13 variedades de uvas:

Merlot, Cabernet Sauvignon, Saint Macaire, Flame Seedless, Viognier, Ruby Seedless, Riesling, Muscat Hamburg, Purple Cornichon, Pinot Noir, Sultana, Sauvignon Blanc y Chardonnay. Sin embargo, tener acceso a más variedades depende principalmente de la región. Este conjunto cumplió con la mayoría de las características, a excepción de que las imágenes de referencia no son adecuadas a la tarea de segmentar los frutos individuales.

Tabla 2.1: Sumario de los conjuntos de datos encontrados a la fecha: imágenes de racimos de uvas. Incluye una lista de cotejo de las características que debe incluir un conjunto de datos según Berg et al. [48].

Conjunto	Variabilidad	Escala	Precisión	Adecuado	Representativo	Disponibilidad	Resolución	Variedades
Škrabánek [38] 2015	✓	✓	✓	✗	✗	✓	40×40	Welschriesling
Seng et al. [50] 2018	✓	✓	✓	✗	✓	✗	–	13 variedades
BIVcolor [45] 2015	✓	✗	✗	✗	✗	✓	2448×2050	–
Kicherer [51] 2016	✗	✗	✗	✗	✗	✓	2448×2050	Riesling
Zabawa [47] 2021	✗	✗	✓	✓	✗	✓	2592×2048	Riesling, Felicia y Regent

Hasta aquí se han explicado los trabajos donde se propusieron soluciones de visión por computadora de la mano con aprendizaje de máquina. Ya que este enfoque requiere imágenes para aprender, se dio un resumen de los esquemas de captura y los conjuntos que incluyeron imágenes de racimos de uvas. Ahora se explican los conceptos alrededor del aprendizaje profundo, y se establece la diferencia principal en comparación a un enfoque de ML tradicional.

2.5. Aprendizaje profundo

El aprendizaje de máquina es un paradigma de computación que consiste en estimar un valor o valores con base en un conjunto de datos. Este paradigma comprende dos tipos de aprendizaje: supervisado y no-supervisado. El primero requiere que los datos de entrenamiento vengan en pares, las variables medidas y la información de referencia; mientras que el segundo no incluye esta información. Se pueden encontrar dos tipos de aprendizaje más: por refuerzo y auto-supervisado, pero esta investigación no se enfocó en ellos.

El enfoque de aprendizaje profundo (DL) presenta dos diferencias principales con respecto al ML clásico. La primera es que, en ML se deben escoger manualmente las características de interés para el proceso de entrenamiento, mientras que en DL, estas características son calculadas durante el proceso de aprendizaje de manera implícita. La segunda radica en la cantidad de veces que los datos son transformados, es decir, en un enfoque de DL, los datos se procesan con mayor profundidad, de ahí el nombre de aprendizaje profundo (Figura 2.13). Ya que el enfoque de DL es de mayor interés para esta investigación, a continuación se explican conceptos relevantes.

2.5.1. ¿Qué tan profundo hay que ir?

A diferencia de los algoritmos de ML clásicos, en el aprendizaje profundo, los modelos aprenden de los datos sin procesamiento previo. Esto libera al desarrollador de construir un vector de características [52] (Figura 2.13), sin embargo, DL utiliza más recurso computacional para lograrlo. Los esfuerzos se enfocan ahora, en construir un conjunto de entrenamiento estable y balanceado para que el modelo de aprendizaje realice su tarea de manera satisfactoria.

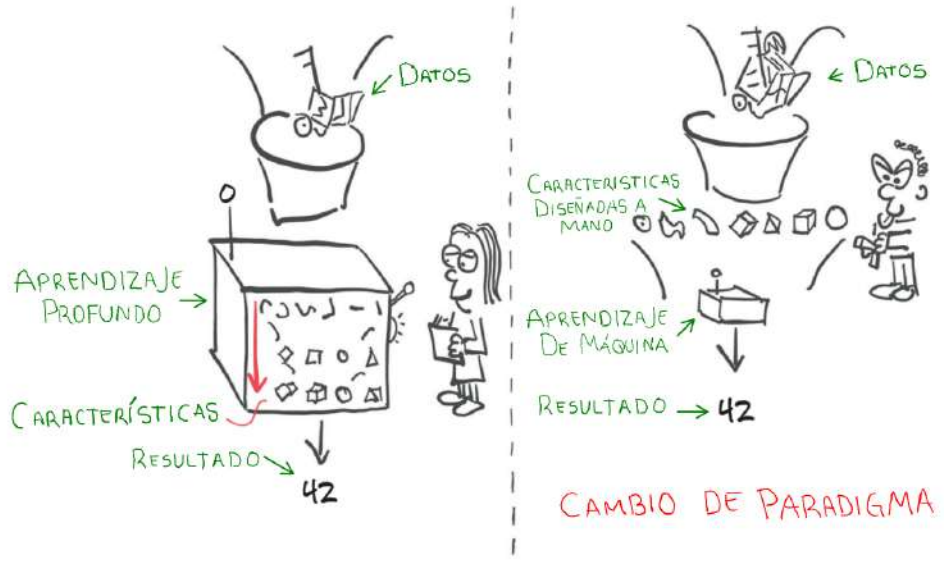


Figura 2.13: El cambio a un paradigma profundo. Aprendizaje profundo intercambia la necesidad de buscar representaciones dentro de los datos, por costo computacional [52].

Este trabajo solo se enfocó en un algoritmo de DL el cual se explica a continuación.

2.5.2. Redes neuronales convolucionales

Un ejemplo de DL son las redes neuronales convolucionales (CNN, por su sigla en inglés para *convolutional neural network*). Ellas hacen uso de conjuntos de capas de *convolución* para procesar cada imagen; una neurona consiste en un proceso de convolución con un kernel 2D y una función de activación. Para llevar a cabo el proceso de aprendizaje, la red actualiza los valores en cada neurona y evalúa su rendimiento con una función de pérdidas (Figura 2.14). También es necesario dividir al conjunto de datos en: entrenamiento y validación. El conjunto de validación permitirá verificar si el modelo entrenado sufrió sobre-ajuste, es decir, el modelo solo aprendió del conjunto de entrenamiento [52].

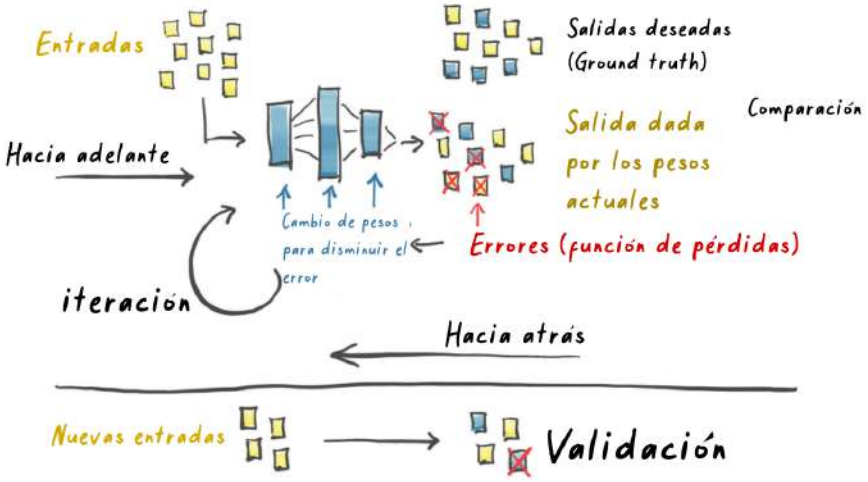


Figura 2.14: Diagrama general del proceso de aprendizaje profundo [52].

2.5.3. Conceptos clave

En esta sección se describen de manera breve los conceptos clave alrededor de las redes convolucionales.

Convolución

Una convolución discreta, consiste en el producto escalar entre una matriz de pesos –función kernel– y una imagen de entrada [52], deslizando el kernel sobre la imagen de izquierda a derecha y de arriba hacia abajo. El objetivo de este proceso es, extraer la información sobre la relación espacial entre píxeles y sus vecinos. Los conceptos alrededor de este procedimiento se explican a continuación:

- **Kernel**: función o matriz 2D que contiene los pesos para realizar el proceso de convolución, generalmente tiene tamaño $3 \times 3 \times d$, donde d representa la cantidad de kernels contenidos en esa matriz (Figura 2.15).
- ***Padding***: relleno de ceros al borde de la imagen de entrada para compensar la convolución.
- ***Stride***: el paso o la tasa de muestreo a la que se hace la convolución.
- **Convolución transpuesta:** sobre-muestreo que aumenta el tamaño de la imagen de salida [53], es usada en las CNNs de segmentación para reconstruir la imagen y es a veces llamada deconvolución (Figura 2.16), aunque matemáticamente, este es un nombre incorrecto.

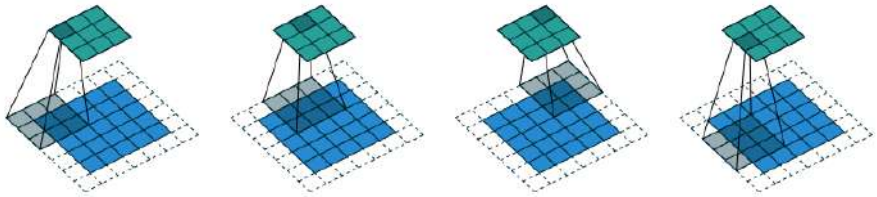


Figura 2.15: Proceso de convolución. Ilustración tomada de [53]. La parte azul corresponde a la imagen de entrada, la gris al kernel y la verde al resultado producto de este procedimiento.

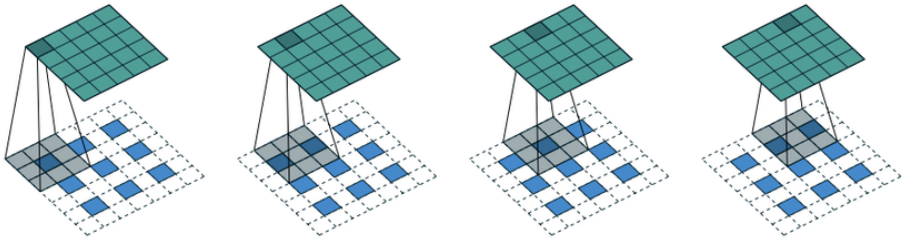
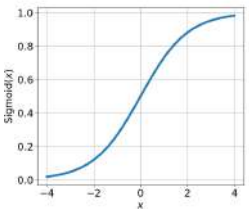
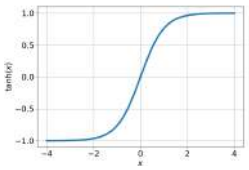
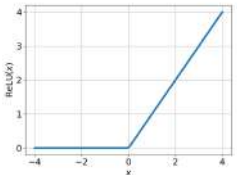
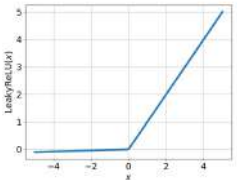
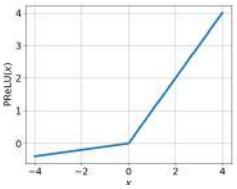
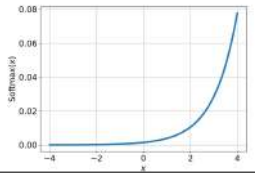


Figura 2.16: Convolución transpuesta. Ilustración tomada de [53]. La parte azul corresponde a la imagen de entrada, la gris al kernel y la verde al resultado producto de este procedimiento.

Función de activación

Compuerta matemática entre la salida de la actual neurona y la entrada a la siguiente. Las versiones no-lineales, se usan para aprender datos complejos [52] (Tabla 2.2).

Tabla 2.2: Funciones de activación. Sumario de las funciones de activación más comunes.

Nombre	Ecuación	Figura
Sigmoidal	$\text{Sigmoid}(x) = \dfrac{1}{1+e^{-x}}$	
Tangente Hiperbólica	$\tanh(x) = \dfrac{e^{x}-e^{-x}}{e^{x}+e^{-x}}$	
ReLU	$\text{ReLU}(x) = \text{máx}(0, x)$	
Lealy ReLU	$\text{LeakyReLU}(x) = \text{máx}(0.01x, x)$	
Parametric ReLU	$\text{PReLU}(x, \alpha) = \text{máx}(\alpha x, x)$	
Softmax	$\text{Softmax}(x) = \dfrac{e^{x_j}}{\sum_{i=1}^{K} e^{x_i}}$	

La función de ReLU y sus variantes son las más utilizadas en las arquitecturas de segmentación y Softmax permite manejo de múltiples clases.

Funciones de pérdidas

Calcula un solo valor numérico que se intentará minimizar. Las pérdidas corresponden a la diferencia entre la salida deseada para un conjunto de entrenamiento y las salidas reales producidas por el modelo cuando se alimenta con ese conjunto [52]. Las funciones de pérdidas comúnmente utilizadas son las siguientes:

- **MSE**: para regresión

$$\text{MSE} = \frac{1}{m}\sum_{i=1}^{m}(y - \hat{y})^2$$

- **Entropía cruzada**: para clasificación

$$E = -\sum_{i=1}^{m}\log(p(y|\mathbf{x}_i))$$

  donde y es el valor real, $\hat{y}$ el valor estimado, $\mathbf{x}_i$ la muestra evaluada y $p(y|\mathbf{x}_i)$ la probabilidad condicional.

Adicionales

- **Normalización por lotes**: normalización estadística que intenta reducir el corrimiento de la distribución a la salida de las capas dentro de las CNN [54]. Se estandariza la entrada x_k:

$$\hat{x}_k = \frac{x_k - E[x_k]}{\sqrt{\text{Var}[x_k]}}$$

  donde $E[]$ es la esperanza y Var[] la varianza y se introducen dos pesos que se aprenderán durante el entrenamiento (γ_k y β_k,):

$$y_k = \gamma_k\hat{x}_k + \beta_k$$

  de tal manera que y_k será el lote normalizado de una transformación lineal de $\hat{x}_k$.

- **Dropout**: Está técnica es usada para prevenir sobre-ajuste y consiste en hacer cero algunas neuronas con una probabilidad $p \in [0, 1.0]$ usando muestras de una distribución tipo Bernoulli [54].

- **Bloques residuales**: introducido por [55] y consiste en sumar a la salida de la capa el valor original de la entrada.
- ***Skip path***: usado por UNet [56], y consiste en concatenar una parte del principio de la CNN a otra cercana al fin.
- **Capa de clasificación**: también llamada capa completamente conectada (FCL) y es una donde ocurre la clasificación [57].
- **Redes totalmente convolucionales**: CNNs que no incluyen una FCL, generalmente usadas para la tarea de segmentación [56, 58, 59].

2.5.4. Arquitecturas de CNN para segmentación

Las arquitecturas definen el camino que va a seguir una red convolucional, es decir, la combinación de operaciones que transformarán a los datos de entrada para generar un resultado. En la literatura se puede encontrar diversas arquitecturas de CNN. Las que abordaron el problema de segmentación semántica tienen una estructura de tipo codificador/decodificador y en este trabajo se exploraron tres arquitecturas de este tipo.

Segnet

Esta arquitectura propuesta por Badrinarayanan et al. [58], también de tipo codificador/decodificador pero con una clasificación píxel a píxel al final. Segnet usó como base la arquitectura **VGG16** [57]. Esta es usada normalmente en la tarea de clasificación, sin embargo, ellos quitaron la capa totalmente conectada de VGG16, permitiendo reducir la cantidad de parámetros de 134 a 14.7 millones. Segnet almacena los índices en la parte del codificador y los usa para reconstruir la salida en la parte del decodificador. Se agrega normalización por lotes entre la función de activación y la capa de convolución (Figura 2.17)

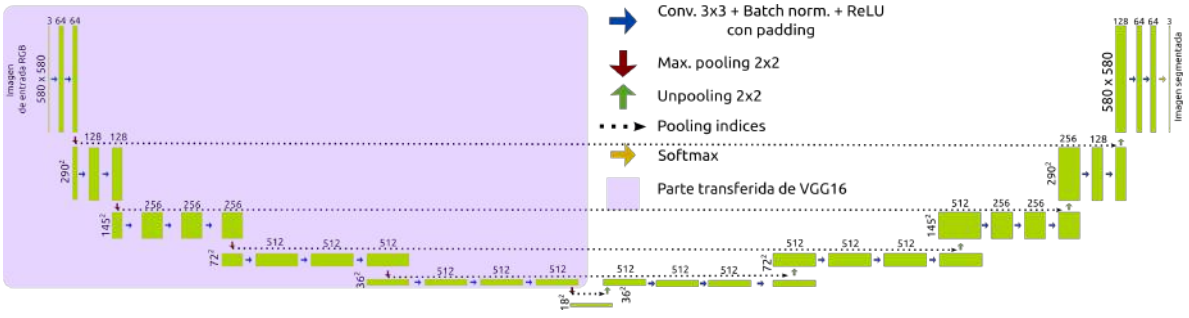


Figura 2.17: Diagrama de la arquitectura de Segnet [58]. Se ilustra en la forma que se presentó el diagrama de UNet [56].

UNet

Ronneberger et al. [56] propusieron una CNN en forma de *U*. UNet aumenta el número de características en la parte decodificador, pero disminuye la resolución de la imagen; mientras que en el decodificador hace el proceso opuesto, concatenando su contraparte correspondiente a la parte del codificador (Figura 2.18). La característica principal de esta red es que no usa relleno a la salida de cada neurona, conservando solo zonas válidas de convolución. Como preprocesamiento, la imagen de entrada es reflejada en sus bordes y se aumenta el tamaño para evitar pérdidas de información. Además, como es efectuado en [60], una parte de la arquitectura VGG16 [57], puede ser transferible a UNet (rectángulo rosa, Figura 2.18).

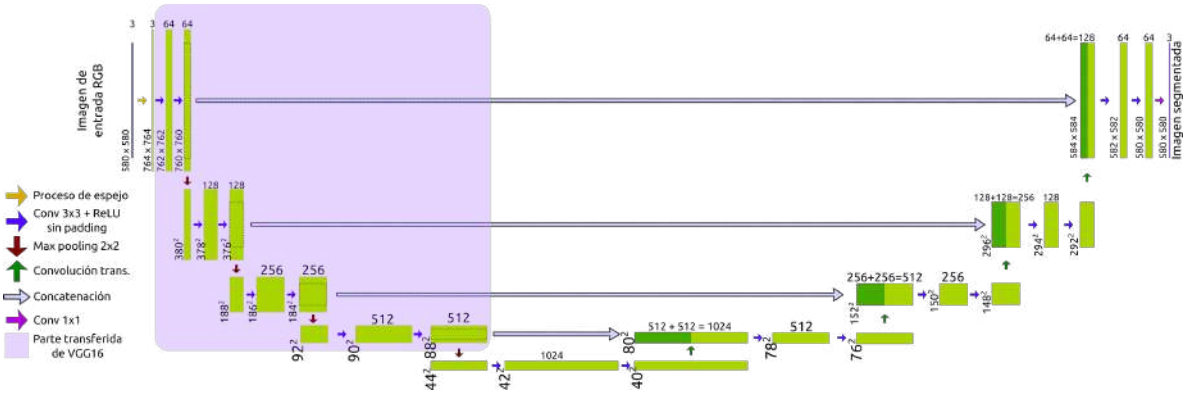


Figura 2.18: Diagrama de la arquitectura de UNet [56].

Zabnet

Esta arquitectura fue usada por Zabawa et al. [16, 9] en el dominio de imágenes de racimos de uvas. Se basó en un híbrido de la arquitectura MobileNetV2 [61] como codificador y de la arquitectura DeepLabV3+ [59] como decodificador (Figura **??**). Zabawa et al. [16, 9] usaron la implementación de Bonnet et al. [62] y el *framework* se encuentra en el siguiente repositorio: https://github.com/PRBonn/bonnet, sin embargo, no hay detalles reportados sobre la adaptación.

MobileNetV2: MobileNetV2 introdujo una manera ligera de convolución: la convolución separable en profundidad (DSC, por su sigla en inglés *depthwise separable convolutions*). La idea principal fue reemplazar el operador de **capa de convolución completa** en dos capas simples:

1. **Convolución en profundidad** (*dwise*): la cual consiste en un filtrado ligero, aplicando un filtro por canal de entrada.

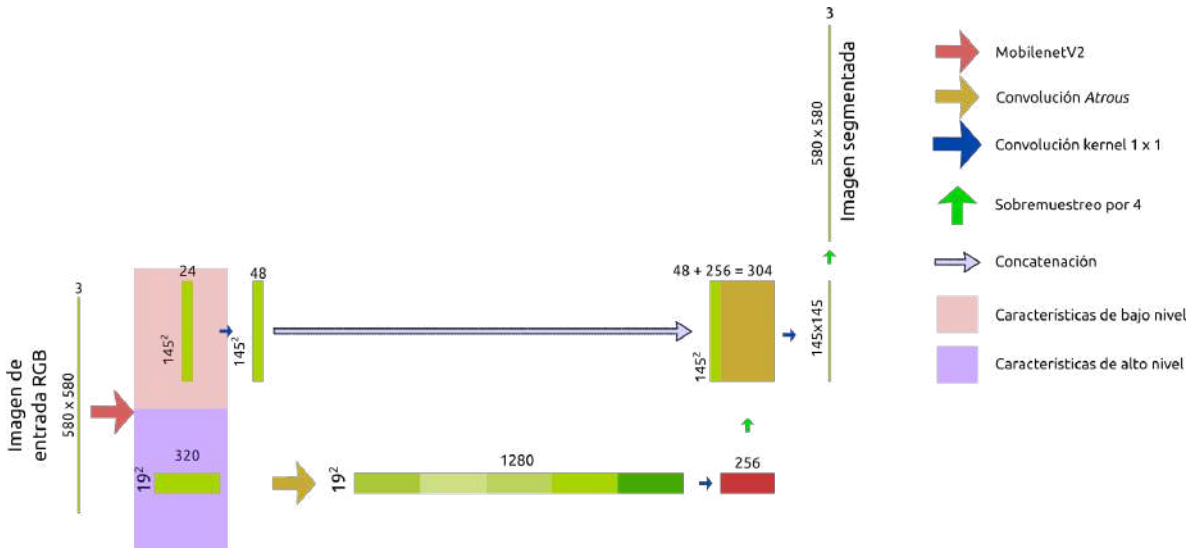


Figura 2.19: Diagrama de la arquitectura usada por Zabawa et al. [16, 9].

2. La segunda capa es llamada **convolución puntual**, ya que es convolución con un kernel k de tamaño 1×1

La capa de convolución estándar toma un tensor $L_i \in \mathbb{R}^{h_i \times w_i \times d_i}$ de entrada, lo convoluciona con un kernel $K \in \mathbb{R}^{k \times k \times d_i \times d_j}$, produciendo un tensor $L_j \in \mathbb{R}^{h_i \times w_i \times d_j}$; con un costo computacional de $h_i \cdot w_i \cdot d_i \cdot d_j \cdot k \cdot k$. Mientras que las DSC, empíricamente cuestan:

$$h_i \cdot w_i \cdot d_i \cdot (k^2 + d_j) \tag{2.1}$$

Esto consiste en la suma en profundidad y la convolución puntual 1×1. MobileNetV2 usó un valor de $k = 3$ (DSCs 3×3), siendo así el costo computacional 8 a 9 veces más pequeño que las capas de convolución estándar, con una pequeña reducción en la exactitud. Además usó un operador-bloque de cuello de botella $\mathcal{F}(x)$ expresado como composición de tres operadores $\mathcal{F}(x) = [A \circ \mathcal{N} \circ B]x$, donde A es una transformación lineal $A : \mathbb{R}^{s \times s \times k} \to \mathbb{R}^{s \times s \times n}$, $\mathcal{N}$ es una transformación no-lineal $\mathcal{N} : \mathbb{R}^{s \times s \times n} \to \mathbb{R}^{s' \times s' \times n}$, B es una transformación lineal al dominio de la salida $B : \mathbb{R}^{s' \times s' \times n} \to \mathbb{R}^{s' \times s' \times k}$, y $\circ$ es la operación de convolución. Para MobileNetV2 $\mathcal{N} = \text{ReLU6} \circ \text{dwise} \circ \text{ReLU6}$ (Figura 2.20) y usa dos valores de *stride* (Figura 2.21).

La arquitectura MobileNetV2 incluye una cantidad menor de capas de convolución en comparación a UNet o Segnet (Tabla 2.3). Las capas que se incluyeron mayormente son los cuello de botella residuales (*Bottleneck*, Tabla 2.4). Donde h y w indican la cantidad de renglones y columnas en la imagen; k y k' indican el número de canales de entrada y de salida, respectivamente; por último, t y s indican el factor de expansión y el tamaño del paso (*stride*).

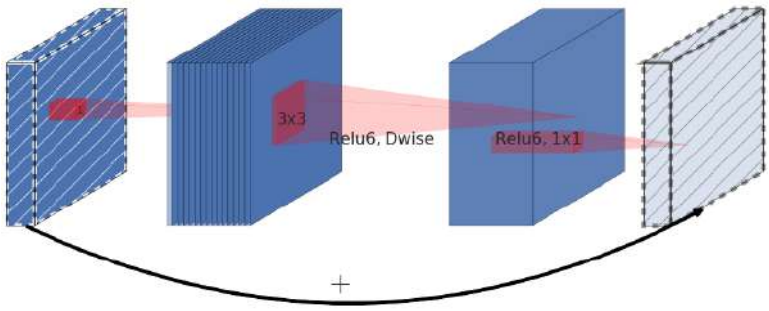


Figura 2.20: Operador-bloque de cuello de botella residual; imagen tomada del trabajo [61].

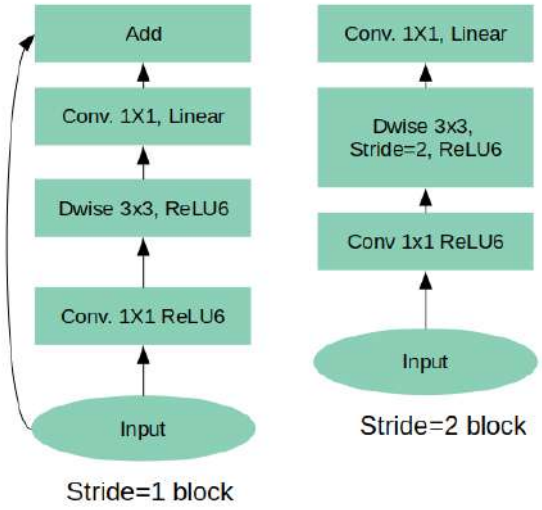


Figura 2.21: Bloques según el valor del *stride*; imagen tomada del trabajo [61].

Tabla 2.3: Estructura de la arquitectura MobileNetV2 [61].

Entrada	**Operador**	**Factor de expansión**	**Canales de salida**	**Repeticiones**	**Strides**
$224^2 \times 3$	convolución 2D 3×3	–	32	1	2
$112^2 \times 32$	*Bottleneck*	1	16	1	1
$112^2 \times 16$	*Bottleneck*	6	24	2	2
$56^2 \times 24$	*Bottleneck*	6	32	3	2
$28^2 \times 32$	*Bottleneck*	6	64	4	2
$14^2 \times 64$	*Bottleneck*	6	96	3	1
$14^2 \times 96$	*Bottleneck*	6	160	3	2
$7^2 \times 160$	*Bottleneck*	6	320	1	1
$7^2 \times 320$	convolución 2D 1×1	–	1280	1	1
$7^2 \times 1280$	avg pool 7×7	–	-	1	-
$1 \times 1 \times 1280$	convolución 2D 1×1	–	Número de clases	–	

Tabla 2.4: Procedimiento general de un *Residual bottleneck block* [61].

Entrada	**Operador**	**Salida**
$h \times w \times k$	Convolución 1×1, ReLU6	$h \times w \times (tk)$
$h \times w \times (tk)$	Convolución dwise 3×3 stride $= s$,ReLU6	$h/s \times w/s \times (tk)$
$h/s \times w/s \times (tk)$	Convolución lineal 2D 1×1	$h/s \times w/s \times k'$

DeepLabV3+: Esta arquitectura fue propuesta por Chen et al. [59]. En ella, aportaron una convolución de tipo *Atrous*, que hace referencia a una convolución dilatada a distintas tasas de muestreo; en cada tasa, el resultado es concatenado para formar una pirámide. Luego, dicha pirámide es convolucionada e incrementada 4 veces su tamaño para concatenarse con el resultado de los niveles bajos del codificador. En la parte del decodificador se refiere a *low-level features* a la salida de las primeras capas de MobileNetV2.

Por último, se hizo una comparación de las tres arquitecturas explicadas anteriormente, resaltando la información más relevante de cada una de ellas (Tabla 2.5). Zabnet presentó la menor cantidad de parámetros a aprender, UNet fue probada en imágenes médicas, donde el nivel de segmentación requerido es alto, y por su parte Segnet fue probada en imágenes de exteriores.

Durante este capítulo se describieron de manera general los trabajos de visión por computadora alrededor de viticultura. Se explicaron los temas relevantes de ML, para

Tabla 2.5: Resumen de las arquitecturas exploradas en este trabajo.

Arquitectura	**Red Base**	**Cantidad de Parémtros**	**Imágenes usadas**	**Característica principal**
Zabnet	DeeplabV3+ MobileNetV2	5.2 M	Racimos de uvas	Convolución atrous.
UNet	Parcialmente VGG16	31.03 M	Médicas	Concanetación de decodificador al final de la capa de. doble convolución.
Segnet	VGG16	29.4 M	Exteriores	Almacen de los índices del *max pooling* para la reconsutrucción de imagen.

abordar la problemática central de esta investigación: detectar uvas individuales dentro de una imagen digital. Se observó que la calidad del conjunto de datos, es una parte esencial para impactar el desempeño de los algoritmos de ML; que el color es una característica particular en los frutos de mismas variedades, y que se requiere un vector de características para entrenar a los algoritmos de ML. Además, se explicaron los conceptos de las redes convolucionales, y las diferencias entre el ML clásico y el aprendizaje profundo. En el siguiente capítulo se explican los materiales utilizados y las propuestas desarrolladas durante esta investigación.

Capítulo 3

Materiales y métodos

Sin importar la aplicación, los algoritmos de aprendizaje de máquina (ML) requieren datos para aprender. Ellos producen un resultado en un dominio definido y la calidad de los datos que se usan para aprendizaje, está altamente relacionada con su desempeño. Las actividades principales del desarrollador de este tipo de soluciones, son: escoger o desarrollar un conjunto de datos para entrenamiento, definir el algoritmo de ML a utilizar y sintonizar los parámetros del algoritmo al problema específico. Este capítulo describe los conjuntos de datos que se desarrollaron durante esta investigación, y las metodologías propuestas para abordar el problema central: *detección de frutos individuales en imágenes de racimos de uvas.*

3.1. Conjuntos de datos desarrollados

El proceso de armar un conjunto de datos es tedioso y propenso a errores. Este proceso es el punto de partida de las soluciones de visión por computadora. Cuando un conjunto de datos está compuesto por imágenes es necesario decidir: i) el esquema de captura de imágenes, ii) el tipo de anotaciones y iii) la información adicional que se va a incluir. A continuación se explica cómo armar un conjunto de datos que cumpla con los requisitos de calidad descritos por Berg et al. [48].

3.1.1. Esquema de captura

A diferencia de los enfoques mencionados en la sección 2.4, este trabajo usó un esquema más conservador. Se utilizó una cámara DSLR Nikon D5100 con un lente 18-55 mm obteniendo imágenes con resolución a 4928×3264 (16 megapíxeles). Se usaron dos tipos fuentes de iluminación: el flash *pop-up* interno de la cámara y un panel LED VILTROX VL-162T CRI95+. Para la configuración de captura, se usó un tripié y se colocó a 30 cm aproximadamente de cada racimo (Figura 3.1).

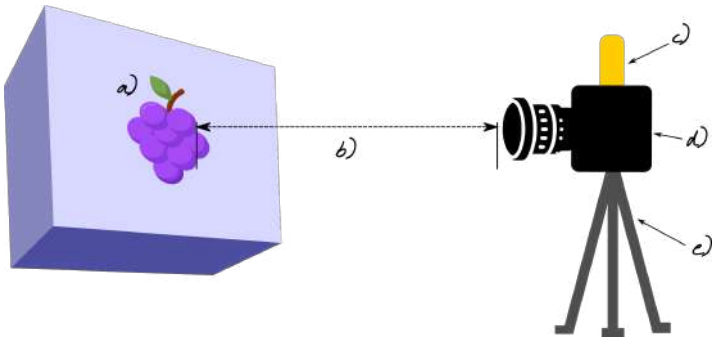


Figura 3.1: Esquema de captura utilizado. (a) Objeto a capturar, (b) distancia entre objetos, (c) fuente de iluminación, (d) dispositivo de captura y (e) soporte del dispositivo.

En este trabajo se realizaron capturas de imágenes en dos viñedos durante dos años. La primer serie de capturas tuvo lugar en Rondo del Valle[1]. Se capturaron imágenes nocturnas a 10 racimos de manera semanal como se puede ver en la (Figura 3.2); las capturas tuvieron fecha entre el 1 de julio y el 14 de agosto del año 2019 y se utilizó como fuente de iluminación el flash *pop-up* de la cámara. Se tuvo acceso a dos variedades de uvas: Merlot y Cabernet Sauvignon y se capturaron 80 imágenes.

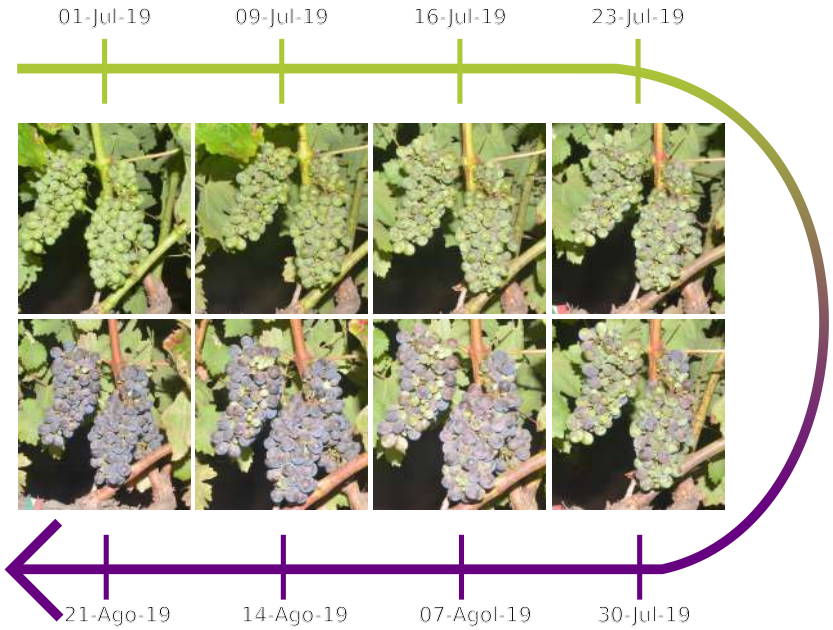


Figura 3.2: Capturas de imágenes en Rondo del Valle. Muestra del mismo racimo capturado en 2019 para observar el cambio de coloración en función a la fecha.

[1]https://www.rondodelvalle.com/, accesado el 19/05/2022

La segunda serie de capturas fue en San Cosme Viñedos[2]. Se capturaron imágenes semanalmente del 26 de junio hasta el 21 de agosto del año 2020 (Figura 3.3). Se tuvo acceso a cinco variedades de uvas tintas: Merlot, Cabernet Franc, Cabernet Sauvignon, Nebbiolo y Syrah y se capturaron imágenes de cinco plantas de cada variedad. Se incluyeron dos tipo de capturas: racimos individuales y plantas completas, junto con tomas diurnas y nocturnas. Para las tomas nocturnas se usó el panel LED mencionado previamente, y para las capturas de plantas completas se colocó el tripié a una distancia de 100 cm aproximadamente.

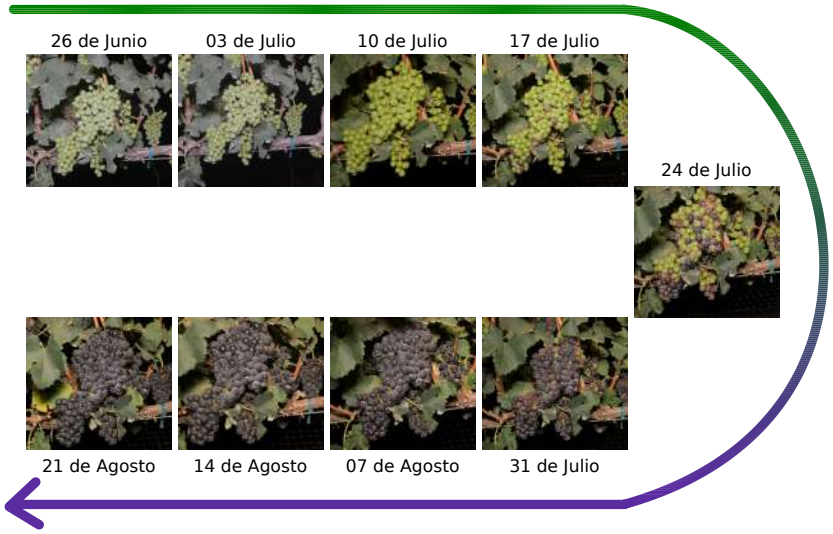


Figura 3.3: Capturas de imágenes en San Cosme Viñedos. Muestra del mismo racimo de la variedad Merlot, capturado en 2020 para observar el cambio de coloración en función a la fecha.

Se realizó una serie de capturas en San Cosme Viñedos el 8 de septiembre de 2020. Se capturaron imágenes diurnas a 30 racimos cosechados en cuatro ángulos distintos (Figura 3.4); se hizo para cada variedad a excepción de Nebbiolo, ya que esta fue cosechada antes de lo esperado. Se incluyó como información adicional el peso real en gramos de cada racimo. En San Cosme Viñedos se capturaron 1,125 imágenes de plantas, 675 de racimos individuales antes de la cosecha y 480 de racimos cosechados, para un total de 2,280 imágenes.

Una vez capturadas las imágenes, el siguiente paso fue hacer un proceso de depuración y etiquetado. Este paso es crucial ya que cumplir la característica de precisión [48] se relaciona con el desempeño de los algoritmos de ML.

[2]https://www.facebook.com/ScosmeVinedos/, accesado el 19/05/2022

Figura 3.4: Última serie de capturas de imágenes realizada en San Cosme Viñedos. Muestra del mismo racimo de la variedad Syrah, capturado el día de la cosecha en donde se observan imágenes a diferentes ángulos con respecto a su eje vertical.

3.1.2. Proceso de etiquetado

Antes de iniciar el etiquetado, se hizo una inspección visual y se eliminaron imágenes con desenfoque o con problemas de iluminación. Para el proceso de etiquetado, se tuvo acceso a las siguientes herramientas: Roboflow[3] y Labelme[4]. La primera es totalmente en línea y gratuita, con opción a comprar una membresía premium, mientras que la segunda es gratuita e instalable. Ambas son amigables con el usuario y generan un archivo JSON[5] con toda la información etiquetada. En este trabajo se usó la herramienta Labelme y para el proceso de etiquetado se recibió asistencia de alumnos de servicio social para el proceso de etiquetado manual.

Para etiquetar las imágenes se utilizaron cuatro tipos de anotaciones: círculos, polígonos, centros y rectángulos para identificar manualmente, racimos o uvas individuales. La anotación corresponde a la forma geométrica a ser dibujada sobre la imagen, mientras que la etiqueta es una palabra que identifica a la anotación. El tipo de anotación a utilizar va a depender de cómo se abordó el problema en cuestión. Por ejemplo, usar un detector de círculos para identificar las uvas dentro de una imagen, requerirá de anotaciones circulares (Figura 3.5-c).

Conjunto de Rondo del Valle

Este conjunto contiene 30 imágenes y se encuentran en el siguiente repositorio: https://github.com/ricglez90/rondods. Se nombró a este conjunto como **RondoDS** ya que incluye solamente imágenes capturadas en Rondo del Valle. Este fue etiquetado manualmente con un círculo por cada uva, dando como resultado una imagen de referencia que contiene separado el racimo del fondo, la cantidad de uvas y su posición (Figura 3.5).

[3]https://roboflow.com/, accesado el 19/05/2022
[4]https://github.com/wkentaro/labelme, accesado el 19/05/2022
[5]Archivo en formato texto que sirve para intercambiar información.

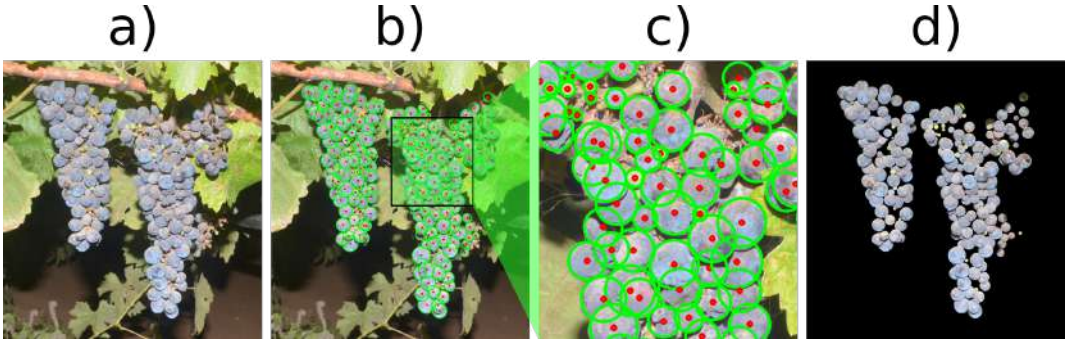


Figura 3.5: Muestra del conjunto de datos RondoDS. (a) La captura original, (b) 211 círculos anotados manualmente, (c) acercamiento a la imagen (b) y (d) la máscara formada al unir los círculos etiquetados.

Conjunto de San Cosme Viñedos

Este conjunto contiene 202 sub-imágenes cuadradas extraídas de 79 imágenes fuente, y fue anotado con un polígono fino por cada uva, a fin de dibujar la forma capturada de las uvas dentro de la escena. Por cada muestra se incluyó una imagen de referencia de la cual se pudo extraer la cantidad de uvas y su posición (Figura 3.6). Se nombró a este conjunto como **CosmeDS**, ya que solo incluye imágenes de las capturas hechas en San Cosme Viñedos. Adicionalmente, se etiquetó manualmente la última serie de capturas hecha el día de la cosecha. Se usó un punto que localiza al centro geométrico de cada uva, y se incluyó el peso real por racimo. La imágenes se encuentran en el siguiente repositorio: https://github.com/ricglez90/cosmeds.

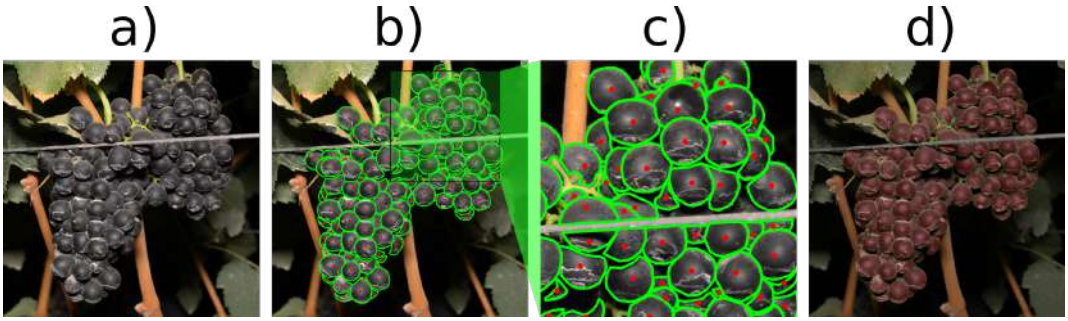


Figura 3.6: Muestra del conjunto de datos CosmeDS. (a) La captura original, (b) 125 polígonos anotados manualmente, (c) acercamiento a la imagen (b) y (d) la máscara formada al unir los polígonos etiquetados.

BIVcolor

Se adaptó este conjunto capturado por la plataforma PHENObot [43] para poder hacer pruebas sobre él. Primero, se identificó cada racimo sobre las 100 imágenes usando un rectángulo como anotación (Figura 3.7-b). Después, se extrajeron 100 sub-imágenes cuadradas que incluyeran racimos con uvas tintas, y se usó un punto como anotación para identificar los centros de uvas individuales (Figura 3.7-d).

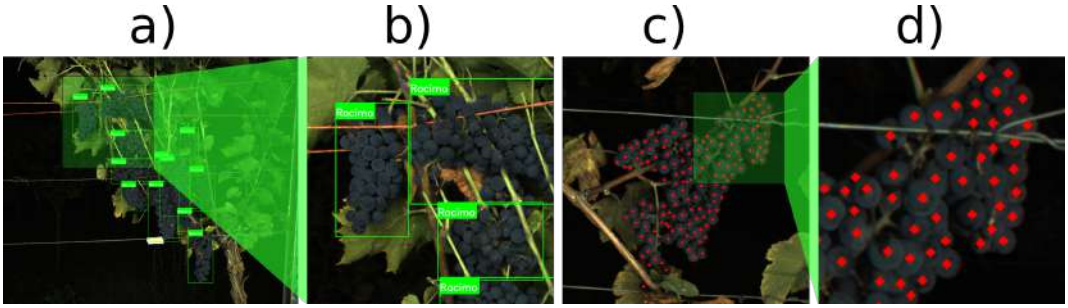


Figura 3.7: Muestra del conjunto de datos BIVcolor [45]. (a) La captura original con 12 racimos anotados manualmente con rectángulos, (b) acercamiento a la imagen (a), (c) sub-imagen anotada manualmente con 146 puntos centrales correspondientes a cada uva y (d) un acercamiento a la imagen (c).

Kicherer

De este conjunto se extrajeron 50 sub-imágenes cuadradas, y también se etiquetaron manualmente los centros de uvas individuales usando como anotación un punto central (Figura 3.8).

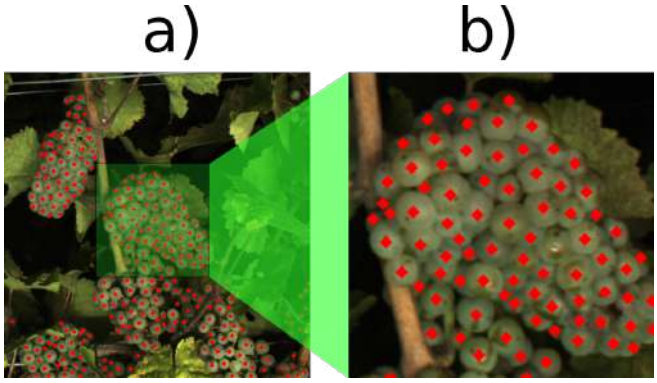


Figura 3.8: Muestra del conjunto de datos Kicherer [51]. (a) Sub-imagen con 276 puntos centrales anotados manualmente correspondientes a cada uva, (b) acercamiento a la imagen (a).

ZabawaDS

Este conjunto ya cuenta con las etiquetas adecuadas (sección 2.4), sin embargo, se hizo una adecuación para poder hacer uso de él. Se extrajeron 202 sub-imágenes cuadradas que sí incluyeran racimos (Figura 3.9-ZabawaDS).

3.1.3. Sobremuestreo de datos

Es necesario que un conjunto de datos cumpla la característica de escalabilidad mencionada en [48]. Esta sugiere que el conjunto de datos debe tener al menos 2000 muestras. Para cumplir con esta característica, se utilizó la técnica de sobremuestreo de datos (DA, por su sigla en inglés *data augmentation*). Esta consiste en aplicar transformaciones aleatorias sobre las imágenes originales [63] y en este trabajo se utilizaron las cuatro transformaciones siguientes:

1. Giro (*flip*): consiste en girar sobre algún eje: vertical, horizontal o ambos. Formalmente definido por las siguientes ecuaciones:

$$\mathbf{A}_v(i,j) = \mathbf{A}(i, M-j-1) \tag{3.1}$$

$$\mathbf{A}_h(i,j) = \mathbf{A}(N-i-1, j) \tag{3.2}$$

$$\mathbf{A}_{vh}(i,j) = \mathbf{A}(N-i-1, M-j-1) \tag{3.3}$$

   donde $\mathbf{A}$ es la imagen original, $\mathbf{A}_{v,h \text{ ó } vh}$ la imagen transformada, N y M el número de renglones y columnas, respectivamente.

2. Difuminado (*blur*): se hizo la operación de convolución a la imagen $\mathbf{A}$ con un kernel 2D K que se precalcula con una distribución Gaussiana, de manera formal:

$$\mathbf{A}_{blur} = \mathbf{A} \circledast K \tag{3.4}$$

   donde $\circledast$ es la operación de convolución, y el tamaño de K define el nivel de difuminado en esta operación.

3. Ruido (*noise*): esta transformación consiste en añadir algún tipo de distorsión a la imagen. El más común es el ruido Gaussiano y de manera formal se define como:

$$\mathbf{A}_{noise} = \mathbf{A} + R_{gauss} \tag{3.5}$$

   donde R_{gauss} es una imagen del mismo tamaño que $\mathbf{A}$ pero con valores aleatorios que provienen de una distribución normal. El valor de la desviación estándar determina el nivel de ruido.

4. Brillo (*bright*): este proceso requiere que la imagen se procese en el espacio HSV. Sea $\mathbf{A}_{hsv}$ la imagen HSV, y la transformación de brillo dada por:

$$\mathbf{A}'_s = \begin{cases} 255, & \text{si } b * \mathbf{A}_s \geq 255 \\ b * \mathbf{A}_s, & \text{cualquier otro caso.} \end{cases} \tag{3.6}$$

$$\mathbf{A}'_v = \begin{cases} 255, & \text{si } b * \mathbf{A}_v \geq 255 \\ b * \mathbf{A}_v, & \text{cualquier otro caso.} \end{cases} \tag{3.7}$$

$$\mathbf{A}_{bright} = \text{RGB}(\mathbf{A}'_{hsv}) \tag{3.8}$$

donde RGB() es la función que transforma del espacio HSV a RGB, $\mathbf{A}'_{hsv}$ la imagen con cambio de brillo en el espacio HSV y b es el factor de brillo que se añade.

Se aplicó sobremuestreo a los conjuntos de CosmeDS y ZabawaDS (Figura 3.9). Nótese que al girar la imagen original se requirió también, transformar la imagen de referencia, a fin de que los píxeles anotados manualmente coincidan con su correspondiente píxel en la imagen original. A continuación, se analizó el contenido de los conjuntos CosmeDS y el desarrollado por Zabawa [47].

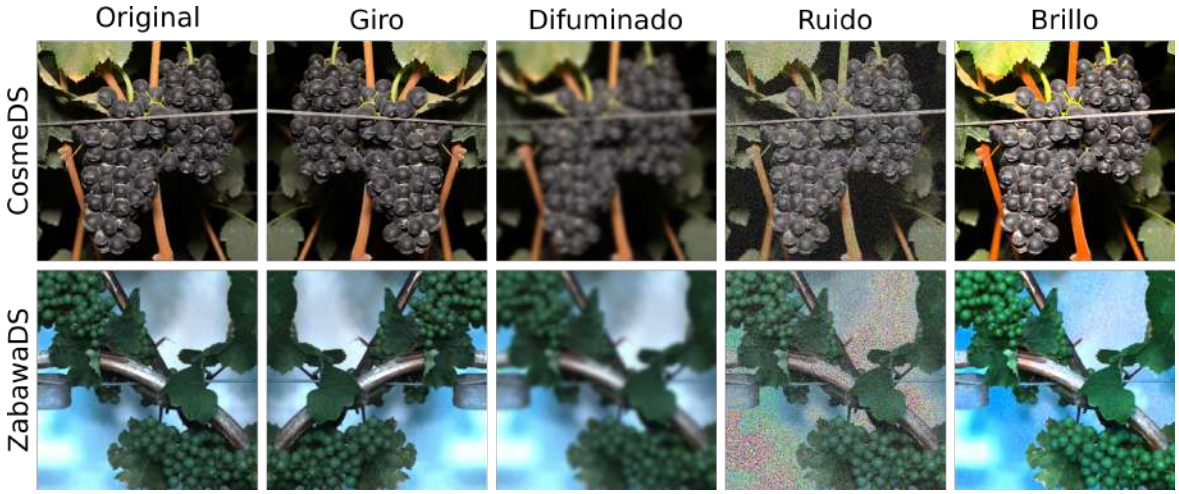


Figura 3.9: Sobremuestreo de datos a los conjuntos CosmeDS y ZabawaDS [47]. Muestra de: giro vertical, difuminado con K de tamaño 31×31, ruido con desviación estándar de 0.25 y brillo con un factor $b = 1.5$.

3.1.4. CosmeDS vs ZabawaDS

El conjunto usado para entrenar un algoritmo de ML debe ser preciso y representativo [48]. En el contexto de esta investigación, este conjunto debe contener imágenes de

referencia que identifiquen a cada fruto individualmente; además, debe incluir la mayor cantidad de variedades de uvas posible y en un mismo número de muestras por cada variedad.

La cualidad de precisión se pudo alcanzar a través de etiquetar las imágenes disponibles, por lo tanto, ambos conjuntos, CosmeDS y ZabawaDS [47] la cumplen. En lo que respecta a la característica de representatividad, el conjunto de ZabawaDS solo incluyó tres variedades de uvas de las cuales, dos son variedades de uvas blancas. Además, la variedad Riesling, las imágenes capturadas el día 31 de Julio y el tipo de siembra SMPH[6], se presentaron con mayor frecuencia en comparación a los demás (Figura 3.10); mientras que para el conjunto de CosmeDS se incluyeron cinco variedades de uvas tintas y solo la variedad Merlot se presentó con una frecuencia ligeramente mayor en comparación al resto de las variedades (Figura 3.11).

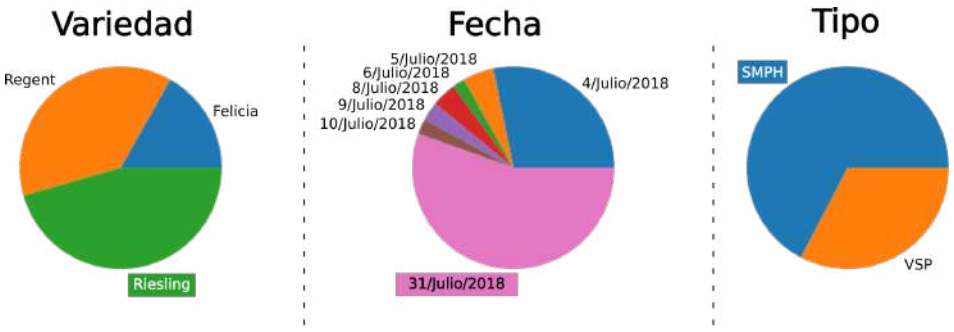


Figura 3.10: Análisis sobre el conjunto ZabawaDS [47]. Gráficas de pastel que ilustran su distribución por: variedad, fecha y tipo de captura .

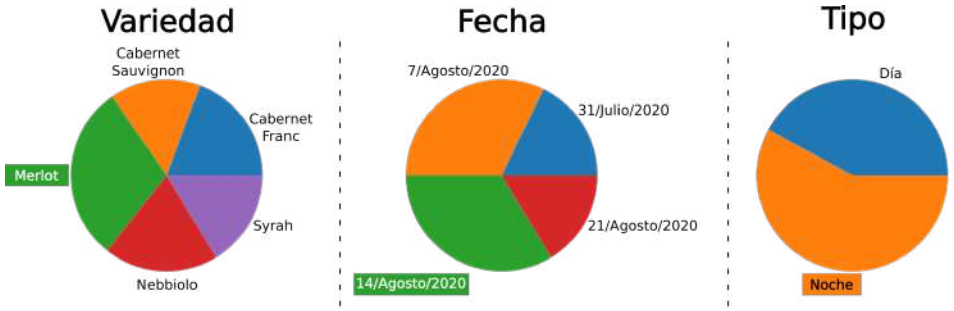


Figura 3.11: Análisis sobre el conjunto CosmeDS. Gráficas de pastel que ilustran su distribución por: variedad, fecha y tipo de captura .

Aunque se observó un ligero desequilibrio en el conjunto CosmeDS, este se presentó en menor proporción comparado con el conjunto ZabawaDS [47]. La forma de las

[6] *Semi Minimal Pruned Hedge*. Tipo de poda de la vid [9]

uvas tiene mayor relevancia en comparación a la variedad a la que pertenece, ya que el problema central de esta investigación fue abordado vía la tarea de segmentación semántica y no clasificación. Este trabajo usó solo imágenes donde todas las formas de las uvas son redondas o casi redondas. La forma de ellas es un factor que se debe tomar en cuenta para abordar el problema de distinguir entre frutos individuales. Por esta razón, se propuso tomar ventaja de la geometría de las uvas. En la siguiente sección se explica cómo modelar dentro de una imagen, la forma de las uvas y cómo explotarla para abordar el problema central de esta investigación.

3.2. La forma de las uvas

La forma de las uvas es determinada por la variedad a la que pertenecen. Esta característica se puede dar de la siguiente manera [64]: 1) Oblonga, 2) Elíptica estrecha, 3) Elíptica, 4) Redonda, 5) Oblata, 6) Aovada, 7) Obtusa-aovada, 8) Obovada y 9) Arqueada (Figura 3.12). Es una suposición lógica abordar el problema vía detección de círculos, ya que las uvas presentan una forma casi circular cuando son capturadas en imágenes digitales. En la siguiente sección se explica el problema de detección de círculos en imágenes.

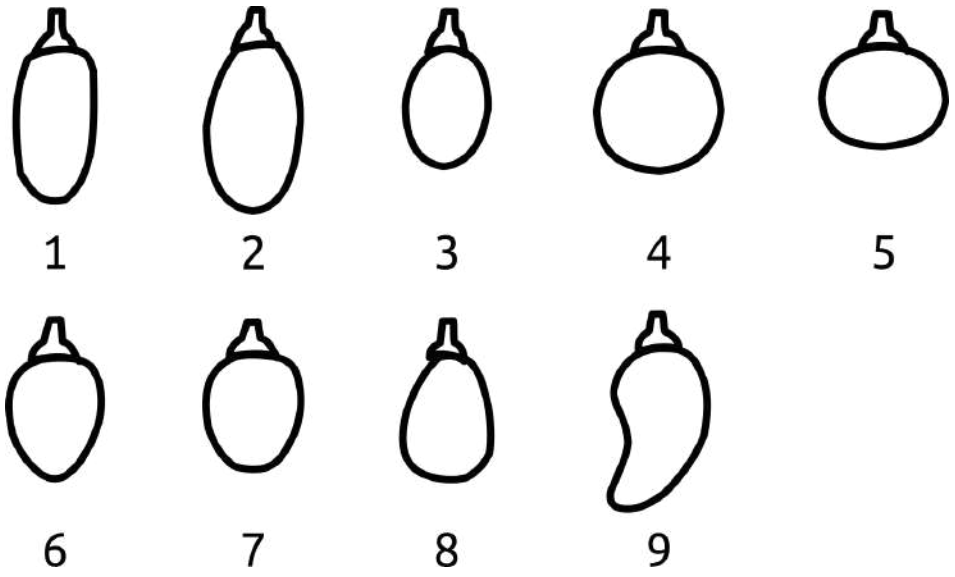


Figura 3.12: La forma de las uvas. Ilustración donde se muestran las variantes de la forma física de las uvas [64].

3.2.1. El problema de detección de círculos

La detección de círculos se ha estudiado ampliamente en las últimas décadas [65, 66, 67]. Básicamente, este problema consiste en localizar todas las formas circulares dentro de una imagen dada. Un enfoque simple, es generar un número finito de círculos con tamaños y posiciones aleatorias para verificar si se pueden incrustar dentro de la imagen de entrada. Después, se evaluá que tan bien encajaron y se da como salida aquel que alcance la mejor evaluación. Una revisión de literatura sobre este problema de detección es presentada por González et al. [67].

La definición formal de la circunferencia en el problema de detección de círculos fue presentada por Chen et al. [68]. Sea S una circunferencia definida por una tripleta $[(x_0, y_0), r]$ que corresponden a sus coordenadas centrales y su radio, respectivamente [69, 70, 71, 72, 65, 73]. Esta tripleta es calculada a través de las ecuaciones 3.9, 3.10 y 3.11; S proviene de tres puntos $(x_i, y_i), (x_j, y_j), (x_k, y_k)$ en su perímetro (Figura 3.13).

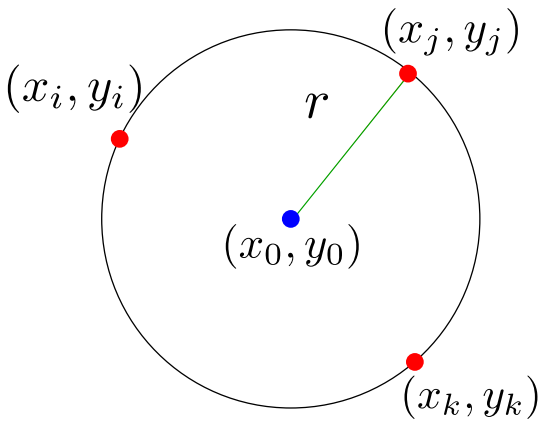


Figura 3.13: Tres puntos en una circunferencia. Ilustración tomada de [67]; se muestran los tres puntos que pasan por el perímetro de una circunferencia.

$$x_0 = \frac{\begin{vmatrix} x_j^2 + y_j^2 - (x_i^2 + y_i^2) & 2(y_j - y_i) \\ x_k^2 + y_k^2 - (x_i^2 + y_i^2) & 2(y_k - y_i) \end{vmatrix}}{4((x_j - x_i)(y_k - y_i) - (x_k - x_i)(y_j - y_i))} \tag{3.9}$$

$$y_0 = \frac{\begin{vmatrix} 2(x_j - x_i) & x_j^2 + y_j^2 - (x_i^2 + y_i^2) \\ 2(x_k - x_i) & x_k^2 + y_k^2 - (x_i^2 + y_i^2) \end{vmatrix}}{4((x_j - x_i)(y_k - y_i) - (x_k - x_i)(y_j - y_i))} \tag{3.10}$$

$$r = \sqrt{(x_{i,j \circ k} - x_0)^2 + (y_{i,j \circ k} - y_0)^2} \tag{3.11}$$

Para detectar círculos se requiere un mapa de bordes y para esto, es necesario transformar la imagen de entrada a un espacio donde se identifiquen los píxeles que corresponden a las fronteras entre objetos.

3.2.2. La importancia de los bordes

Los bordes dentro de una imagen corresponden a los píxeles que separan objetos o al fondo [74]. Definen la frontera entre cada objeto y es crucial para la detección de círculos. Durante esta investigación, se encontró que la mayoría de las soluciones para este problema, usaron como entrada un mapa de bordes para la detección. Sea $\mathbf{E} \in \mathbb{R}^{M \times N}$ este mapa donde:

$$\mathbf{E}(i,j) = \begin{cases} 1, & \text{Si } \mathbf{A}(i,j) \text{ es un píxel de borde} \\ 0, & \text{Cualquier otro caso} \end{cases} \tag{3.12}$$

donde $\mathbf{A} \in \mathbb{R}^{M \times N}$ es una imagen 2D en escala de grises. Uno de los detectores de bordes ampliamente usado es el método de Canny [75]. En el trabajo de Parker et a. [74] se mencionó que este detector de bordes abordó tres problemas principalmente:

- La tasa de error: todos las fronteras entre objetos deben ser detectadas correctamente.
- Localización: la distancia entre el borde detectado y el real debe ser cercana a cero.
- Respuesta: los bordes múltiples no se deben confundir con bordes sencillos.

En la literatura se pudo encontrar otros detectores de bordes tales como: Sobel [76], Prewitt [76] y EDPF [77] (*edge drawing parameter free*, por su sigla en inglés). Sin embargo Canny es el más utilizado para detección de círculos.

Además, cuando las imágenes se ven afectadas por ruido, es decir, hay píxeles que distorsionan la escena original, es necesario agregar una etapa de filtrado previa al detector de bordes, ya que de lo contrario, se verá afectado el desempeño del detector de círculos. Los filtros más usados son: el filtro de medianas y el filtro Gaussiano [18]. En lo siguiente se explica el método propuesto por esta investigación para detectar círculos.

3.2.3. Un detector de círculos usando un algoritmo genético

Los algoritmos genéticos (GA) son técnicas de optimización inspiradas en la selección natural y la supervivencia del más apto; fueron introducidos por Holland en los 60's [83] y este tipo de métodos deben incluir las siguientes características [84]:

1. Una población de soluciones, llamados individuos,
2. una forma de evaluar qué tan buena es cada solución, esta función se llama función de *fitness* u objetivo,

3. un método combinatorio de fragmentos de soluciones con *fitness* alto, para generar nuevas soluciones. A este método se le conoce como operador de combinación o cruzamiento, y

4. un método que ayude a mantener la diversidad en la población, también conocido como operador de mutación.

Teniendo una población de círculos descritos por tres puntos en su perímetro, un GA puede evolucionarla y al ser asistido por una función objetivo puede encontrar formas circulares dentro de una imagen (Figura 3.14). Se nombró a esta propuesta CDSGA (por su sigla en inglés *Circle detector by a simple genetic algorithm*).

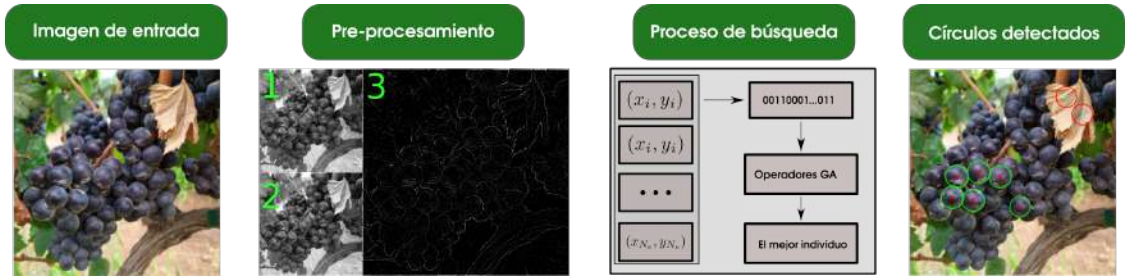


Figura 3.14: Esquema del proceso de detección de círculos.

A continuación se muestran los pasos que sigue CDSGA para detectar círculos:

1. Se toma una imagen $\mathbf{A}_{gs}$ en escala de grises y se propuso usar un filtro de mediana, para reducir el ruido; usando Canny se calcula el mapa de bordes $\mathbf{E}$.

2. Se almacena todas las posiciones de píxeles de borde en forma (x, y); sea V un arreglo con N_e coordenadas tal que: $V = \{(x_1, y_1), (x_2, y_2), ..., (x_{N_e}, y_{N_e})\}$.

3. Se crea la población de círculos candidatos; sea S_i una concatenación de tres puntos aleatorios en V y calculando centro (x_0, y_0) y su radio r con las ecuaciones 3.9, 3.10 y 3.11, respectivamente, cada solución se almacena para el proceso evolutivo.

4. La población creada pasa por un proceso estándar de evolución vía un GA, y se detiene cuando un criterio se alcanza.

5. El círculo más apto $S^* = [(x^*, y^*), r^*]$ en la última generación es seleccionado.

6. Las posiciones de V que pasan por S^* son eliminadas y el proceso se repite desde el paso 3 hasta que se cumple un criterio de parada o V se vacíe por completo.

Para evaluar un círculo candidato S, se crean puntos de prueba creados alrededor de él y se comparan con los píxeles de borde existentes en $\mathbf{E}$ [71, 65]. Formalmente,

sea $T(S) = \{t_1(S), t_2(S), ..., t_n(S)\}$ un arreglo con n puntos de prueba $t_i(S) = (x_i, y_i)$ generados por las ecuaciones 3.13 y 3.14. Se nombró a este método TP (por su sigla en inglés, *test points*).

$$x_i = x_0 + r\cos\frac{2\pi i}{n} \tag{3.13}$$

$$y_i = y_0 + r\sin\frac{2\pi i}{n} \tag{3.14}$$

Sin embargo, TP no es el único método para crear estos puntos. El algoritmo de círculo de punto medio (MCA, por su sigla en inglés *Midpoint Circle Algorithm*) define la cantidad de puntos para dibujar un círculo dentro una imagen digital. MCA requiere una coordenada central (x_0, y_0) y un valor de radio r; este algoritmo calcula solamente los puntos sobre el primer octante de la circunferencia y replica en espejo al resto, es decir, si se calculan (x, y) los puntos $\{(y, x), (-y, x), (-x, y), (-x, -y), (-y, -x), (y, -x), (x, -y)\}$ también son calculados. MCA inicia en $(x_0 + r, 0)$ y moviéndose hacía la izquierda finaliza en (x, y) cuando $y > x$; esto indica que todo el primer octante ha sido calculado. Una vez que el primer punto t_i se calcula el siguiente punto es calculado de la siguiente forma:

$$t_{i+1} = \begin{cases} (x_i, y_i + 1), & \text{si } (x_i - 0.5 + x_0)^2 + (y_i + 1 - y_0)^2 - r^2 \leq 0 \\ (x_i - 1, y_i + 1), & \text{cualquier otro caso} \end{cases} \tag{3.15}$$

MCA fue usado por Cuevas et al. [69, 70] y Lopez et al. [72]. Dado un círculo candidato $S = (x_0, y_0, r)$, su calidad se calcula como:

$$ev(S) = \sum_{j=1}^{nte} P[t_j(S)] \tag{3.16}$$

donde:

$$P[t_j(S)] = \begin{cases} 1, & \text{si la distancia entre } t_j \text{ y } V \text{ es menor a } e \\ 0, & \text{cualquier otro caso.} \end{cases} \tag{3.17}$$

Recordar que V almacenó las posiciones de píxeles de borde, mientras que e, determina la distancia máxima permitida para que un borde se considere vecino y es fijado a $\sqrt{2}$ [85]. Una de los aportes de esta investigación fue el cálculo dinámico de la cantidad de puntos de prueba generados para evaluación [67].

Ahora se describe los pasos del proceso realizado por el algoritmo genético:

- **Codificación:** cada solución se compone de una concatenación binaria de los índices de tres puntos en V (Figura 3.15). Esta representación fue usada también por Cuevas et al. [69, 70], Dong et al. [71] López et al. [72] y Ayala-Ramírez [65].

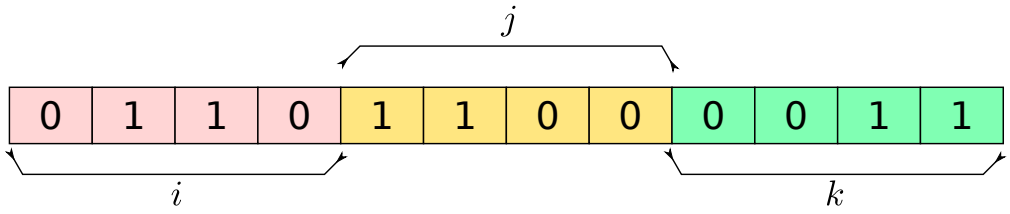


Figura 3.15: Ejemplo de una solución codificada. Los tres índices $i = 6, j = 12$ y $k = 3$.

- **Creación de la primer población:** ya que el espacio del mapa de bordes está contenido en N renglones y M columnas; es lógico suponer que el diámetro de los círculos a encontrar no podrá exceder estas dimensiones. Se fijó un diámetro máximo d_{max} de la siguiente manera:

$$d_{max} = \begin{cases} N, & \text{si } N \leq M \\ M, & \text{cualquier otro caso.} \end{cases} \tag{3.18}$$

  Se propuso iniciar la primer generación de soluciones siguiendo los siguientes pasos:

  1. Generar un número aleatorio i desde 1 hasta N_e (longitud de V).
  2. Generar un número aleatorio j desde 1 hasta N_e, hasta que la distancia entre V_i y V_j sea menor o igual a d_{max}.
  3. Generar un número aleatorio k desde 1 hasta N_e, hasta que la distancia entre V_k y V_j sea menor o igual a d_{max}, y la distancia entre V_k y V_i también lo sea.
  4. $i \neq j \neq k$ /* No seleccionar puntos repetidos */
  5. Concatenar en código binario i, j y k.

  Hacer esto para cada solución en la población.

- **Función de aptitud:** Se toman los valores de V en V_i, V_j, V_k y se calcula un círculo candidato usando las ecuaciones 3.9, 3.10, y 3.11 para obtener un círculo S y crear los puntos de prueba T, la función de *fitness* se calcula de la siguiente manera [69, 70, 71, 72, 65]:

$$F(S_i) = \frac{\sum_{j=1}^{nte} P[t_j(S_i)]}{n} \tag{3.19}$$

  Entonces, de todos los círculos S_i el más apto es $S^* = [(x^*, y^*), r^*] = \text{argmax} \quad F(S_i)$ al término del proceso evolutivo del GA y por tanto es un círculo detectado. Notar que $F()$ evalúa qué tan bien encaja un círculo en el mapa de bordes $\mathbf{E}$ y que además se encuentra en el intervalo $[0, 1]$, donde un valor cercano a 1 representa

un círculo perfectamente formado, y uno cercano a 0 un círculo que no existe en la imagen evaluada.

- **Operadores de selección y variación:** el método de ruleta y reemplazo generacional son usados aquí, como fueron usados por Ayala-Ramírez et al. [65]; el cruzamiento de 1 punto y la mutación de tipo *bit-wise* también fueron usados.
- **Vaciando V:** se propuso tomar el círculo más apto S^* de este proceso evolutivo y calcular la distancia entre su centro (x^*, y^*) y cada punto de V; si es menor a $r^* + \varepsilon$ y mayor que $r^* - \varepsilon$, ese punto se elimina del vector de posiciones V. Esto reduce el tamaño de V y por lo tanto el tiempo de cómputo en la siguiente iteración, este proceso se hace de manera iterativa para encontrar múltiples círculos.

La intención de identificar uvas a través de un detector de círculos es tomar ventaja de su forma geométrica y relacionar el número de círculos detectados con la cantidad de frutos contenidos en la escena. Aunque se sabe que las uvas están aglomeradas y esto aumenta la dificultad de detectarlas. Por esta razón, se generó la primer propuesta de detección, en donde primero se localiza el racimo dentro de la imagen.

3.3. Localización de racimos en imágenes

El proceso general propuesto incluye un marco de referencia para localizar racimos de uvas dentro de imágenes digitales. Después estimar el número de frutos para luego calcular el área de cada uno, a nivel de píxel y por último determinar el área ocluida de los frutos localizados (Figura 3.16).

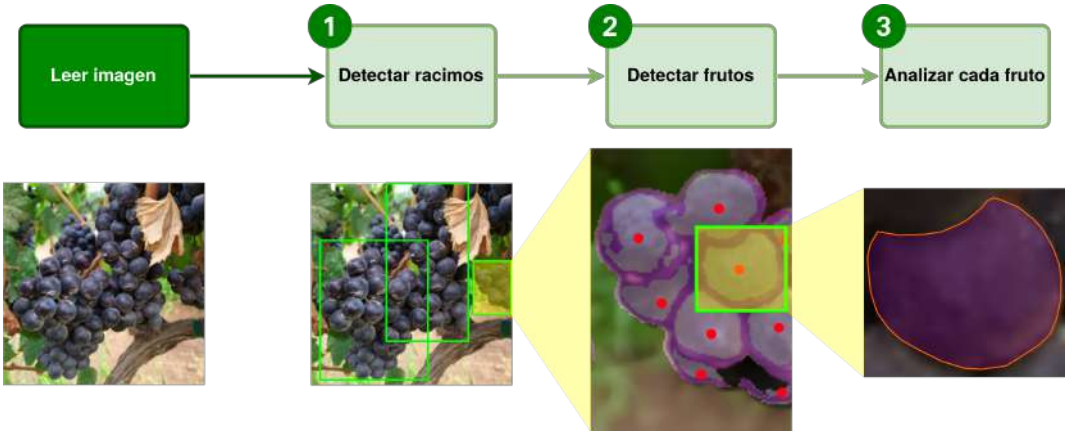


Figura 3.16: Diagrama general de la propuesta en esta investigación. Localizar los racimos (1), frutos individuales (2) y analizar cada fruto (3).

Con el objetivo de localizar regiones correspondientes a racimos de uvas tintas en imágenes, se propuso un enfoque de ML a través de clasificar cada píxel de las imágenes

de evaluadas. Los pasos para el proceso de clasificación píxel-a-píxel son los siguientes (Figura 3.17):

1. Se extrajeron las características interés basadas en el color de la imagen original.
2. Se entrenó un algoritmo de ML con estas características para identificar los píxeles correspondientes a racimos de uvas.
3. Se localizaron los cúmulos de píxeles contiguos, para determinar las regiones que corresponden a los racimos.

En lo siguiente se explica cómo se definió este enfoque de clasificación.

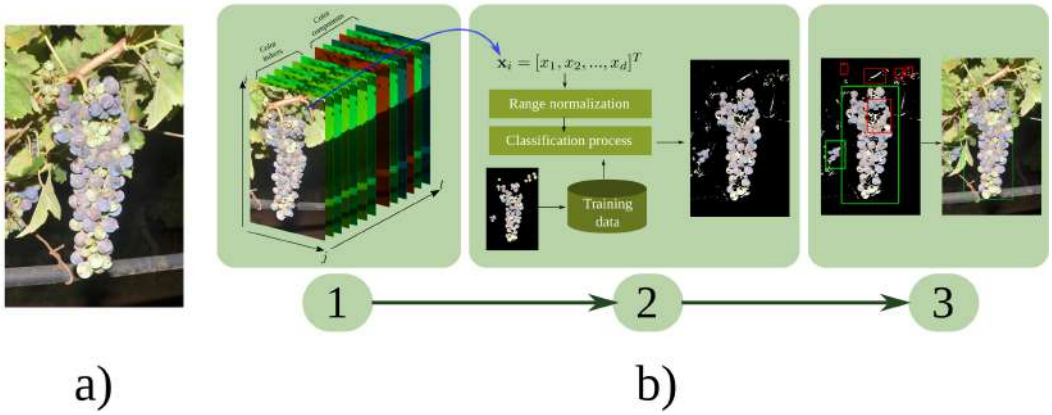


Figura 3.17: Diagrama del enfoque píxel-a-píxel. La imagen (a) corresponde a la imagen de entrada y la (b) al proceso general de la clasificación píxel-a-píxel, para identificar los racimos.

3.3.1. Clasificación píxel-a-píxel

La estrategia de clasificar píxel-a-píxel (*pixel-wise*) junto con un vector de características basado en el color fue una de las propuestas en este trabajo. Este enfoque de clasificación surge de la necesidad de localizar las regiones que corresponden a los frutos y fue usada por Abdelghafour et al. [42] en este dominio de imágenes. Dentro de una imagen digital persisten en su mayoría, píxeles que corresponden al follaje, tallos y objetos que no caen en la categoría de frutos. Por esto, identificar de manera correcta los píxeles que corresponden a los frutos es una tarea retadora. La etapa 1 consistió en construir un vector que cuantifique las características de color para cada píxel (Figura 3.17-1); se propuso usar 22 valores de intensidad por cada píxel, donde cada valor corresponde a:

- 10 índices de color, usados comúnmente para separar plantas verdes del suelo, los detalles son explicados por Hamuda et al. [40] (Apéndice A.1.3).
- 12 componentes de color, de los espacios de color RGB, HSV, CIELab y YCbCr, detalles encontrados en el trabajo de García et al. [86] (Apéndice A.1.2).

Para la etapa 2 se emplearon algoritmos de ML que abordaran la tarea de clasificación (Figura 3.17-2). Estos se entrenaron con una muestra aleatoria del 5 % del total de píxeles de alguno de los conjuntos desarrollados en esta investigación, a fin de construir un conjunto de entrenamiento que tenga el tamaño siguiente: $0.05 \cdot \hat{N} \times 22$. El número 22 corresponde al tamaño del vector de características, y $\hat{N}$ al total de píxeles dentro del conjunto utilizado.

Por último, la etapa 3 incluyó las siguientes sub-etapas (Figura 3.17-3):

1. **Regularización:** como es explicado por Abdelghafour et al. [42] este proceso consistió en incluir la relación espacial de los píxeles con sus vecinos, debido a que la clasificación píxel-a-píxel no incluye esta relación. Para esto se aplicaron operadores morfológicos, los cuales consisten en convolucionar la imagen I_y resultado de la clasificación con un kernel K en 2D. Estos operadores son comúnmente los siguientes [18]: erosión y dilatación y su definición se puede encontrar en las ecuaciones 3.20 y 3.21, respectivamente.

$$I_y \ominus K = \{p \in \mathbb{R}^2; p + k \in I_y \text{ para cada } k \in K\} \tag{3.20}$$

$$I_y \oplus K = \{p \in \mathbb{R}^2; p = p + k, p \in I_y \text{ y } k \in K\} \tag{3.21}$$

   Además, si se aplican los dos operadores morfológicos en el orden siguiente: erosión-dilatación, se podrá remover ruido o píxeles aislados (Figura 3.18-izquierda); este proceso es llamado *apertura* y su definición formal es presentada en la ecuación 3.22. Por su parte, si se aplican los operadores en el orden inverso, es decir dilatación-erosión, se podrán rellenar huevos que hayan quedado en la imagen (Figura 3.18-derecha); este proceso es conocido como *cierre* (ecuación 3.23).

$$I_y \bigcirc K = (I_y \ominus K) \oplus K \tag{3.22}$$

$$I_y \centerdot K = (I_y \oplus K) \ominus K \tag{3.23}$$

2. **Umbral de áreas:** para esta etapa se identificaron los objetos dentro de la imagen resultado de la fase regularización y se descartaron aquellos cuyas áreas están por debajo de un valor de umbral fijo.

Figura 3.18: Implementación de operadores morfológicos [87], izquierda: apertura, derecha: cierre

3. **Áreas traslapadas:** para esta etapa se aplicó un proceso llamado supresión no-máxima (NMS, por su sigla en inglés *non-maximum suppresion*) y es usado para identificar áreas totalmente traslapadas.

Una vez localizado el racimo, se propuso estimar el número de frutos contenidos dentro del racimo.

3.4. ¿Cómo localizar uvas en una imagen?

La propuesta consistió en detectar círculos dentro de las regiones correspondientes a racimos (Figura 3.19-2). Después en extraer una sub-imagen cuadrada y evaluar si el círculo candidato realmente correspondía a una uva (Figura 3.19-2.1.3). El vector de características utilizado para este proceso, fue una concatenación de tres descriptores: gradientes (HOG), textura (LBP), y el color (FCH) los detalles de cada descriptor son explicados en el Apéndice A.1.4. Pérez-Zavala et al. [12] propusieron un esquema similar, solo que ellos detectaron las uvas candidatas a través de buscar picos de reflexión de luz dentro de la imagen, y no incluyeron características de color para representar cada uva evaluada.

Ya que la detección de círculos puede ser realizada por cualquier detector, se tienen disponibles públicamente: la transformada de Hough para círculos (CHT) [88], EDcirlces [66] o CDSGA, este último corresponde a una de las propuestas realizadas en esta investigación [67]. Por último, se exploró una solución que incluyó aprendizaje profundo, a fin de segmentar frutos individuales; este enfoque es el más complejo que este trabajó exploró.

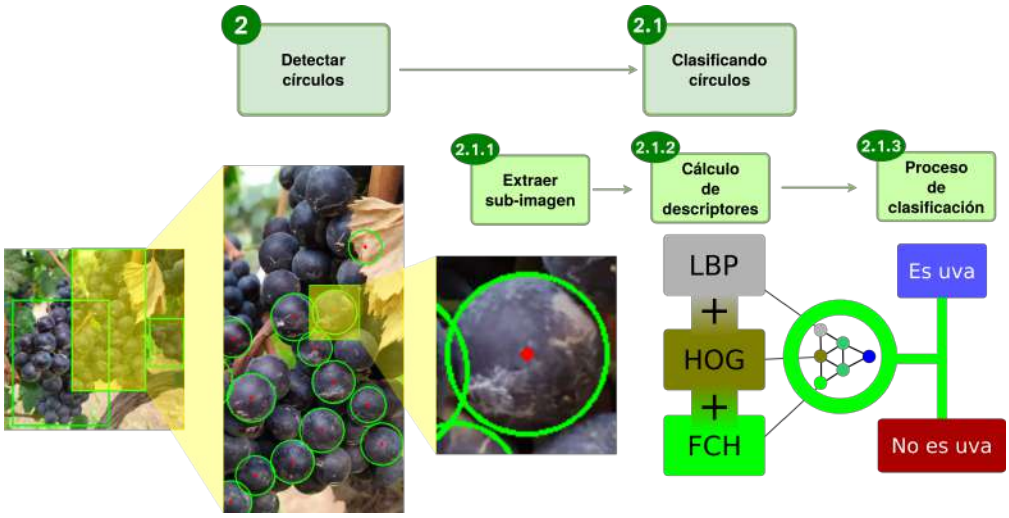


Figura 3.19: Diagrama clasificación de círculos candidatos.

3.5. Segmentación profunda

En esta sección se propuso utilizar un enfoque de aprendizaje profundo, a través del algoritmo de redes convolucionales (CNN). Se utilizaron arquitecturas de CNN de tipo codificador/decodificador, para segmentar frutos individuales dentro de un racimo. El objetivo de este tipo de enfoques es aprender de un conjunto de imágenes de referencia, donde se hayan identificado cada fruto individual. Ejemplos de un conjunto que cumple estas características pueden ser: ZabawaDS [47] o el conjunto CosmeDS desarrollado en esta investigación. Algunas de las CNN propuestas en la literatura y que abordan la tarea de segmentación pueden ser: Segnet [58], UNet [56] o DeepLabV3+ [59]. Las categorías a aprender, que se incluyeron en los conjuntos previamente mencionados son: "fondo", "uva" y "borde", la última corresponde al borde de cada uva, como fue propuesto por Zabawa et al. [9].

Se pueden encontrar tres problemas persistentes en este tipo de soluciones: que el número de muestras para entrenamiento no sea suficientes, que las categorías a aprender aparezcan fuera de balance y que el recurso computacional disponible sea escaso.

El primer y último problema, se pueden contrarrestar usando una técnica llamada *transferencia de conocimiento*, mientras que el segundo se puede manejar usando una función de pérdidas ponderada durante el proceso de aprendizaje. En lo siguiente se explica de manera breve cada técnica.

3.5.1. Transferencia de conocimiento

Generalmente, antes de iniciar el proceso de entrenamiento desde cero de una CNN, se recomienda repasar las siguientes preguntas [89]:

- ¿Se puede cuantificar el grado de aprendizaje en una CNN? es decir ¿qué parte de la red aprende generalidades, y cuál aprende la tarea específica?
- ¿Esta transición ocurre en alguna parte puntual o se transmite a toda la CNN?
- ¿Dónde ocurre esta transición: al principio, en el centro o al final de la red?

Como lo explicaron Yosinski et al. [89] se puede transferir una parte de una red entrenada y re-entrenarla a una tarea y/o dominio de datos nuevo. La nueva tarea puede ser clasificación o segmentación, mientras que el dominio dependerá de los datos de entrada. En ciencias de la computación se le conoce a este proceso como transferencia de conocimiento (*transfer learning*). Una vez transferida la parte de interés, hay dos maneras de realizar el entrenamiento. La primera consiste en dejar que la parte transferida se actualice durante el re-entrenamiento, mientras que la segunda, consiste en congelar los valores transferidos y no modificarlos. Nos referimos a los modos de entrenamiento con los siguientes nombres:

1. ***From-scratch***: entrenar sin hacer transferencia.
2. ***Freezee***: transferir y congelar los valores transferidos.
3. ***Fine-tune***: hacer la transferencia de los valores y permitir que se actualicen.

Yosinski et al. [89] concluyeron que entrenar una red en modo *fine-tune* es preferible a hacerlo desde cero, y que además, el modo *freeze* presentó poca adaptación al nuevo dominio de los datos. En esta investigación se probaron los tres modos de entrenamiento explicadas previamente.

Esta técnica se usó debido a que el número de muestras disponibles para entrenar es pequeño, y que además el recurso computacional disponible tiene limitaciones de memoria.

3.5.2. Manejo de clases sin balance

El problema de clases sin balance se refiere a tener en un conjunto de datos, una categoría con mayor proporción que otra. Para manejar esta situación se emplean algunas técnicas [90]. Una de ellas, consiste en sobre-muestrear la clase con menos muestras o sub-muestrear la clase con más presencia. Pero la más común consiste en penalizar la función de pérdidas con un factor de ponderación, basado en la probabilidad de ocurrencia de cada clase. Para la tarea de segmentación semántica, la última opción es la más viable, ya que muestrear las clases a nivel de píxel compromete la relación espacial de las imágenes, y el desempeño de la segmentación se puede ver afectado.

Para calcular el factor de ponderación para cada clase este trabajo utilizó la siguiente ecuación [91]:

$$s'_x = 1 - c_s + \frac{c_s}{p_x N_c} \tag{3.24}$$

donde s'_x es el factor de la clase x, $c_s \in [0, 1]$ es una constante que especifica la cantidad de escalamiento, p_x es la probabilidad de la clase x y N_c es el número de clases. Notar que si $c_s = 0$ no hay factor de ponderación, Lawrence et al. [91] propusieron un valor de c_s cercano a 0.8.

Para terminar esta sección, se explica la propuesta de cómo predecir el área oculta de uvas parcialmente ocluidas.

3.6. Área ocluida

Esta propuesta tomó ventaja de los bordes de las uvas localizados por las CNNs entrenadas. Como se mencionó anteriormente, reconocer las fronteras de los objetos de interés dentro de imágenes, es una de las tareas con mayor relevancia en las soluciones de visión por computadora.

El primer paso de esta propuesta consistió en almacenar las posiciones (x, y) muestreadas de manera equidistante, de los bordes de uvas. Luego se determinó si los segmentos de recta trazados por cada dos puntos formaban un arco. Se regresó el círculo que mejor se ajustó a las posiciones de bordes previamente almacenadas [66] (Figura 3.20).

Sea V un arreglo de n posiciones muestreadas en los bordes de cada uva. Sea $L_i \in \{(V_i, V_{i+1})\}$ un segmento de recta definido por dos posiciones de V consecutivas. De tal manera que, si al menos tres segmentos L_i consecutivos están en la misma dirección y sus ángulos están entre los 6 y 60 grados, esta consecución será un arco. El ángulo θ se calcula con la siguiente ecuación:

$$\theta = \cos^{-1} \frac{\vec{v}_1 \cdot \vec{v}_2}{\|\vec{v}_1\| \|\vec{v}_2\|} \tag{3.25}$$

donde:

- $\vec{v}_1 =< L_2(x) - L_1(x), L_2(y) - L_1(y) >$,
- $\vec{v}_2 =< L_3(x) - L_2(x), L_3(y) - L_2(y) >$,
- $\cdot$ y $\|\vec{x}\|$ representan el producto punto y la magnitud, respectivamente.

Y la dirección se determina de la siguiente manera:

$$\text{Dirección} = \begin{cases} \text{Derecha}, & \text{si } \vec{v}_1 \times \vec{v}_2 \leq 0 \\ \text{Izquierda}, & \text{cualquier otro caso.} \end{cases} \tag{3.26}$$

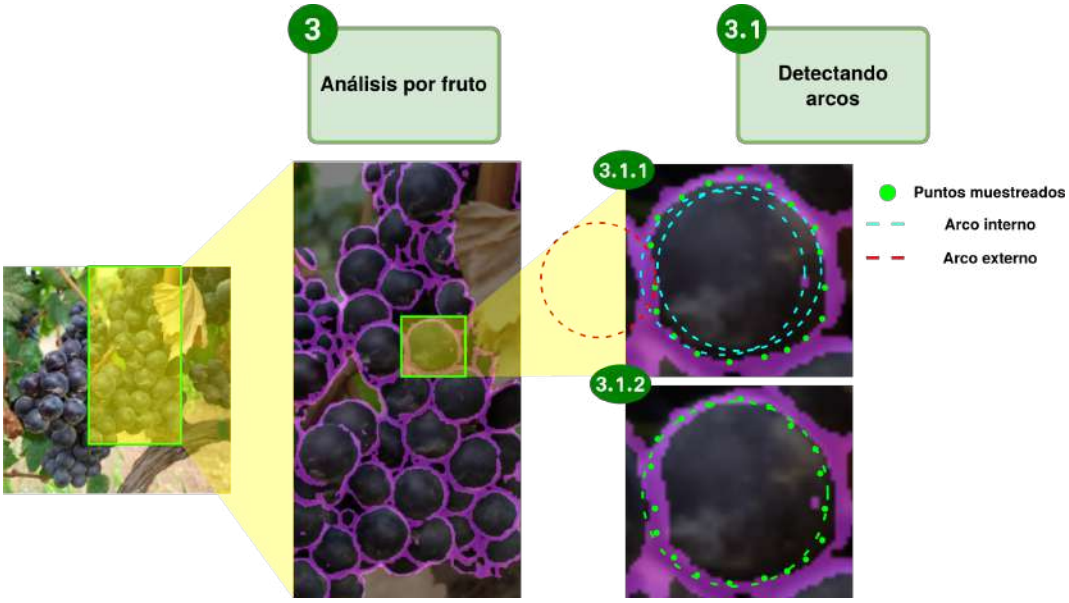


Figura 3.20: Detección de arcos. Esquema que ilustra cómo evaluar si un arco se ajusta a los bordes de uva en una imagen: (3) análisis de cada fruto y (3.1) proceso de detección de arcos para estimación de área ocluida.

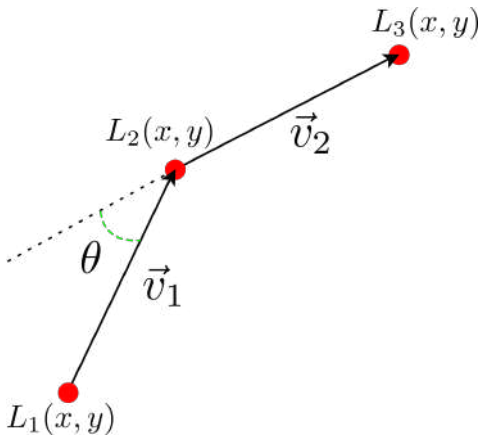


Figura 3.21: Cálculo de ángulo y dirección. Ilustración que corresponde al cálculo realizado en la ecuación 3.25.

donde $\times$ corresponde al producto cruz. Resultando en el arco formado por los puntos $V^* \subset V$, que cumple con las condiciones explicadas anteriormente (Figura 3.21).

Se tomó a V^* y se evaluó con el método *least-squares circle fit* [92], el cual consiste en devolver el círculo que mejor encaje en un conjunto de coordenadas dado. Además, se incluyó a este proceso dos etapas extra:

1. El círculo detectado ¿es interno o externo a la uva?. Ya que se conoce el área en píxeles de la uva detectada, si la intersección del área del círculo contenido en V^* es mayor al 25 % del área de la uva en revisión, este será un círculo interior, de lo contrario se descarta (Figura 3.20-2.1.2).

2. Si los centros y radios de los círculos detectados son cercanos, solo se conservará el círculo que mejor se ajuste en los bordes (Figura 3.20-2.1.3).

Para finalizar el capítulo, se describen las métricas utilizadas para evaluar el desempeño de las propuestas de esta investigación.

3.7. Métricas de evaluación

En esta sección se explica el proceso de cada métrica utilizada para hacer una evaluación cuantitativa.

Dado un clasificador y una instancia evaluada hay cuatro posibles resultados [93]. Si la instancia es positiva y es clasificada como positiva, entonces se considera como verdadero positivo (TP, *true positive*); si es clasificada como negativa, es considerada como falso negativo (FN, *false negative*). Si la instancia es negativa y es clasificada como negativa, es considerada como verdadero negativo (TN, *true negative*), de lo contrario, es un falso positivo (FP, *false positive*).

Para una evaluación cuantitativa, en este trabajo se utilizaron las siguientes métricas:

- Tasa de TP, también conocida como tasa de aciertos, incluyen al *Recall* o sensibilidad y *Precisión*, y se calculan de la siguiente manera:

$$\text{Sensibilidad} = \frac{TP}{TP + FN} \tag{3.27}$$

$$\text{Precisión} = \frac{TP}{TP + FP} \tag{3.28}$$

- Tasa de TN, también llamada: tasa de falsa alarma ó especificidad y se calcula de la siguiente manera:

$$\text{Especificidad} = \frac{TN}{TN + FP} \tag{3.29}$$

- Exactitud, es un porcentaje que permite evaluar el desempeño de todo el clasificador y es calculado de la siguiente manera:

$$\% \text{ exactitud} = \frac{TP + TN}{TP + TN + FP + FN} \tag{3.30}$$

- F1-Score es una media armónica entre la *Sensibilidad* y la *Precisión* y se calcula de la siguiente manera:

$$\text{F1-score} = \frac{2 \cdot \text{Precisión} \cdot \text{Sensibilidad}}{\text{Precisión} + \text{Sensibilidad}} \tag{3.31}$$

- El coeficiente de correlación de Matthew (MCC) se calcula de la siguiente manera:

$$\text{MCC} = \frac{(TP \cdot TN) - (FP \cdot FN)}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}} \tag{3.32}$$

- El error cuadrático medio (MSE por su sigla en inglés *mean squared error*) para evaluar el desempeño de conteo:

$$\text{MSE} = \frac{1}{n} \sum_{i=1}^{N} (\hat{C}_i - C_i)^2 \tag{3.33}$$

  donde:

  - $\hat{C}_i = \sum_j G(\hat{p_j}, P)$
    - $\hat{p_j}$ corresponde al centro del grupo de píxeles segmentados como "uva",
    - P a todo los centros anotados manualmente, y
    -

$$G(p_j, P) = \begin{cases} 1, & \text{si la distancia euclidiana entre } p_j \text{ y un} \\ & P_k \text{ es menor que } \mu_r + \sigma_r. \\ 0, & \text{cualquier otro caso.} \end{cases} \tag{3.34}$$

    - C_i corresponde al conteo manual.
  - $\hat{C}_i$ y C_i representan la estimación y el conteo manual, respectivamente.

- Error absoluto medio (MAE por su sigla en inglés *mean abosolute error*) para evaluar la predicción del peso, y se define para:

$$\text{MAE} = \frac{1}{n} \sum_{i=1}^{N} |\hat{P}_i - P_i| \tag{3.35}$$

  donde $\hat{P}_i$ y P_i corresponden al peso estimado y real, respectivamente.

Además para evaluar la segmentación no supervisada se usó la métrica de información mutua, y es definida por la siguiente ecuación:

$$MI(U,V) = \sum_{i=1}^{|U|}\sum_{j=1}^{|V|} \frac{|U_i \cap V_j|}{N} \log \frac{N|U_i \cap V_j|}{|U_i||V_j|} \tag{3.36}$$

donde U y V son las imágenes de referencia y la estimación, respectivamente. Las operaciones $|*|$ y $\cap$ corresponden a la cantidad de elementos y la intersección, respectivamente; mientras que N es la cantidad de elementos en total a evaluar.

3.7.1. Validación cruzada

Este método se utilizó para evaluar los resultados de un clasificador y garantizar que las particiones entre los datos de entrenamiento y validación, son independientes. De manera general, la estrategia de validación cruzada (CV, por su sigla en inglés, *cross-validation*) consiste en dividir el conjunto de datos disponible, en dos conjuntos: entrenamiento y validación y promediar múltiples métricas de calidad, *e.g.* tasa de precisión [94].

Validación cruzada con K dobleces

La validación cruzada (CV) consiste en dividir los datos de muestra en K subconjuntos. Usando cada subconjunto k_i como conjunto de validación y el resto como conjunto de entrenamiento (Figura 3.22). Este proceso se repite K iteraciones, en cada subconjunto de validación, al final se promedia el estimador de calidad para cada subconjunto k_i, para obtener un único resultado en cada subconjunto. El ordenamiento de cada uno puede ser fijo, es decir, manteniendo el orden original de los datos, o de manera aleatoria.

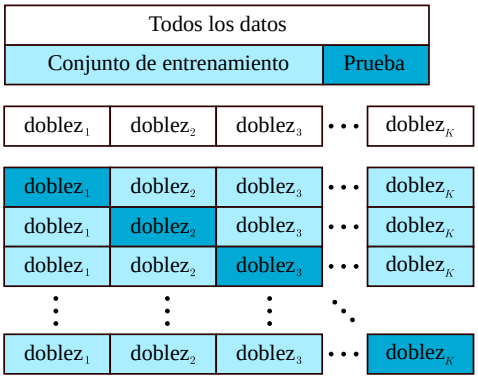


Figura 3.22: Validación cruzada. Proceso general de las particiones que se hacen en cada iteración [95].

CV dejando uno fuera

Este tipo de CV consisten en entrenar el clasificador con todo el conjunto, dejando una sola observación fuera para dato de prueba. El estimador de calidad usado aquí es el error. Como se puede deducir, el número de iteraciones es $K = N$, donde N es el número total de observaciones, lo que lo hace computacionalmente muy costoso.

Dentro de este proceso se puede incluir el proceso sintonización de los parámetros de un clasificador, el cual consiste en hacer un barrido con distintos valores en cada parámetro para determinar, los valores óptimos con los que se obtiene el desempeño más alto.

La detección de frutos individuales presenta un reto particular en imágenes de racimos de uvas, por la forma natural de aglomeración dentro de los racimos. En el siguiente capítulo se muestran los resultados de las propuestas presentadas anteriormente.

Capítulo 4

Resultados y discusión

Este capítulo inicia con la evaluación del vector de características de color, propuesto en esta investigación. Se evaluó también, la detección de círculos en el dominio de imágenes de racimos de uva, y ademas, se compararon cinco clasificadores de ML y tres arquitecturas de DL, a fin de determinar cuál es el enfoque que presenta el mejor desempeño al abordar la problemática central de esta investigación. Este capítulo cierra con un experimento con el que se determinó la relación lineal entre el número de uvas estimadas y el peso real de racimos individuales.

4.1. Implementaciones y repositorios

El lenguaje de programación usado en este trabajo para las implementaciones fue Python (versión 3.6.9). Para los algoritmos de espacios e índices de color, y aprendizaje no supervisado (segmentación por color), se generaron implementaciones propias y los códigos se encuentran disponibles en el siguiente repositorio: *Unsupervised learning*[1]. Para la implementación de aprendizaje supervisado se usó el paquete Scikit Learn; mientras que para el aprendizaje profundo se usó el *framework* Pytorch y los códigos se encuentran en el siguiente repositorio: *Grape learning*[2].

Para ejecutar las implementaciones se utilizaron dos estaciones de trabajo:

1. Dell precision: Procesador Intel Xeon E5-1620 v3 3.5 GHz × 8, con 24 Gb de RAM y una GPU Nvidia Quadro K2200 de 4Gb.

2. Thinkstation: Procesador Intel Xeon w-2104 3.2 GHz × 4, con 15 Gb de RAM y una GPU Nvidia Quadro P1000 de 4 Gb.

La estación 1 se usó para los entrenamientos de aprendizaje profundo, mientras que la estación 2 para las pruebas y el resto de los procesos. Para dar inicio, con los resultados

[1] https://github.com/ricglez90/unsupervised_segmentation
[2] https://github.com/ricglez90/grape_learning

de esta investigación se analizó el desempeño de los algoritmos de agrupamiento usando el color como característica principal.

4.2. El color como característica principal

4.2.1. Índices de color

En esta sección, se llevó a cabo un análisis para determinar cuál índice de color tiene mejor desempeño en la tarea de segmentar racimos dentro de una imagen.

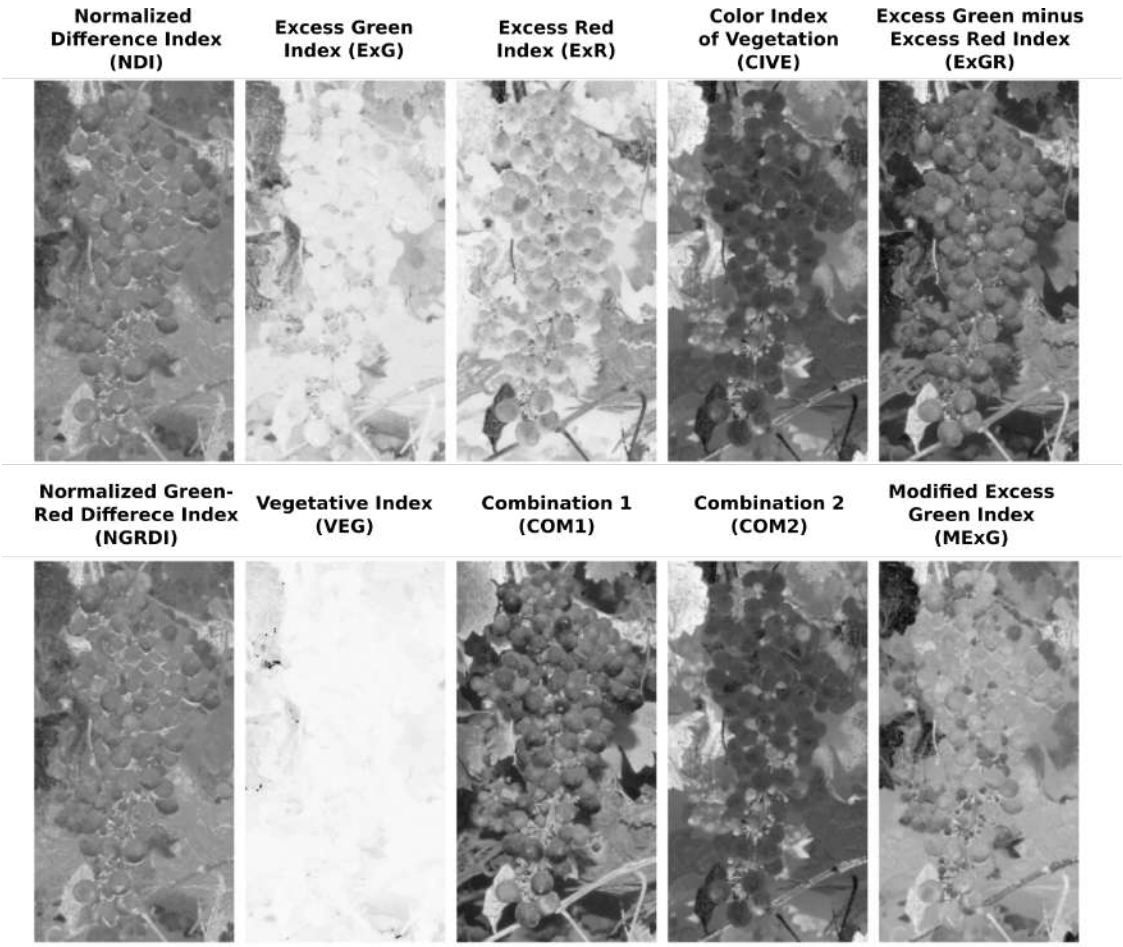


Figura 4.1: Análisis al calcular cada índice de color utilizado en este trabajo (Apéndice A.1.3). El nombre de cada índice se muestra en la parte superior de cada imagen.

Primero, se transformó la imagen de entrada en cada índice de color (Figura 4.1);

luego se calculó un valor de umbral óptimo usando el método Otsu[3]; por último, se hizo una segmentación de la imagen en dos regiones: "fondo" y "racimos" (Apéndice A.2.1).

Para evaluar el desempeño, se comparó la imagen de referencia contra la imagen segmentada, a través de la métrica información mutua (ecuación 3.36). Esta métrica está en el intervalo dinámico $[0, 1]$, donde un valor cercano a 1 corresponde a una segmentación buena, y por el contrario, un valor cercano a 0 es una segmentación mala.

En esta sección se presentan los resultados en forma de gráficos de cajas y bigotes, donde cada caja tiene la siguiente estructura: los bordes de la caja, superior e inferior, corresponden al primer y tercer cuartil, respectivamente; los bordes de los "bigotes" corresponden a 1.5 de la distancia inter-cuartil; estos se indican al final de cada línea vertical proveniente de los bordes superior e inferior de la caja. La línea que divide a la caja corresponde a la mediana de la muestra y los círculos por encima o debajo de los bigotes, corresponden a valores atípicos (*outliers*). Un gráfico de este tipo tiene como objetivo resumir la distribución de un conjunto de muestras. Para esta evaluación, los resultados corresponden a evaluar cada conjunto de datos. Además, para comparar cada caja y determinar si existe o no una diferencia estadística significativa se usó la prueba no-paramétrica de Wilcoxon [96]. Esta se basa en las medianas de la muestra. Estas pruebas son ampliamente utilizadas ya que son robustas cuando los datos no provienen de una distribución normal (Gaussiana). La muestra que presenta diferencia estadística significativa se observará en cada gráfico señalada con un "*", mientras que las muestras donde no hay diferencia, se indicará con una flecha roja.

En los resultados de este experimento se observó que el desempeño está debajo de 0.5 (información mutua), indicando que la segmentación por umbrales no es adecuada para la tarea de separar racimos del fondo (Figura 4.2). Se observó también, que los índices ExG, ExR y ExGR son los que presentaron el desempeño más alto en el conjunto RondoDS, mientras que ExG y CIVE fueron los más altos para el conjunto CosmeDS.

La segmentación por un valor de umbral y los índices de color, no separan correctamente a los frutos del fondo (Figura 4.3). Los índices de color habían sido utilizados, para separar el suelo del follaje en imágenes de cultivos. Se observó que para el conjunto RondoDS, las zonas que corresponden al follaje fueron separadas correctamente (Figura 4.3-RondoDS), mientras que en el conjunto CosmeDS esto no sucedió (Figura 4.3-CosmeDS). Ya que los resultados fueron pobres, se exploraron algoritmos de agrupamiento, a los cuales se les da de entrada características de color extraídas de las imágenes crudas.

4.2.2. Espacios de color

En esta sección, se evaluó el desempeño de algoritmos de agrupamiento usando espacios de color para separar racimos del fondo dentro de una imagen. Estos espacios

[3]Método utilizado para calcular un umbral óptimo en imágenes en escala de grises [18].

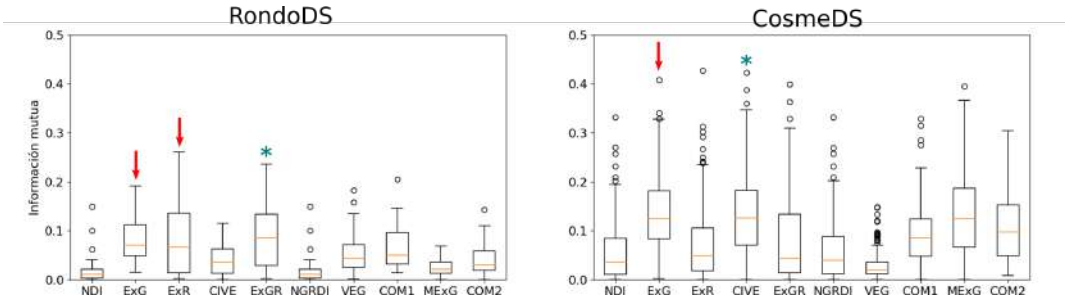


Figura 4.2: Segmentación por umbral usando índices de color. Los gráficos de cajas corresponden a los conjuntos RondoDS (izquierda) y CosmeDS (derecha). Cada caja representa un índice de color y la métrica utilizada fue información mutua. El asterisco azul indica la caja que se comparó con el resto y que presentó diferencia estadística, a través de la prueba de Wilcoxon. Las muestras que no presentaron diferencia están señaladas por la flecha roja.

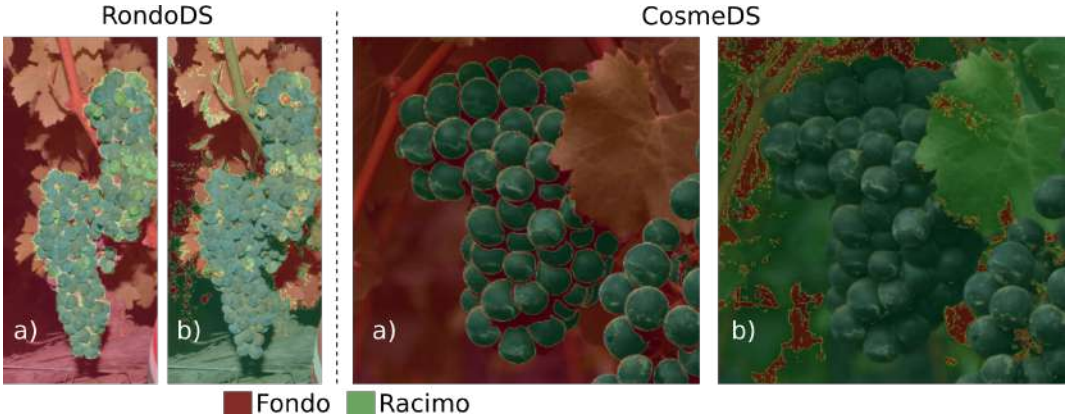


Figura 4.3: Segmentación por umbral usando índices de color. Las imágenes mostradas corresponden a los conjuntos RondoDS (izquierda) y CosmeDS (derecha). Las imágenes (a) corresponden a la imagen de referencia sobre-puesta a la original; mientras que la (b) al resultado de Otsu sobre los índices ExGR y CIVE, para los conjuntos RondoDS y CosmeDS, respectivamente.

separan el color en dos componentes principales: cromaticidad e intensidad (Figura 4.4).

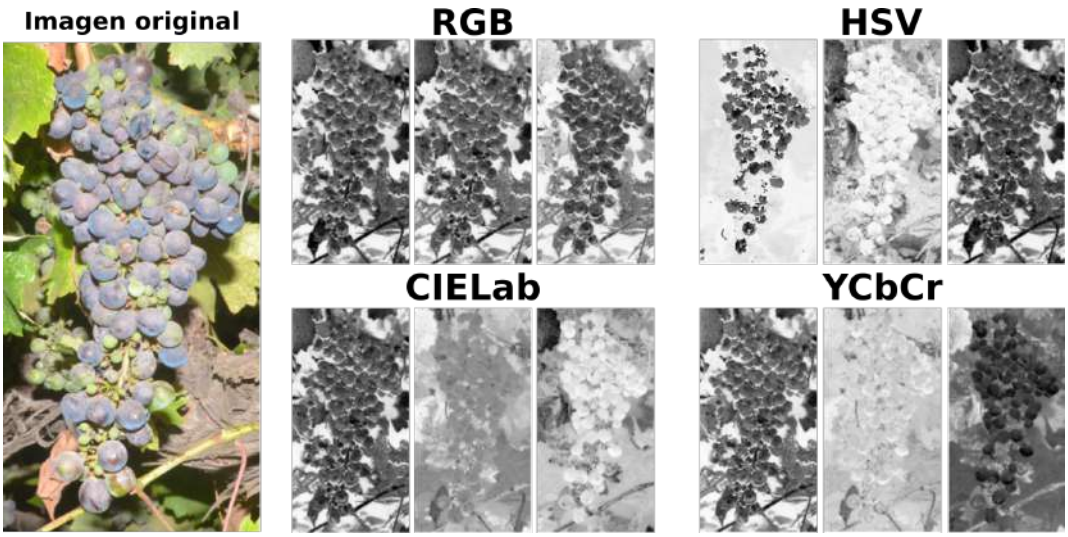


Figura 4.4: Análisis al calcular cada espacio de color utilizado en este trabajo (ver Apéndice A.1.2). El nombre de cada índice se muestra en la parte superior de cada imagen.

Los algoritmos de agrupamiento utilizados fueron: K-medias y FC-medias (Apéndice A.2.1), y las características de color correspondieron a los componentes de cromaticidad de los espacios: HSV, CIELab y YCbCr. El experimento consistió en catalogar los píxeles en dos grupos: "fondo" y "racimo" ($k = 2$). Los resultados de este experimento también se presentaron en forma de gráficos de cajas y bigotes (Figura 4.5).

Nótese que los resultados en ambos conjuntos siguen debajo del 0.5. Para RondoDS la combinación espacio HSV y el algoritmo K-medias presentó el mejor desempeño; mientras que para CosmeDS la combinación de HSV y FC-medias proporcionaron el mejor desempeño. Al usar componentes de espacios de color, se esperaría una mejora en el proceso de segmentación, y aunque no se observa una mejora en el desempeño, si se comparan las figuras 4.2 y 4.6 en la muestra de CosmeDS, la hoja es ahora identificada como fondo al incluir componentes de color y algoritmos de agrupamiento.

Como se mencionó en la sección 3.3.1, una propuesta formulada en esta investigación, consistió en combinar los índices y espacios de color en un solo vector de características y a continuación se muestran los resultados usando un enfoque de segmentación no supervisada.

4.2.3. Combinando índices y espacios de color

Este experimento consistió en crear una imagen de 22 valores por cada píxel: 12 aportadas por los espacios de color y 10 por los índices. Esta imagen se dio como

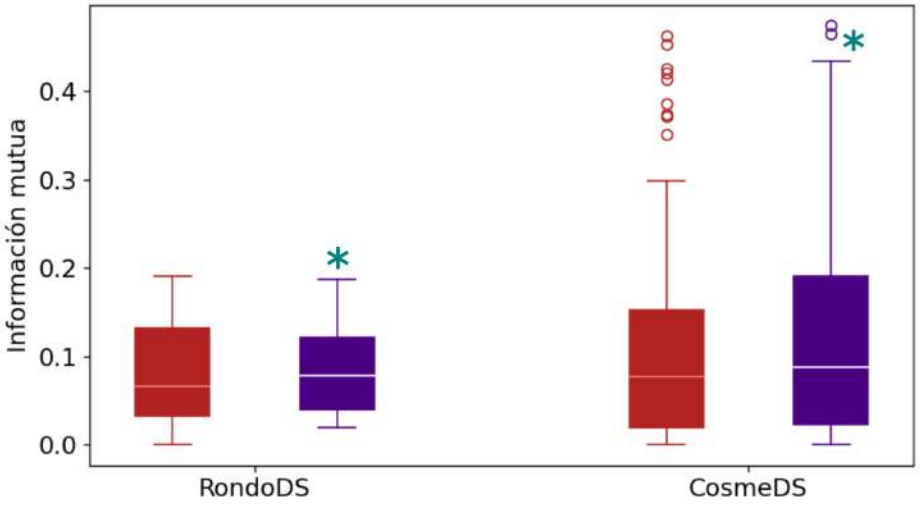


Figura 4.5: Segmentación no supervisada. Los gráficos de cajas corresponden a los conjuntos RondoDS (izquierda) y CosmeDS (derecha). Cada caja representa un espacio de color y la métrica utiliza fue información mutua; las cajas de color rojo corresponden al algoritmo K-medias, mientras que las moradas al FC-medias. El asterisco azul indica la caja que se comparó con el resto y que presentó diferencia estadística, a través de la prueba de Wilcoxon.

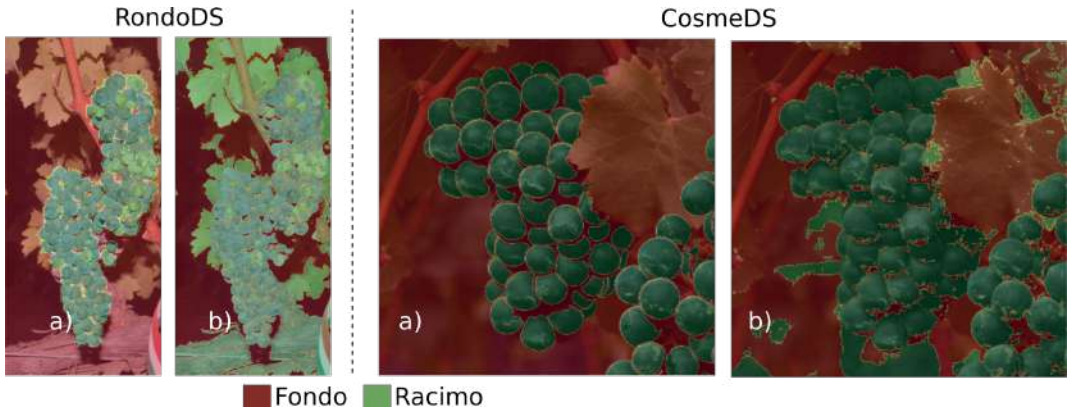


Figura 4.6: Muestra de segmentación no supervisada. Las imágenes mostradas corresponden a los conjuntos RondoDS (izquierda) y CosmeDS (derecha). Las imágenes (a) corresponden a la imagen de referencia sobrepuesta a la original; mientras que la (b) al resultado de segmentar usando K-medias (RondoDS) y FC-medias (CosmeDS) sobre el espacio HSV.

entrada a los algoritmos de agrupamiento utilizados en la sección anterior (K-medias y FC-medias). De nueva cuenta se evaluó el desempeño sobre los conjuntos de RondoDS y CosmeDS (Figura 4.7).

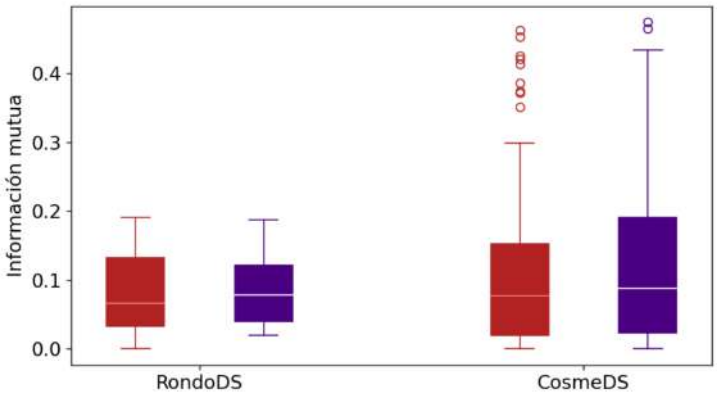


Figura 4.7: Segmentación no supervisada. Los gráficos de cajas corresponden a los conjuntos RondoDS (izquierda) y CosmeDS (derecha). Cada caja representa los resultados de usar el vector de características de 22 elementos y la métrica utiliza fue información mutua; las cajas de color rojo corresponden al algoritmo K-medias, mientras que las moradas al FC-medias. Se hizo una prueba estadística a través de la prueba de Wilcoxon, pero no se encuentra diferencia significativa.

Al evaluar los resultados usando la métrica "información mutua" no se observó una mejoría significativa, en comparación con solo usar espacios de color. Cabe mencionar que medir el desempeño de los algoritmos de agrupamiento no es una tarea fácil, ya que esta métrica no compara píxel a píxel los resultados, si no que evaluá dependencia estadística entre ambas imágenes. Además, el aprendizaje no supervisado aprende de la imagen original y no de la imagen de referencia. Se pudo percibir una mejoría visual en la segmentación sobre el conjunto de CosmeDS (Figura 4.8).

Para determinar si la combinación de índices y espacios de color, presentan un buen desempeño al segmentar racimos en imágenes, se propuso la inclusión del aprendizaje supervisado.

4.3. Localizando racimos con aprendizaje supervisado

El objetivo en este experimento fue localizar racimos en imágenes digitales. Esto se hizo apoyado de índices y espacios de color, formando un vector de 22 características y

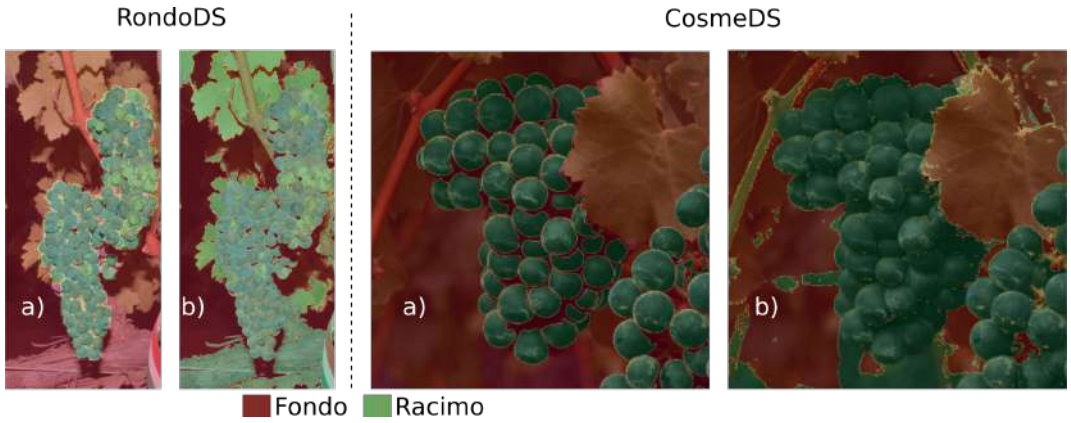


Figura 4.8: Muestra de segmentación no supervisada. Las imágenes mostradas corresponden a los conjuntos RondoDS (izquierda) y CosmeDS (derecha). Las imágenes (a) corresponden a la imagen de referencia sobre-puesta a la original; mientras que la (b) al resultado de segmentar usando FC-medias (CosmeDS) sobre el vector de características de 22 elementos.

entrenando tres clasificadores con él, a fin de estimar si cada píxel era "uva" o "fondo" [97].

En esta sección se evaluó el desempeño del aprendizaje supervisado al comparar cinco clasificadores (Apéndice A.2.2): vecinos cercanos (KNN), máxima verosimilitud (MaL), regresión logística multinomial (MLR), perceptrón multicapa (MLP) y bosque aleatorio (RF). Cada evaluación se hizo sobre los dos conjuntos de datos desarrollados en esta investigación: RondoDS y CosmeDS.

4.3.1. RondoDS

De este conjunto se tomaron 20 imágenes para entrenamiento y 10 para validación. De la partición de entrenamiento se tomó el 5 % de los píxeles para formar un conjunto de tamaño 1,544,888×22, que contienen en igual proporción las clases: "fondo" y "racimo"; se escogió un porcentaje arbitrario de entrenamiento, debido a la memoria disponible, pero este valor, se puede incrementar si el problema lo requiere o si se cuenta con mayor memoria disponible.

Como primer experimento, se evaluó el desempeño de los cinco clasificadores mencionados anteriormente. Se utilizó la métrica de exactitud (sección 3.7), y a través del procedimiento de validación cruzada con 10 dobleces, se llevo a cabo una sintonización de parámetros con múltiples valores de entrada. La Tabla 4.1 muestra la exactitud promedio de cada clasificador junto con su desviación estándar bajo el conjunto de parámetros que arrojaron los mejores resultados.

Los clasificadores fueron entrenados con el conjunto de parámetros que arrojaron el

Tabla 4.1: Validación cruzada sobre RondoDS. Como conjunto de entrenamiento se usó el 5 % de los píxeles de RondoDS y cada muestra corresponde al vector de 22 componentes de color; en cada clasificador se muestra la exactitud promedio ± su desviación estándar en el conjunto de parámetros que mejor se desempeñó.

Nombre	**Exactitud**
K vecinos cercanos (KNN)	0.83 ± 0.03
Máxima verosimilitud (MaL)	0.79 ± 0.04
Regresión logística Multinomial (MLR)	0.79 ± 0.03
Perceptrón multicapa (MLP)	0.82 ± 0.3
Bosque aleatorio (RF)	0.83 ± 0.02

mejor desempeño para cada uno. Después, se evaluaron usando la métrica de F1-score en cada clase para observar el desempeño de cada uno. Se usó esta métrica ya que es la que presenta mayor robustez frente a clases sin balance, y se encuentra en el intervalo dinámico de $[0, 1]$; donde un valor cercano a 1 se interpreta como una segmentación buena y cercano a 0 como una mala. Se repitió el mismo formato de presentación de resultados en gráficos de cajas y bigotes (Figura 4.9).

El clasificador bosque aleatorio (RF, *random forest*) alcanzó el mejor desempeño en ambas categorías aprendidas (fondo y uva, Figura 4.9). Este clasificador presentó diferencia estadística en comparación a los otros clasificadores. La Figura 4.10 presenta una muestra de RondoDS de la segmentación de cada algoritmo evaluado.

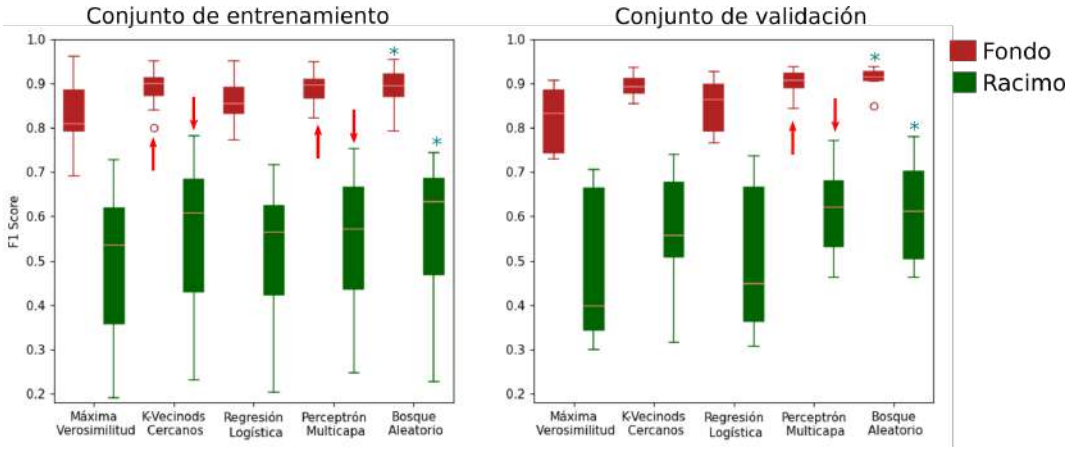


Figura 4.9: Segmentación supervisada sobre RondoDS. Cada clasificador se muestra sobre el eje horizontal y el eje vertical corresponde al F1-score sobre las clases aprendidas: fondo (rojas) y uva (verde). Las cajas con * corresponden a las que presentaron diferencia estadística significativa con intervalo de confianza a 95 %, a través de una prueba de Wilcoxon.

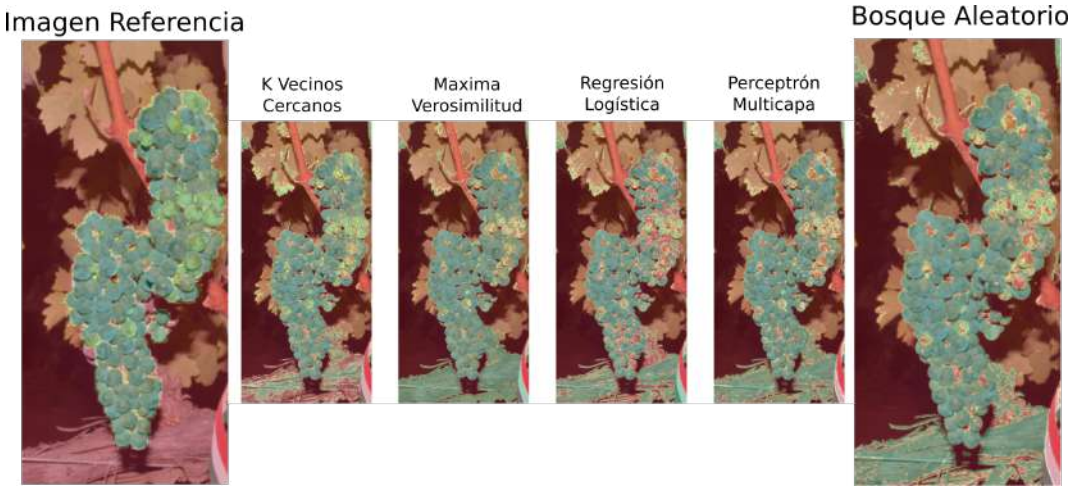


Figura 4.10: Segmentación supervisada sobre RondoDS. Imagen de entrada con el resultado de cada clasificador sobrepuesto; clases fondo y uva corresponden a los colores rojo y verde, respectivamente.

El siguiente experimento consistió en aplicar tres operadores morfológicos: apertura, cierre y ambos. Estos agregan relación espacial al enfoque de clasificación píxel-a-píxel. Esta operación se hizo sobre la imagen resultado del algoritmo bosque aleatorio. Se observó diferencia estadística al aplicar ambos operadores con respecto al no aplicarlos (Figura 4.11). Sin embargo, no se encontró diferencia visual entre los operadores (Figura 4.12).

Ahora se repitió este experimento completo pero sobre el conjunto de CosmeDS como datos de entrada.

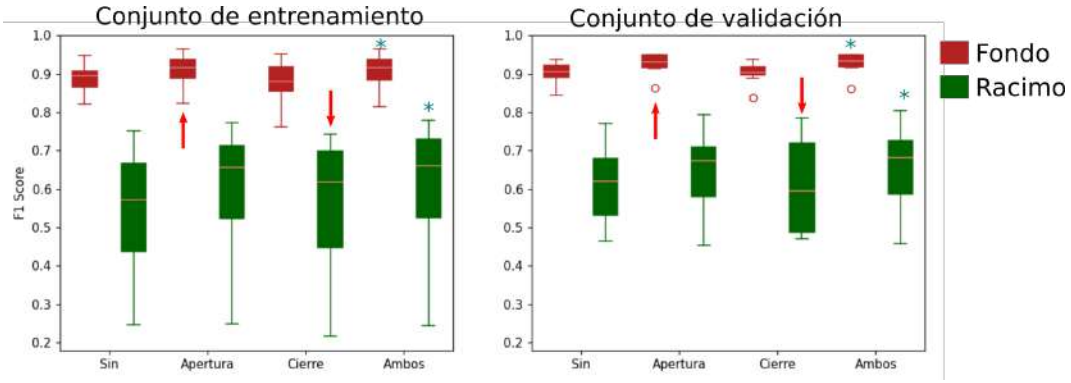


Figura 4.11: Operadores de morfología. Cada operador se muestra sobre el eje horizontal y el eje vertical corresponde al F1-score sobre las clases aprendidas: fondo (rojas) y uva (verde). Las cajas con * corresponden a las que presentaron diferencia estadística significativa con intervalo de confianza a 95 %, a través de una prueba de Wilcoxon, a excepción de las señaladas (flecha roja).

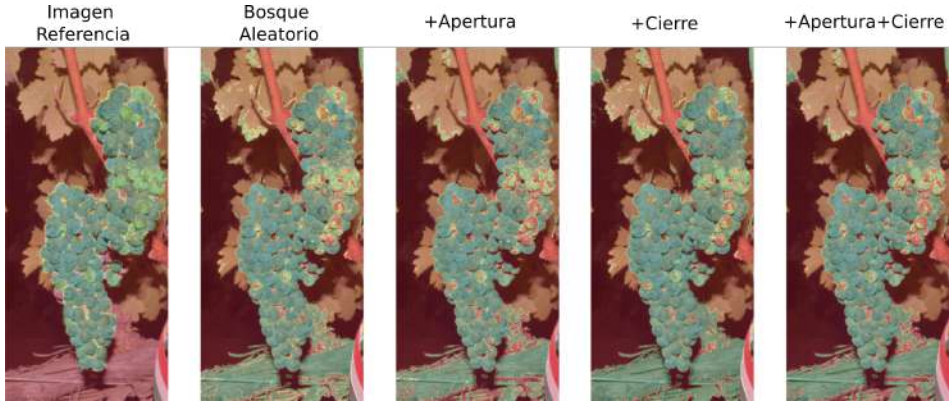


Figura 4.12: Operadores de morfología sobre RondoDS. Imagen de entrada con el resultado de cada operador sobrepuesto; clases fondo y uva corresponden a los colores rojo y verde, respectivamente.

4.3.2. CosmeDS

Para este experimento se tomaron del conjunto de CosmeDS 162 imágenes para entrenamiento y 40 para validación. De las 162 imágenes, se tomó el 5 % de los píxeles y se formó un conjunto de tamaño 2,724,840×22 características de color. Este contiene en igual proporción las clases: "fondo" y "racimos". Para evaluar el desempeño de cada clasificador se hizo un proceso de sintonización de parámetros y en cada configuración se llevó a cabo validación cruzada de 10 dobleces. La Tabla 4.2 muestra los resultados promedios de cada clasificador de los 10 dobleces en el mejor conjunto de parámetros. De nueva cuenta, el clasificador RF es el que mejor desempeño presentó.

Tabla 4.2: Validación cruzada de 10 pliegues sobre CosmeDS. Como conjunto de entrenamiento se usó el 5 % de los píxeles de CosmeDS y cada muestra corresponde a vector de 22 componentes de color; en cada clasificador se muestra la exactitud promedio *pm* su desviación estándar en el conjunto de parámetros que mejor se desempeñó.

Nombre	**Exactitud**
K vecinos cercanos (KNN)	0.85 ± 0.02
Máxima verosimilitud (MaL)	0.78 ± 0.06
Regresión logística Multinomial (MLR)	0.81 ± 0.03
Perceptrón multicapa (MLP)	0.88 ± 0.03
Bosque aleatorio (RF)	0.89 ± 0.03

El modelo RF presentó diferencia estadística significativa en ambas clases con respecto al resto de los clasificadores (Figura 4.13) y la Figura 4.14 presenta una muestra del resultado de segmentación de cada clasificador.

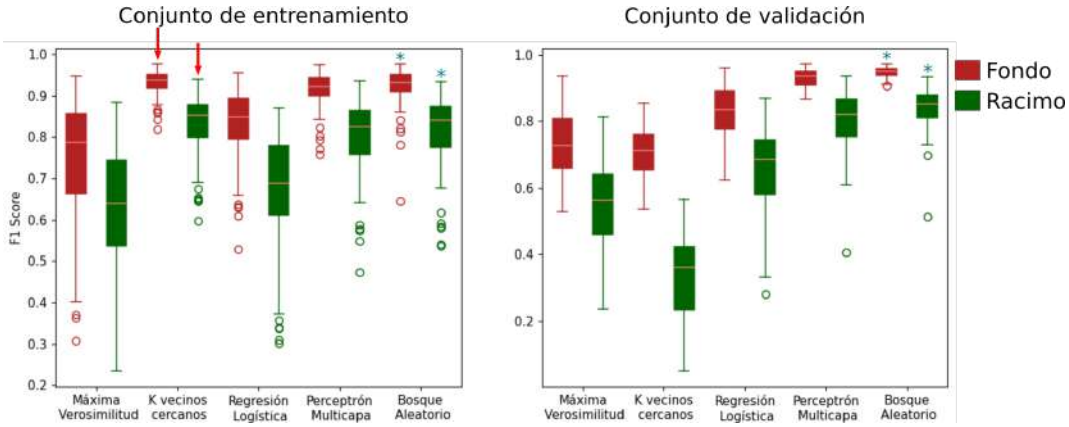


Figura 4.13: Segmentación supervisada sobre CosmeDS. Cada clasificador se muestra sobre el eje horizontal y el eje vertical corresponde al F1-score sobre las clases aprendidas: fondo (rojas) y uva (verde). Las cajas con * corresponden a las que presentaron diferencia estadística significativa con intervalo de confianza a 95 %, a través de una prueba de Wilcoxon.

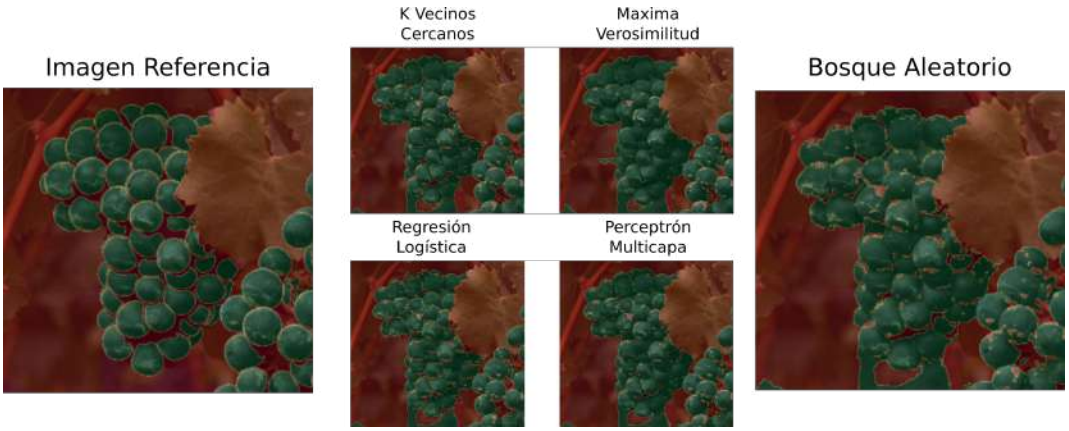


Figura 4.14: Segmentación supervisada sobre CosmeDS. Imagen de entrada con el resultado de cada clasificador sobrepuesto; clases fondo y uva corresponden a los colores rojo y verde, respectivamente.

De nueva cuenta, se aplicaron los tres operadores morfológicos: apertura, cierre y ambos. Al igual que en los resultados con el conjunto RondoDS, el uso de ambos operadores morfológicos presentó diferencia estadística significativa en comparación a no usarlos (Figura 4.15). Una muestra resultado de cada operador se pueden observar en la Figura 4.16.

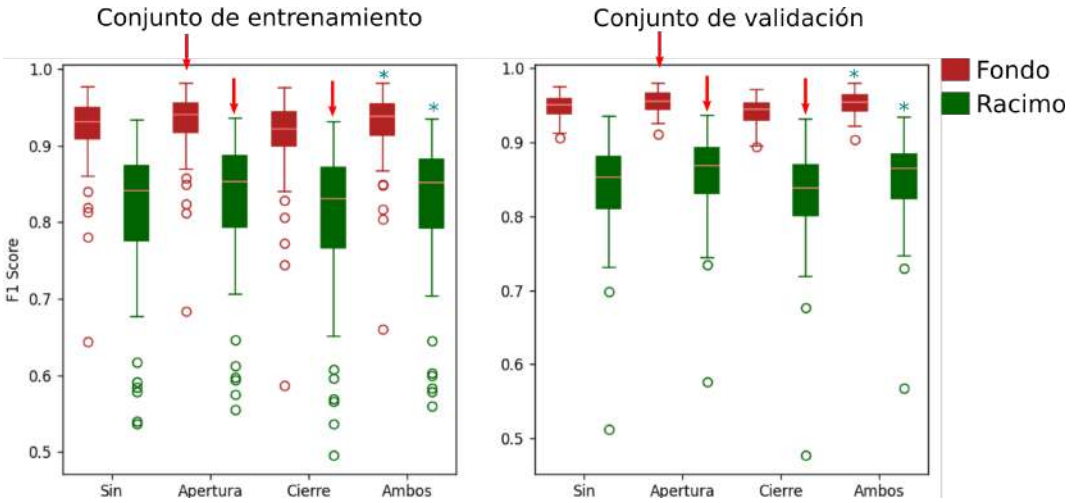


Figura 4.15: Operadores de morfología. Cada operador se muestra sobre el eje horizontal y el eje vertical corresponde al F1-score sobre las clases aprendidas: fondo (rojas) y uva (verde). Las cajas con * corresponden a las que presentaron diferencia estadística significativa con intervalo de confianza a 95 %, a través de una prueba de Wilcoxon, a excepción de las señaladas (flecha roja).

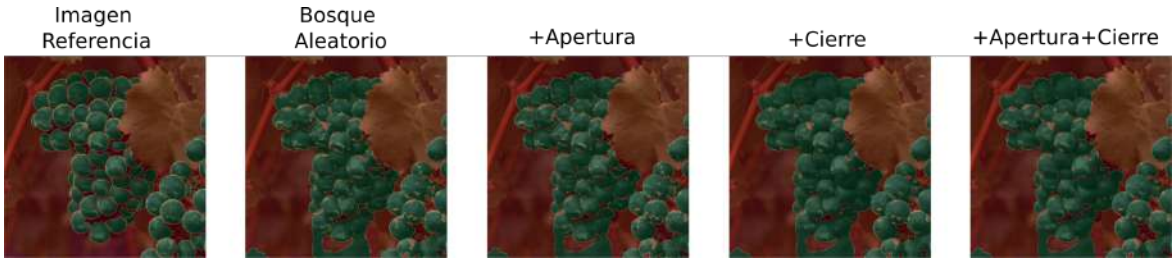


Figura 4.16: Operadores de morfología sobre CosmeDS. Imagen de entrada con el resultado de cada operador sobre puesto; clases fondo y uva corresponden a los colores rojo y verde, respectivamente.

Abdelghafour et al. [42] presentaron también un enfoque píxel-a-píxel, para separar en cuatro grupos de interés: “uvas”, “centro de la hoja”, “tallos” y “borde de la hoja”. Ellos obtuvieron una sensibilidad y precisión de 0.85 y 0.95, respectivamente, resultando en un F1-score de **0.89**. Nótese que el resultado con el mejor desempeño del algoritmo

bosque aleatorio, presentado previamente, alcanzó un F1-score 0.95 y 0.88, para las clases de "fondo" y "racimo", respectivamente; resultando en un F1-score promedio de **0.91**. No se pudo realizar una comparación directa con los resultados de Abdelghafour et al. [42] ya que no se tuvo acceso a sus datos.

Para validar el modelo RF se probó sobre el conjunto de datos BIVcolor [45]. Este es un conjunto externo y no contiene imágenes de referencia, por lo que no es posible hacer una evaluación cuantitativa.

4.3.3. Validación en BIVcolor

Para esta validación se tomó al modelo con el mejor desempeño del experimento anterior (RF, Bosque aleatorio) entrenado con los datos de CosmeDS junto con los operadores morfológicos. Se siguió el mismo procedimiento de la sección anterior, separando la imágenes del conjunto BIVcolor [45] en dos regiones de interés: "fondo" y "racimo". Este conjunto no cuenta con etiquetas adecuadas para realizar una evaluación cuantitativa, y por lo tanto, solo se hizo una inspección visual de los resultados (Figura 4.17).

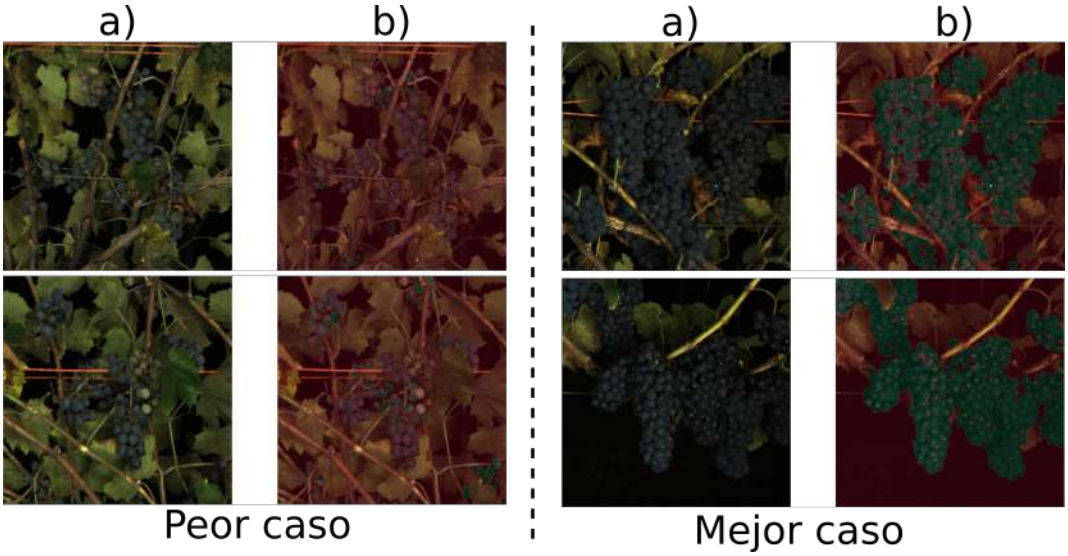


Figura 4.17: Segmentación supervisada sobre BIVcolor [45]. La imagen de entrada corresponden a las columnas (a) mientras que a los resultados sobrepuestos (b); las columnas (b) corresponden a las imágenes de entrada con las regiones de interés sobrepuestas, que el modelo bosque aleatorio localizó; clases "fondo" y "racimo" corresponden a los colores rojo y verde, respectivamente.

Al probar el modelo RF sobre BIVcolor y encontrando el porcentaje de píxeles catalogados como "racimo", se obtuvieron resultados con un desempeño alto (Figura

4.17-Mejor caso) y con desempeño bajo (Figura 4.17-Peor caso). Se observó que en los casos con bajo desempeño, el color presente de los racimos era distinto al color de los racimos usados para el entrenamiento.

El algoritmo propuesto tiene un paso posterior y corresponde a filtrar falsos positivos y señalar la posición de los grupos de píxeles clasificados como racimos (Figura 4.18). Se puede observar que las cajas verdes señalan la posición y el tamaño de los racimos localizados dentro la imagen.

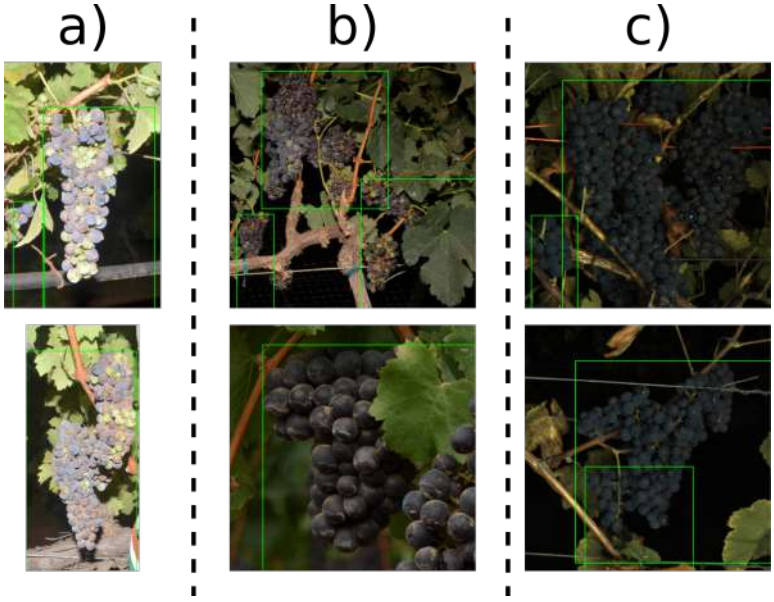


Figura 4.18: Localización de racimos dentro de imágenes. La imagen (a) corresponde a una muestra sobre RondoDS, la (b) sobre CosmeDS, mientras que la (c) a BIVcolor [45]; las cajas de color verde corresponden a las zonas donde se localizaron los grupos de píxeles catalogados como "racimo".

Los algoritmos de ML entrenados hasta este punto, son capaces de localizar regiones correspondientes a racimos dentro de la imagen. Sin embargo, esto no resuelve el problema de localizar frutos individuales.

4.4. Algoritmo para la localización de frutos

El siguiente experimento consistió en tomar las imágenes segmentadas con el modelo bosque aleatorio (RF) y calcular el área clasificada como "racimos", y a su vez, estimar el número de frutos de cada área calculada con el método propuesto por Millan et al. [13]. Este método requiere dos parámetros de entrada: el valor del área catalogada como "racimo" y el valor del área circular promedio de las uvas en el racimo evaluado. Por un lado. este método es sensible a falsos positivos, ya que píxeles mal clasificados afectarían

en el número de frutos estimados. Por otro lado, el segundo parámetro es complicado de calcular, ya que conocer de antemano el área del círculo a encajar en la realidad no es posible.

Los resultados se presentan en forma de un gráfico-R, el cual consiste en un gráfico de dispersión donde el eje x corresponde al conteo manual de uvas visibles en la imagen, mientras que el eje y a la estimación del número de frutos. Además, se incluyó una línea verde que corresponde al conteo ideal, junto con dos líneas rojas (arriba y abajo) que corresponden a un valor de sesgo permitido de conteo. Este tipo de gráficos tienen como objetivo determinar el desempeño de conteo de los métodos en cuestión.

Se evaluó este método solo en los dos conjuntos desarrollados en esta investigación (RondoDS y CosmeDS) ya que solo ellos incluyen el área promedio en píxeles de frutos individuales. Se observó que en ambos conjuntos se obtuvo un sobre-conteo y por lo tanto un desempeño de conteo bajo (Figura 4.19).

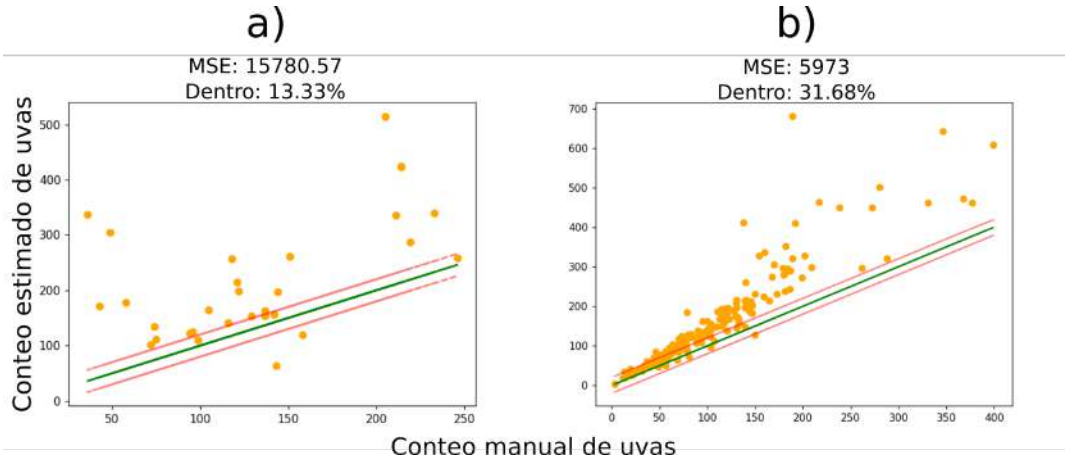


Figura 4.19: Estimación de frutos basado en el método Millan et al. [13]. La imagen (a) presenta los resultados sobre **RondoDS**, mientras que la (b) sobre **CosmeDS**. En ambas el eje horizontal corresponde al conteo manual de frutos individuales visibles en la imagen, mientras el vertical el conteo estimado de frutos. La línea verde representa el conteo perfecto ($R^2 = 1$), mientras que la línea roja al valor permitido para este conteo ideal. El MSE y el porcentaje que cae cerca al conteo ideal son incluidos arriba de cada gráfico.

Millan et al. [13] reportaron una $R^2 = 0.8$ con un error de conteo de 20 frutos individuales para este enfoque de estimación de frutos. Estos resultados quieren decir que el desempeño de conteo se ajusta a una recta en un 80 % y se desvía de ella en 20 unidades (frutos) promedio. Mientras que los resultados obtenidos en esta investigación arrojaron un desempeño de conteo del 13.33 % y 31.68 %, para los conjuntos RondoDS y CosmeDS, respectivamente (Figura 4.19). Nótese que este método presentó un sobre-conteo en ambos conjuntos y se debe a que este método es sensible píxeles

mal categorizados. En el trabajo de Millan et al. [13] estimaron el número de frutos en imágenes que fueron capturadas con un fondo de color naranja con la intención de separar al racimo individual del fondo. Sin embargo, a pesar de las condiciones controladas de captura, solo se alcanzó un 80 % de desempeño en el conteo. Por esta razón, se decidió no utilizar este enfoque de estimación y se propuso usar un detector de círculos para aprovechar la forma geométrica de las uvas.

4.4.1. Localización de círculos

El primer experimento consistió en entrenar a seis algoritmos de clasificación con imágenes de uvas individuales tomadas del conjunto RondoDS. El segundo consistió en detectar círculos en imágenes de racimos de uvas, y usar el clasificador que presentó el mejor desempeño del experimento anterior.

Preparando al clasificador

Se extrajeron del conjunto RondoDS 7816 sub-imágenes de 40×40 píxeles de resolución que contuvieran uvas individuales. Estas corresponden a la anotación manual de cada fruto individual y contienen tanto uvas completas como parcialmente ocluidas. Además, el 50 % de ellas corresponden a áreas que no contienen uvas. De cada sub-imagen se extrajeron las característica de textura, orientación y color (descriptores LBP, HOG y FCH, respectivamente) y se concatenaron para obtener un vector de 516 características. El conjunto de entrenamiento generado tuvo el tamaño de 7816×516 (muestras×características). Los algoritmos evaluados fueron de nuevo: KNN, MLR, ML, MLP, RF y el basado en vectores de soporte (SVM). Se repitió el proceso de validación cruzada con 10 dobleces y se sintonizaron los parámetros para cada algoritmo. La Tabla 4.3 muestra los resultados sobre la mejor configuración de parámetros. De nueva cuenta, el algoritmo basado en bosque aleatorio (RF) presentó el mejor desempeño.

Tabla 4.3: Clasificación en imágenes de uvas individuales. Evaluación de algoritmos para la tarea de catalogar sub-imágenes de uvas individuales.

Clasificador	**Exactitud**
KNN	0.93±0.002
MLR	0.91±0.046
MaL	0.82±0.014
MLP	0.94±0.011
RF	0.95±0.007
SVM	0.94±0.009

Por un lado, en la tarea de clasificar sub-imágenes cuadradas que corresponden a uvas, Škrabánek et al. [38] obtuvo una Sensibilidad y una Precisión de 0.98 y 0.99,

respectivamente, en su conjunto de entrenamiento; mientras que para el de de pruebas, lograron una Sensibilidad de 0.99 y una *Precisión* de 0.79. Nótese que ese 0.79 en la Precisión indica un incremento del 20 % en falsos positivos. Ellos únicamente entrenaron con uvas blancas y una sola variedad. Por otro lado, Pérez-Zavala et al. [12] obtuvieron un Sensibilidad y Precisión de 0.8 y 0.88, respectivamente. Cabe destacar que ellos sí incluyeron diversas variedades en el entrenamiento. Nótese que no se pueden realizar comparaciones directas con los resultados del clasificador RF de esta investigación, ya que no se tuvo acceso a los conjuntos de datos con los que Pérez-Zavala et al. [12] se evaluó. El siguiente experimento consistió en detectar círculos dentro del conjunto RondoDS. Se evaluaron los siguientes detectores de círculos: la transformada de Hough (CHT), EDcircles [66] y CDSGA [67]; la imagen de entrada usada para detección de círculos es un mapa de bordes obtenido de la función Canny de OpenCV (sección 3.2.2). Después, se determinó con el clasificador RF, si la sub-imagen contenida en el círculo detectado era una uva (Figura 4.20).

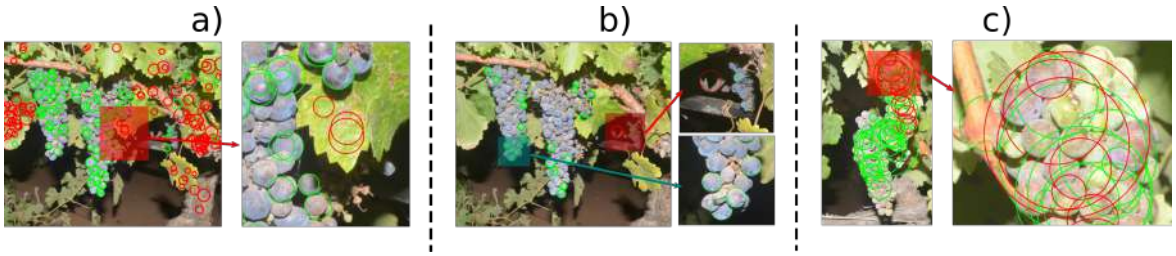


Figura 4.20: Detectando uvas como círculos. La imagen (a) presenta los resultados de la CHT, la (b) los de EDcircles [66] y la (c) los de la propuesta de esta investigación [67]. Los círculos verdes son los clasificados como uvas, mientras que los rojos son los falsos positivos. Los resultados de cada detector son presentados en la imagen donde se observaron mayormente círculos catalogados como uvas.

Nótese que el método de detección de círculos propuesto en esta investigación (Figura 4.20-c), presentó visualmente un desempeño bajo. Esto se debe a que el detector no está sintonizado para imágenes de racimos de uvas. Por otro lado, el detector EDcircles [66] es el que mejor detecta uvas completas como círculos dentro de los racimos (Figura 4.20-b), en comparación a los resultados de la CHT (Figura 4.20-a) los cuales presentaron múltiples falsos positivos. La CHT requiere múltiples parámetros de entrada para detectar círculos, tales como el radio mínimo y máximo a buscar, para este experimento se le dio el mínimo y máximo de cada imagen con respecto al etiquetado manual del conjunto RondoDS. En el trabajo de González et al. [67] se probó el método solo en imágenes genéricas y de una resolución más baja; en él también se concluyó que en algunos casos EDcircles [66] presentó el mejor desempeño para detectar círculos. El modelo RF entrenado para clasificar uvas tiene un buen desempeño al clasificar uvas dentro de imágenes digitales (Figura 4.20-a).

Aunque Murillo et al. [36] usaron la CHT para detectar frutos, no incluyeron resultados de conteo con múltiples imágenes. Por su parte en [37] presentaron una $R^2 = 0.93$, pero su desempeño de conteo solo fue medido en una muestra aleatoria de 10 por encima de las 108 imágenes capturadas. Estas incluyen uvas en una etapa temprana de crecimiento, es decir, las uvas son más pequeñas y no se aglomeran en el racimo. Basado en estos resultados y por lo antes explicado, no se profundizó más en el enfoque de detección de círculos.

4.5. ¿Cuál CNN es la más adecuada?

Hasta este punto no se ha logrado identificar frutos individuales dentro de imágenes de racimos de uvas, por esta razón, se decidió explorar el enfoque de aprendizaje profundo.

Se usó la semántica propuesta por Zabawa et al. [16, 9]: "fondo", "uva" y "borde" (la frontera entre frutos). Se empleó un esquema de aprendizaje profundo y se compararon tres arquitecturas de CNN, que abordasen la tarea de segmentación, a fin de reconocer las regiones correspondientes a frutos individuales. Para este proceso se utilizó el conjunto de datos CosmeDS: 162 imágenes (80 %) para entrenamiento y 40 (20 %) para validación. Ya que el tamaño de este conjunto incluye pocas muestras, se implementó la técnica de sobremuestreo de datos, con la intención de incrementar el número de muestras disponibles para entrenamiento. Se usaron cuatro transformaciones: *flip*, brillo, ruido y difuminado. Cada transformación se aplicó de tres maneras de cada una para obtener un incremento de 12 veces por cada muestra, obteniendo un conjunto de entrenamiento de 2106 imágenes con una resolución de 580×580 píxeles.

En esta sección se compararon tres arquitecturas de CNN y se determinó cuál de ellas presentó mejor desempeño a la tarea de segmentar frutos individuales en imágenes de racimos de uvas. Se aplicó una comparación estándar que incluyó los siguientes análisis:

1. **Proceso de entrenamiento:** se evaluó sobre los conjuntos de entrenamiento y validación en cada época de entrenamiento, para aseguras evitar sobre-ajustar el modelo. Cada CNN se entrenó de tres maneras *from scratch*, *freeze* y *fine-tune* (sección 3.5.1). En total, nueve modelos distintos fueron entrenados en este proceso.

2. **Modelos entrenados:** se tomaron los nueve modelos entrenados al cabo de 100 épocas y se evaluaron con la colección de métricas mencionadas en la sección 3.7 sobre el conjunto CosmeDS. Además, se aplicó a los resultados de cada modelo, una prueba estadística de Wilcoxon para determinar diferencia significativa entre los nueve modelos entrenados. Para esta prueba se usó un intervalo de confianza de 95 %. Los hiperparámetros utilizados se explica más adelante.

3. **Contando frutos:** por último, se tomó el modelo de cada arquitectura que presentó el mejor desempeño y se evaluó su desempeño de conteo y localización los frutos individuales con la métrica MSE; además, se tomó al modelo con el desempeño más alto y se probó en imágenes externas al conjunto CosmeDS.

A continuación se muestran los análisis previamente descritos. Tener en cuenta que en el trabajo de Zabawa et al. [9] presentaron evidencia de que la arquitectura Zabnet funcionó para este dominio imágenes, sin embargo, no demostraron cuál es la mejor arquitectura de CNN para la tarea de segmentar frutos individuales.

4.5.1. Proceso de entrenamiento

Para iniciar este experimento, se entrenaron cada arquitectura de CNN con 100 épocas; cada época corresponde a haber actualizado la arquitectura tantas veces como muestras hay en el conjunto de entrenamiento. Como algoritmo de aprendizaje se usó el descenso de gradiente estocástico (SDG, *stochastic gradient descent*) el cuál requiere dos parámetros de entrada: tasa de aprendizaje y momento; estos fueron fijados a 0.001 y 0.99, respectivamente, y fueron tomados del trabajo de UNet [56]. Estos parámetros definen qué tan rápido converge un modelo a una solución. El tamaño de lote se fijó a 1 para UNet [56] y Segnet [58], mientras que para Zabnet [9] se fijó a 2; esto se debió a la limitación de memoria. Para manejar las clases sin balance se usó la función de pérdidas entropía-cruzada ponderada (sección 2.5.3). Se calculó un valor de ponderación fijo para cada clase, haciendo un conteo de píxeles en el conjunto de entrenamiento y se obtuvo para cada categoría a aprender la siguiente frecuencia y valor de ponderación:

1. Clase 0: fondo, 72 % de píxeles; valor de ponderación 0.035.
2. Clase 1: uva, 23 %; ponderación 0.1046.
3. Clase 2: borde de uva 4 %; ponderación 0.6024.

Nótese que el valor de ponderación es mayor cuando la frecuencia de la clase es menor.

La configuración previamente explicada se utilizó para cada arquitectura comparada: Segnet, UNet y Zabnet. Cada una fue entrenada de tres maneras distintas: sin hacer transferencia de otra CNN (*from scratch*), haciéndola y congelando la parte transferida (*freeze*) y actualizando lo transferido durante el entrenamiento (*fine-tune*).

En esta sección los resultados se presentan en formato de tres gráficas. En ellas se muestran el monitoreo del entrenamiento en cada arquitectura de CNN. La primera consistió en evaluar la función de pérdidas en el conjunto de entrenamiento, mientras que la segunda y tercera en evaluar la exactitud en los conjuntos de entrenamiento y validación. Las tres gráficas se presentan en función de las épocas de entrenamiento. Para la primer gráfica se espera ver un comportamiento descendente ya que el objetivo

es minimizar las pérdidas; mientras que para la segunda y tercera se espera un comportamiento ascendente al avanzar en las épocas. Además, se busca que la exactitud sea mayor en la época 100 que en la época 10; y que la exactitud en los conjuntos de entrenamiento y validación sean cercanos, ya que de lo contrario es un indicativo que el modelo presenta sobre-ajuste.

Segnet

Para esta CNN se pudo observar que el modo *freeze* es el que mejor se desempeña al evaluarlo en el conjunto de entrenamiento (Figura 4.21-b); mientras que al evaluarlo en el conjunto de validación el modo *fine-tune* presentó el desempeño más alto (Figura 4.21-c). Esto sugiere que la arquitectura Segnet se está sobre-ajustando cuando se congela la parte transferida. Además, se observó una dispersión mayor al evaluar a Segnet sobre el conjunto de validación y esto nos habla de un modelo poco estable (Figura 4.21-c).

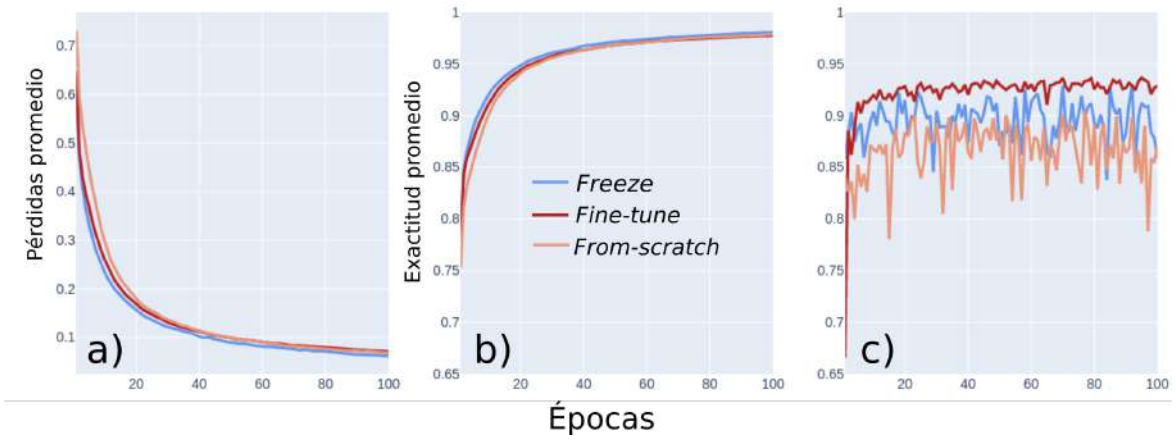


Figura 4.21: Entrenamiento de la arquitectura Segnet [58]. Resultados al evaluar la función de pérdidas (a), exactitud promedio sobre la partición de entrenamiento (b), y exactitud promedio sobre la parte de validación de CosmeDS (c). Los tres gráficos son presentados en función de cada época. Las líneas azules, rojas y naranjas corresponden a los modos de entrenamiento: *freeze*, *fine-tune* y *from-scratch*, respectivamente.

UNet

Para esta arquitectura se observó que, el modo de entrenamiento *fine-tune* fue el que mejor se desempeñó, tanto para el conjunto de entrenamiento, como para el de validación (Figura 4.22). Esto es consistente con lo reportado en el trabajo de Yosinski et al. [89]. Se transfirió a UNet una parte de la arquitectura VGG16[4] en los modos

[4]Esta arquitectura fue entrenada con ImageNet https://www.image-net.org/.

fine-tune y *freeze*. Esta idea fue usada por Pravitasari et al. [60], Cheng et al. [98] y Balakrishna et al. [99]. Ellos reportaron que se obtuvieron mejores resultados al hacer la transferencia (*fine-tune*) que entrenar desde cero (*from-scratch*), y esto es consistente con el desempeño obtenido de UNet durante esta investigación.

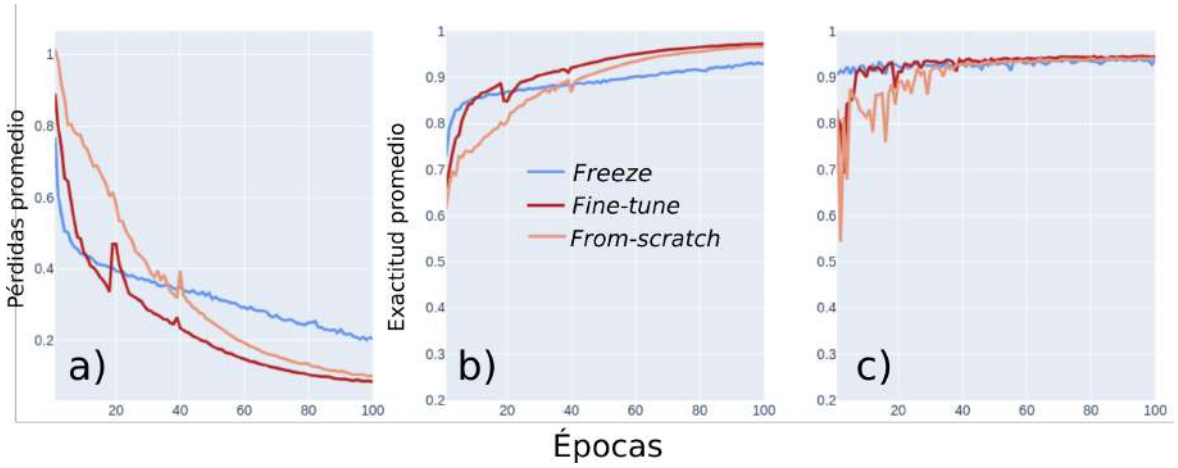


Figura 4.22: Entrenamiento de la arquitectura UNet [56]. Resultados al evaluar la función de pérdidas (a), exactitud promedio sobre la partición de entrenamiento (b), y exactitud promedio sobre la parte de validación de CosmeDS (c). Los tres gráficos son presentados en función de cada época. Las líneas azules, rojas y naranjas corresponden a los modos de entrenamiento: *freeze*, *fine-tune* y *from-scratch*, respectivamente.

Zabnet

De nueva cuenta, para la arquitectura Zabnet [9] se observó que hacer transferencia de una CNN pre-entrenada es la configuración que resulta con mejor desempeño (Figura 4.23). Se observó este comportamiento tanto para el conjunto de entrenamiento como para el de validación. La versión *fine-tune* de Zabnet es la que presentó el mejor desempeño, en comparación a los otros modos de entrenamiento (*freeze* y *from-scratch*).

Al observar el desempeño de cada arquitectura de CNN evaluada sobre el conjunto de validación (figuras 4.23, 4.22 y 4.21), se observó que UNet [56] en el modo de entrenamiento *fine-tune* presentó el mejor desempeño. UNet alcanzó una exactitud promedio de 0.95, mientras que Segnet [58] y Zabnet [9] lograron 0.93 y 0.92, respectivamente. La diferencia es pequeña ente el desempeño de cada CNN, y es muy pronto para llegar a una conclusión. Posteriormente, se tomaron los nueve modelos entrenados y se evaluó cuál de ellos localizó mejor los frutos individuales dentro de las imágenes del conjunto CosmeDS.

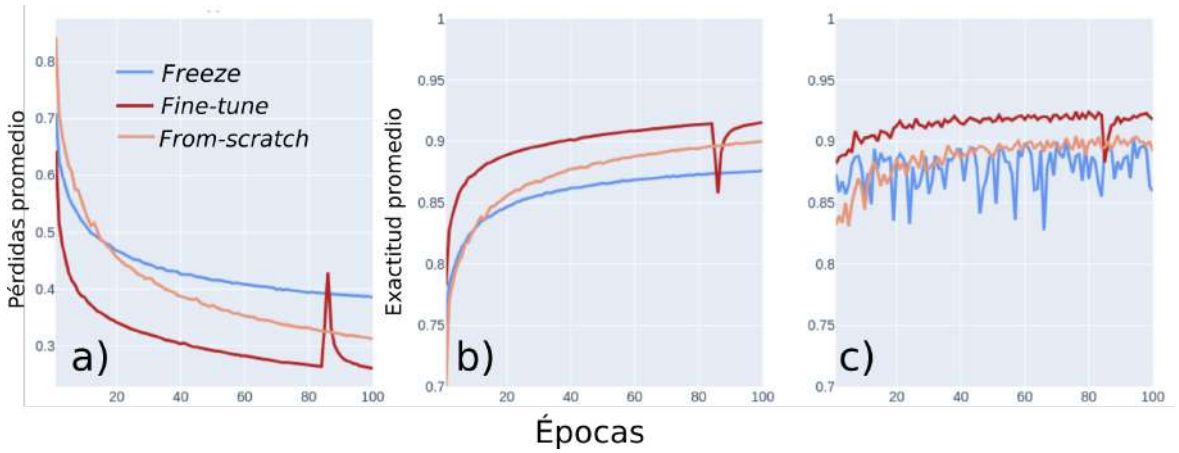


Figura 4.23: Entrenamiento de la arquitectura Zabnet [9]. Resultados al evaluar la función de pérdidas (a), exactitud promedio sobre la partición de entrenamiento (b), y exactitud promedio sobre la parte de validación de CosmeDS (c). Los tres gráficos son presentados en función de cada época. Las líneas azules, rojas y naranjas corresponden a los modos de entrenamiento: *freeze*, *fine-tune* y *from-scratch*, respectivamente.

4.5.2. Modelos entrenados

Como segundo análisis, se evaluaron los nueve modelos entrenados sobre el conjunto CosmeDS (particiones de entrenamiento y validación). Se usaron las métricas F1-score para cada clase y el coeficiente de correlación de Matthew (MCC). Las tablas 4.4 y 4.5 muestran los resultados de la evaluación de cada uno de los nueve modelos entrenados. Esta evaluación se hizo para los conjuntos de entrenamiento (162 muestras) y validación (40 muestras) del conjunto de CosmeDS. Los resultados de cada métrica se presentan en forma de promedio ± desviación estándar en cada partición de CosmeDS; además, se indicó a la CNN con el desempeño más alto de cada modo de entrenamiento indicado en negritas y al mejor desempeño sobre las tres arquitecturas resaltado de color gris.

Al observar los resultados sobre la clase "borde", UNet en el modo de entrenamiento *fine-tune* fue la que mejor se desempeñó logrando un F1-score de 0.82 y 0.58, para las particiones de entrenamiento y validación, respectivamente (tablas 4.4 y 4.5). Se tomó en consideración la clase "borde" ya que es la que hizo posible separar frutos individuales. Para determinar si hay diferencia estadística se aplicó de nueva cuenta, una prueba de Wilcoxon a lo largo de los modos de entrenamiento en cada arquitectura, a fin de seleccionar aquel con el mejor desempeño. El modo *fine-tune* evaluado sobre el conjunto de entrenamiento, presentó diferencia significativa con respecto a los demás (ver Tabla 4.4); mientras que al evaluar sobre el conjunto de validación, en el caso de UNet, no hay diferencia significativa entre los modos *freeze* y *fine-tune* (Tabla 4.5). Los detalles de las pruebas de Wilcoxon son mostrados en las tablas 4.6, 4.7 y 4.8.

Posteriormente, se llevó a cabo una segunda comparación para determinar el desem-

Tabla 4.4: Resultados sobre el conjunto CosmeDS, partición de entrenamiento. Las celdas con "*" corresponden a la existencia de diferencia significativa, cuando se compara con los otros dos modos de entrenamiento. Los renglones resaltados de color gris corresponden al mejor de las tres arquitecturas. Se usó una prueba de Wilcoxon con un valor $p = 0.05$ para verificar si hay diferencia estadística significativa.

Arquitectura	**Modo de entrenamiento**	**F1-score fondo**	**F1-Score uva**	**F1-score borde**	**MCC**
Zabnet	*From -scratch*	0.97 ±0.02	0.88±0.06	0.49±0.08	0.82±0.04
	Freeze	0.94±0.04	0.82±0.09	0.43±0.09	0.74±0.09
	Fine -tune	**0.98±0.01***	**0.91±0.04***	**0.56±0.08***	**0.85±0.04***
UNet	*From -scratch*	0.99 ±0.01	0.98±0.01	0.80±0.06	0.95±0.02
	Freeze	0.99±0.01	0.95±0.01	0.70±0.08	0.92±0.03
	Fine -tune	**0.99±0.004***	**0.98±0.01***	**0.82±0.05***	**0.96±0.02***
Segnet	*From -scratch*	0.95 ±0.04	0.80±0.19	0.50±0.14	0.75±0.17
	Freeze	0.90±0.08	0.62±0.31	0.40±0.20	0.60±0.27
	Fine -tune	**0.97±0.04***	**0.88±0.13***	**0.59±0.13***	**0.83±0.12***

Tabla 4.5: Resultados sobre el conjunto CosmeDS, partición de validación. Las celdas con "*" corresponden a la existencia de diferencia significativa, cuando se compara con los otros dos modos de entrenamiento. Los renglones resaltados de color gris corresponden al mejor de las tres arquitecturas. Se usó una prueba de Wilcoxon con un valor $p = 0.05$ para verificar si hay diferencia estadística significativa..

Arquitectura	**Modo de entrenamiento**	**F1-score fondo**	**F1-Score uva**	**F1-score borde**	**MCC**
Zabnet	*From -scratch*	0.96± 0.02	0.83 ± 0.09	0.43 ± 0.08	0.76 ± 0.8
	Freeze	0.93±0.04	0.78±0.12	0.42±0.09	0.69±0.12
	Fine -tune	**0.97±0.01***	**0.88±0.06***	**0.50±0.09***	**0.82±0.06***
UNet	*From -scratch*	0.98 ±0.01	0.91±0.04	0.55±0.08	0.85±0.04
	Freeze	0.98±0.01	0.91±0.04	0.59±0.08	0.86±0.04
	Fine -tune	0.98±0.01*	0.92±0.03*	0.58±0.08*	0.87±0.04*
Segnet	*From -scratch*	0.93 ±0.04	0.72±0.17	0.33±0.09	0.67±0.15
	Freeze	0.92±0.07	0.64±0.31	0.39±0.19	0.61±0.30
	Fine -tune	**0.97±0.02***	**0.87±0.08***	**0.52±0.09***	**0.82±0.08***

peño de cada arquitectura; se usó de nuevo una prueba de Wilcoxon. De nueva cuenta, se observó que UNet en el modo de entrenamiento *fine-tune* presentó el mejor desempeño(Tabla 4.9).

Tabla 4.6: Resultados de las pruebas de Wilcoxon en la arquitectura Zabnet [9].

***Fine-tune* vs**	**F1-Score Fondo**	**F1-Score Uva**	**F1-Score Borde**	**MCC**
From -scratch	**0.00***	**0.00***	**0.00***	**0.00***
Freeze	**0.00***	**0.00***	**0.00***	**0.00***

(a) Conjunto de entrenamiento.

***Fine-tune* vs**	**F1-Score Fondo**	**F1-Score Uva**	**F1-Score Borde**	**MCC**
From -scratch	**0.00***	**0.00***	**0.00***	**0.00***
Freeze	**0.00***	**0.00***	**0.00***	**0.00***

(b) Conjunto de validación.

Tabla 4.7: Resultados de las pruebas de Wilcoxon en la arquitectura UNet [56].

***Fine-tune* vs**	**F1-Score Fondo**	**F1-Score Uva**	**F1-Score Borde**	**MCC**
From -scratch	**0.00***	**0.00***	**0.00***	**0.00***
Freeze	**0.00***	**0.00***	**0.00***	**0.00***

(a) Conjunto de entrenamiento.

***Fine-tune* vs**	**F1-Score Fondo**	**F1-Score Uva**	**F1-Score Borde**	**MCC**
From -scratch	**0.00***	**0.00***	**0.00***	**0.00***
Freeze	0.45	0.18	0.64	0.26

(b) Conjunto de validación.

Tabla 4.8: Resultados de las pruebas de Wilcoxon en la arquitectura Segnet [58].

***Fine-tune* vs**	**F1-Score Fondo**	**F1-Score Uva**	**F1-Score Borde**	**MCC**
From -scratch	**0.00***	**0.00***	**0.00***	**0.00***
Freeze	**0.00***	**0.00***	**0.00***	**0.00***

(a) Conjunto de entrenamiento.

***Fine-tune* vs**	**F1-Score Fondo**	**F1-Score Uva**	**F1-Score Borde**	**MCC**
From -scratch	**0.00***	**0.00***	**0.00***	**0.00***
Freeze	**0.00***	**0.00***	**0.00***	**0.00***

(b) Conjunto de validación.

Tabla 4.9: Resultados de las pruebas Wilcoxon sobre las tres arquitecturas de CNN comparadas en este trabajo.

***Fine-tune* vs**	**F1-Score Fondo**	**F1-Score Uva**	**F1-Score Borde**	**MCC**
Segnet	**0.00***	**0.00***	**0.00***	**0.00***
Zabnet	**0.00***	**0.00***	**0.00***	**0.00***

(a) Conjunto de entrenamiento.

***Fine-tune* vs**	**F1-Score Fondo**	**F1-Score Uva**	**F1-Score Borde**	**MCC**
Segnet	**0.00***	**0.00***	**0.00***	**0.00***
Zabnet	**0.00***	**0.00***	**0.00***	**0.00***

(b) Conjunto de validación.

Zabawa et al. [9] reportaron un desempeño de 0.99, 0.75 y 0.53 para las clases "fondo", "uva" y "borde" de uva, respectivamente. Ellos usaron la métrica IoU[5]. Mientras que UNet alcanzó un F1-score promedio de 0.98, 0.92 y 0.58, respectivamente para las mismas categorías. Ambas métricas están en el intervalo de $[0, 1]$ y aunque no es típico comparar resultados usando métricas distintas, ambas se pueden relacionar a través de la siguiente ecuación[6]

$$\frac{IoU}{F1 - Score} = \frac{1}{2} + \frac{IoU}{2} \tag{4.1}$$

Esto quiere decir que su cociente se aproxima a $1/2$ mientras ambas se aproximan a 0. En otras palabras, si una métrica nos dice si el clasificador A es mejor que el B, la otra también lo dirá.

El objetivo en este experimento fue evaluar el desempeño de tres arquitecturas de CNN, que fueron desarrolladas para abordar la tarea de segmentación semántica y que además, se pudieran adaptar al dominio de imágenes de racimos de uvas. A diferencia de Zabawa et al. [9] que compararon con dos arquitecturas que abordaban tareas distintas a la de segmentación. La decisión de escoger las tres arquitecturas evaluadas en este trabajo es debido a que Zabnet [9] se probó en imágenes de racimos de uvas, y además su red base DeepLabV3+ [59] superó en el 2018 los resultados más altos en el conjunto de PASCAL VOC 2012[7]; Segnet [58] mostró robustez al segmentar imágenes de exteriores; y UNet [56] se probó en imágenes médicas donde el nivel segmentación requerido es más fino.

Hasta este punto, solo se ha evaluado el desempeño de segmentación semántica de manera global, pero no se ha revisado si los cúmulos de píxeles clasificados como "uva" realmente corresponden a frutos individuales dentro de la imagen.

4.5.3. Contando frutos

En esta sección se evaluó el desempeño de conteo y localización de frutos individuales en cada arquitectura de CNN. Se usaron de nuevo los gráficos-R para presentar los resultados. Cada gráfica representa una dispersión del conteo manual (eje horizontal, C_i) contra el conteo estimado (eje vertical $\hat{C}_i$); el conteo manual se extrajo de las imágenes etiquetadas, mientras que el estimado corresponde a identificar los grupos de píxeles segmentados como "uva", y si la uva segmentada está cerca de alguna manualmente anotada (ecuación 3.34).

A pesar de que es común ver en los gráficos-R, una recta que se ajusta a cada dispersión, en este trabajo se propuso incluir la recta que corresponde al conteo ideal,

[5] *Intersection over union* o también conocida como índice Jaccard

[6] https://stats.stackexchange.com/questions/273537/f1-dice-score-vs-iou, accesado el 3 de julio de 2022

[7] Conjunto de datos tipo *benchmark* para evaluar la tarea de clasificación o segmentación: http://host.robots.ox.ac.uk/pascal/VOC/

el MSE del conteo estimado (ecuación A.44), y el porcentaje de conteo que cayó cerca de del conteo ideal más un valor de tolerancia permitido, el cual se fijó en 20 unidades (uvas). Puesto que UNet es la arquitectura que presentó el mejor desempeñó en los dos análisis anteriores, en esta evaluación se compararon sus resultados contra las demás CNNs.

Primero, en los resultados de Segnet no se observó ningún patrón de conteo (Figura 4.24), es decir, sin importar la cantidad de uvas en la imagen se obtuvo un desempeño de conteo bajo. Esta arquitectura alcanzó 61.73 % y 62.5 % de desempeño de conteo para los conjuntos de entrenamiento y validación, respectivamente. Segundo, UNet logró un desempeño de conteo de 98.77 % y 87.5 % para los conjuntos entrenamiento y validación, respectivamente (Figura 4.25). Por último, Zabnet presentó un 46.91 % y 75 % de desempeño de conteo para los conjuntos entrenamiento y validación, respectivamente. Zabnet mostró una reducción en su conteo al incrementar el número de uvas aumenta dentro de la imagen (Figura 4.26); esto fue consistente con los resultados reportados en su trabajo [9].

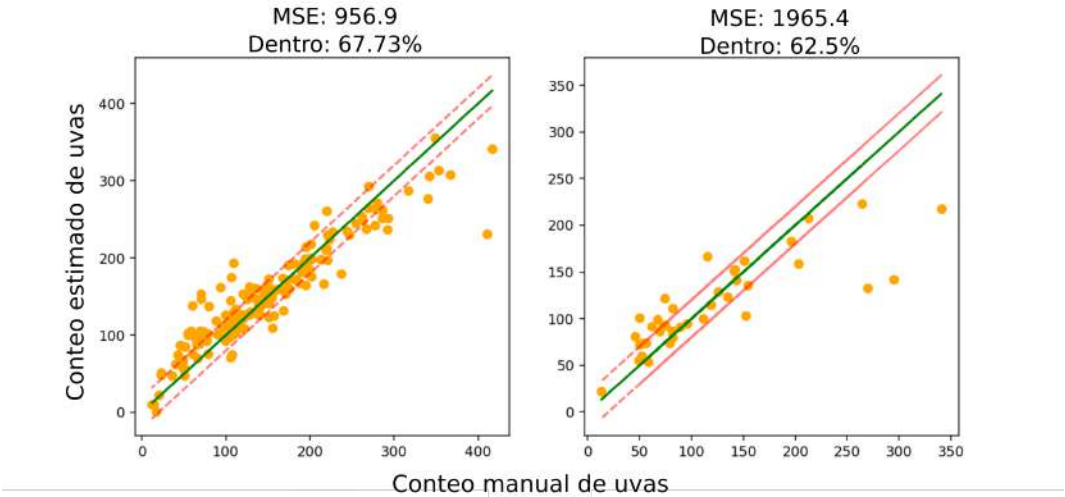


Figura 4.24: Gráfico-R del la arquitectura Segnet [58]. La gráfica del lado izquierdo presenta el conteo sobre el conjunto de entrenamiento, mientras que la gráfica del lado derecho muestra el conteo del conjunto de validación de CosmeDS. El eje horizontal corresponde al conteo manual, mientras que el eje vertical corresponde al conteo estimado. La línea verde representa el conteo perfecto ($R^2 = 1$), mientras que la línea roja al valor de tolerancia permitido. El MSE y el porcentaje que cae cerca al conteo ideal son incluidos arriba de cada gráfico.

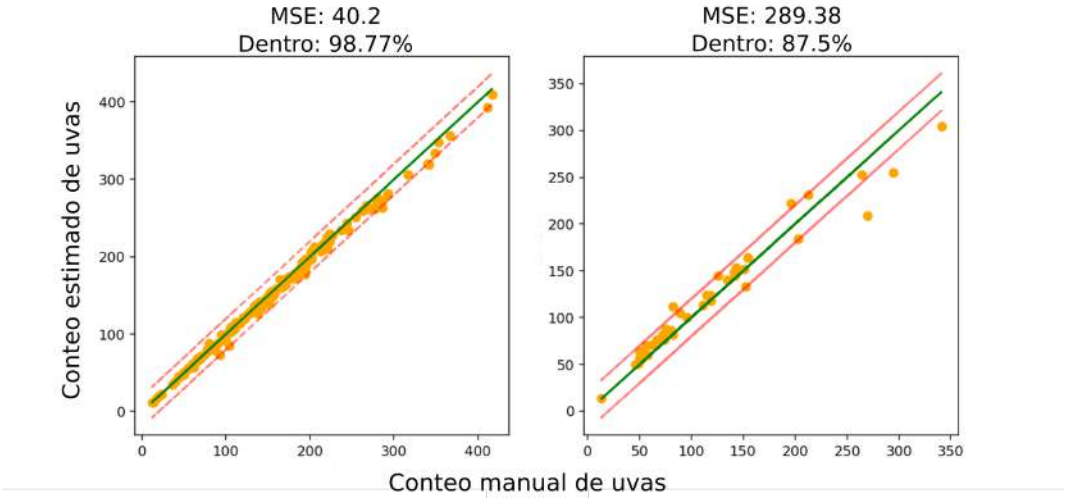


Figura 4.25: Gráfico-R del la arquitectura UNet [56]. La gráfica del lado izquierdo presenta el conteo sobre el conjunto de entrenamiento, mientras que la gráfica del lado derecho muestra el conteo del conjunto de validación de CosmeDS. El eje horizontal corresponde al conteo manual, mientras que el eje vertical corresponde al conteo estimado. La línea verde representa el conteo perfecto ($R^2 = 1$), mientras que la línea roja al valor de tolerancia permitido. El MSE y el porcentaje que cae cerca al conteo ideal son incluidos arriba de cada gráfico.

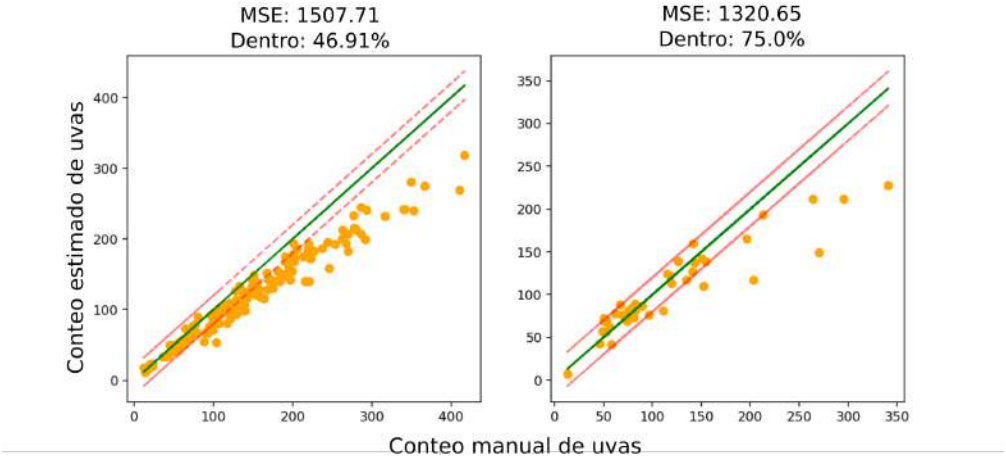


Figura 4.26: Gráfico-R del la arquitectura Zabnet [9].La gráfica del lado izquierdo presenta el conteo sobre el conjunto de entrenamiento, mientras que la gráfica del lado derecho muestra el conteo del conjunto de validación de CosmeDS. El eje horizontal corresponde al conteo manual, mientras que el eje vertical corresponde al conteo estimado. La línea verde representa el conteo perfecto ($R^2 = 1$), mientras que la línea roja al valor de tolerancia permitido. El MSE y el porcentaje que cae cerca al conteo ideal son incluidos arriba de cada gráfico.

Debido a los resultados de los análisis previos y a los resultados de conteo de este experimento, se declaró a UNet la arquitectura que presentó el mejor desempeño para segmentar frutos individuales en el dominio de imágenes de racimos de uvas. Cuando UNet fue propuesta en [56], fue entrenada con imágenes que contenían células aglomeradas, y el reto consistía en separarlas a pesar de los bordes poco definidos entre ellas. Esta problemática luce parecida a separar uvas aglomeradas dentro de un racimo donde los bordes se captura difuminados en una imagen digital. Un factor a considerar para su implementación final es que UNet requiere más de 30 millones de parámetros para procesar la imagen, por lo que usarla en dispositivo (*hardware*) de recursos limitados no es posible.

UNet en los conjuntos Kicherer y BIVcolor

En esta sección se evaluó el desempeñó de UNet sobre dos conjuntos externos a los de entrenamiento. BIVcolor [45] contiene 100 sub-imágenes; mientras que Kicherer [51]) contiene 50 sub-imágenes Ambos conjuntos incluyen la posición central de cada uva capturada y esto se etiquetó manualmente para ser capaz de evaluar el desempeño de conteo. UNet presentó un desempeño de conteo sobre BIVcolor [45] de 65.91 % (Figura 4.27), mientras que para Kicherer [51] resultó en 19.61 % (Figura 4.28). El porcentaje bajo se debió a que Kicherer [51] solo contiene una variedad de uvas blancas (Riesling). Para esto, se requirió un experimento posterior para incluir uvas blancas en el entrenamiento UNet.

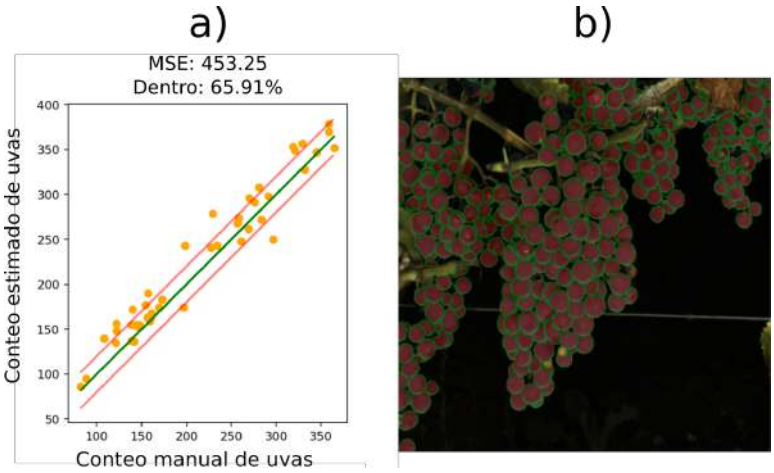


Figura 4.27: Evaluación de UNet sobre el conjunto BIVcolor [45]. En el gráfico-R (a) el eje horizontal corresponde al conteo manual, mientras que el eje vertical corresponde a la predicción. La línea verde representa el conteo perfecto ($R^2 = 1$), mientras que la línea roja indica el valor de tolerancia para este conteo ideal. El MSE y el porcentaje que cae cerca al conteo ideal son incluidos en la parte superior. (b) Se incluyó una muestra de la imagen en donde se observa con la predicción de UNet sobrepuesta a la imagen original.

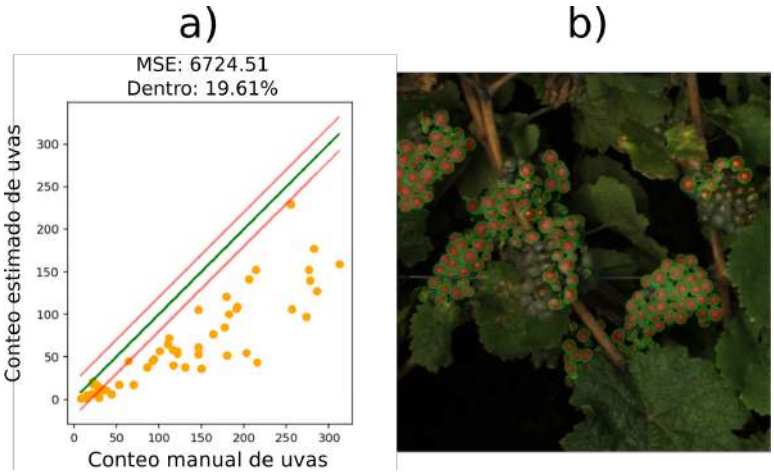


Figura 4.28: Evaluación de UNet sobre el conjunto Kicherer [51]. En el gráfico-R (a) el eje horizontal corresponde al conteo manual, mientras que el eje vertical corresponde a la predicción. La línea verde representa el conteo perfecto ($R^2 = 1$), mientras que la línea roja indica el valor de tolerancia para este conteo ideal. El MSE y el porcentaje que cae cerca al conteo ideal son incluidos en la parte superior.(b) Se incluyó una muestra de la imagen en donde se observa con la predicción de UNet sobrepuesta a la imagen original.

4.5.4. La relevancia de los datos de entrenamiento

Como se mencionó anteriormente, los datos de entrada juegan un papel importante en las aplicaciones específicas de los algoritmos de aprendizaje de máquina. Al entrenar a UNet únicamente con variedades de uvas tintas, disminuyó su desempeño ante algunos casos con uvas blancas (Figura 4.28). Para contrarrestar esto, se incluyó dentro del entrenamiento al conjunto de ZabawaDS (sección 3.1.2). Se tomaron 162 (80 %) de las 202 sub-imágenes extraídas de [47] para entrenamiento y se agregaron a las 162 del conjunto CosmeDS. Las imágenes restantes (40) también se anexaron a la partición de validación del conjunto CosmeDS. Para contar ahora con un conjunto de 324 imágenes para entrenamiento y 80 de validación. A la partición de entrenamiento se aplicó de nueva cuenta, el proceso de sobremuestreo de datos (sección 4.5), esto con el fin de obtener 4212 imágenes de 580×580 píxeles para entrenamiento.

El primer experimento consistió en definir si era necesario aplicar sobremuestreo a las imágenes de entrenamiento. Para esto, se entrenó a UNet en su modo *fine-tune* con el conjunto de datos híbrido **CosmeDS+ZabawaDS** aplicando y sin aplicar sobremuestreo. Para realizar el entrenamiento se utilizó la misma configuración del experimento anterior (sección 4.5.1). Se observó que UNet presentó un desempeñó más cuando fue entrenado con un conjunto de datos sobremuestreado (Figura 4.29).

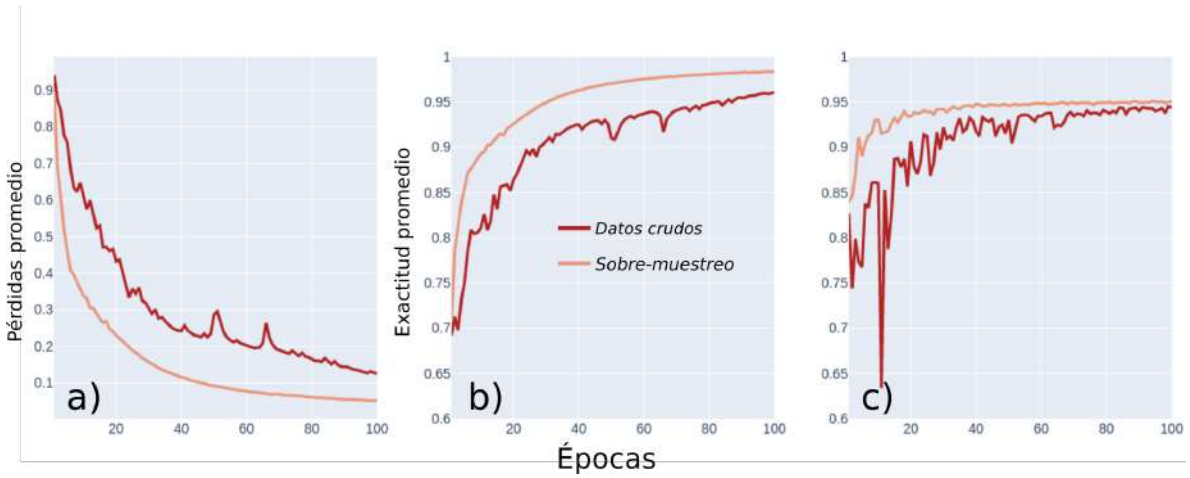


Figura 4.29: Curvas de entrenamiento de UNet-*fine-tune* con el conjunto híbrido CosmeDS+ZabawaDS. La gráfica (a) corresponde a las pérdidas, mientras que la gráfica (b) a la exactitud sobre la partición de entrenamiento, y la gráfica (c) a la exactitud sobre el conjunto de validación. En cada gráfica se muestran dos trazos en función de cada época. El trazo naranja corresponde al incluir sobremuestreo de datos, mientras que el trazo rojo corresponde a no incluirlo.

Se tomó cada uno de los modelos entrenados en este experimento, y se evaluaron a través de la métrica F1-score sobre cada clase aprendida (tablas 4.10 y 4.11). Para

verificar si hay diferencia significativa en los resultados, se aplicó de nueva cuenta, una prueba de Wilcoxon.

Tabla 4.10: Resultados sobre el conjunto de entrenamiento. Las celdas con ***** corresponden a la existencia de diferencia significativa, cuando se compara el uso de DA. Usamos una prueba Wilcoxon con un valor $p = 0.05$ para establecer diferencia estadística.

Tipo	**F1-Score Fondo**	**F1-Score Uva**	**F1-Score Borde**
Con DA	**0.997 ± 0.003***	**0.983 ± 0.005***	**0.886 ± 0.036***
Sin DA	0.989 ± 0.008	0.93 ± 0.02	0.68 ± 0.074

Tabla 4.11: Resultados sobre el conjunto de validación. Las celdas con ***** corresponden a la existencia de diferencia significativa, cuando se compara el uso de DA. Usamos una prueba Wilcoxon con un valor $p = 0.05$ para establecer diferencia estadística.

Tipo	**F1-Score Fondo**	**F1-Score Uva**	**F1-Score Borde**
Con DA	0.982 ± 0.009	**0.892 ± 0.05 ***	0.586 ± 0.104
Sin DA	0.98 ± 0.009	0.88 ± 0.05	0.59 ± 0.08

Se observó que hubo diferencia significativa al evaluar sobre el conjunto de entrenamiento en todas las clases, mientras que para el conjunto de validación, solo hay diferencia significativa para la clase "uva". Zabawa et al. [9] reportaron un F1-score de 0.75 y 0.53 para las clases "uva" y "borde de uva", respectivamente; mientras que la arquitectura UNet, entrenada con el conjunto híbrido, alcanzó 0.89 y 0.58 para las mismas clases, respectivamente.

Por último, se evaluó el desempeño de conteo y se presentaron los resultados en gráfico-R. Se observó que hay una pequeña disminución en el desempeño de UNet entrenada con el conjunto híbrido (Figura 4.30) en comparación a los obtenidos cuando solo se entrenó con el conjunto CosmeDS (Figura 4.25).

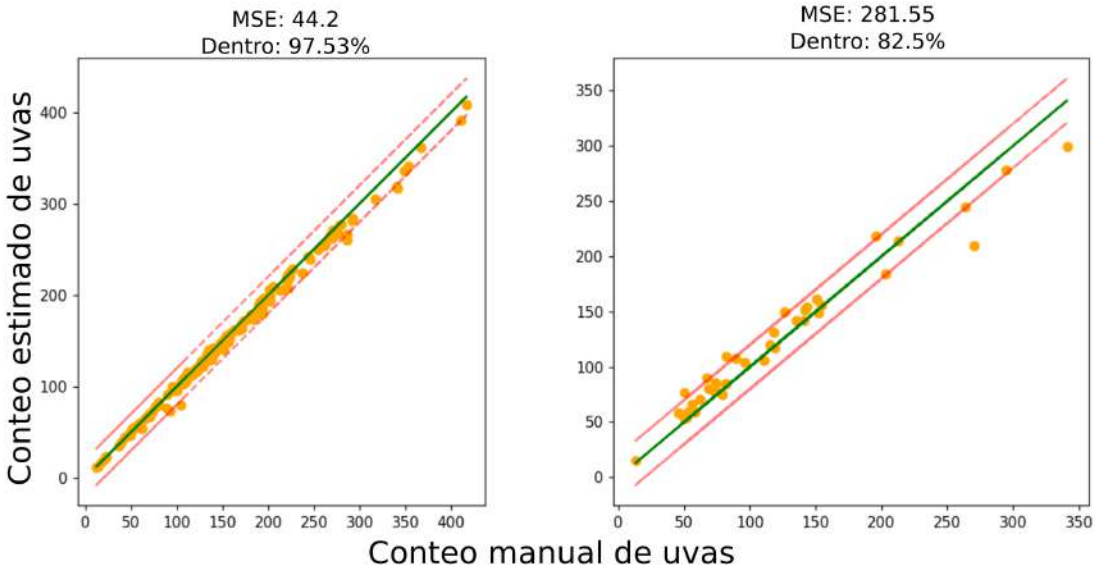


Figura 4.30: Gráfico-R sobre el conjunto **CosmeDS**. Resultados de la arquitectura UNet usando como entrenamiento con el conjunto híbrido CosmeDS+ZabawaDS. La gráfica izquierda representa el conteo sobre el conjunto de entrenamiento, mientras que la gráfica derecha al conjunto de validación. El eje horizontal corresponde al conteo manual, mientras el vertical a la predicción. La línea verde representa el conteo perfecto ($R^2 = 1$), mientras que la línea roja marca la tolerancia permitida para el conteo ideal. El MSE y el porcentaje próximo al conteo ideal son incluidos en la parte superior de cada gráfico.

Se observó que los resultados al evaluar sobre el conjunto ZabawaDS [47] resultó en un MSE de 2.72 y 87.35 para las particiones de entrenamiento y validación, respectivamente (Figura 4.31); estos resultados de conteo son los más altos reportados en esta investigación.

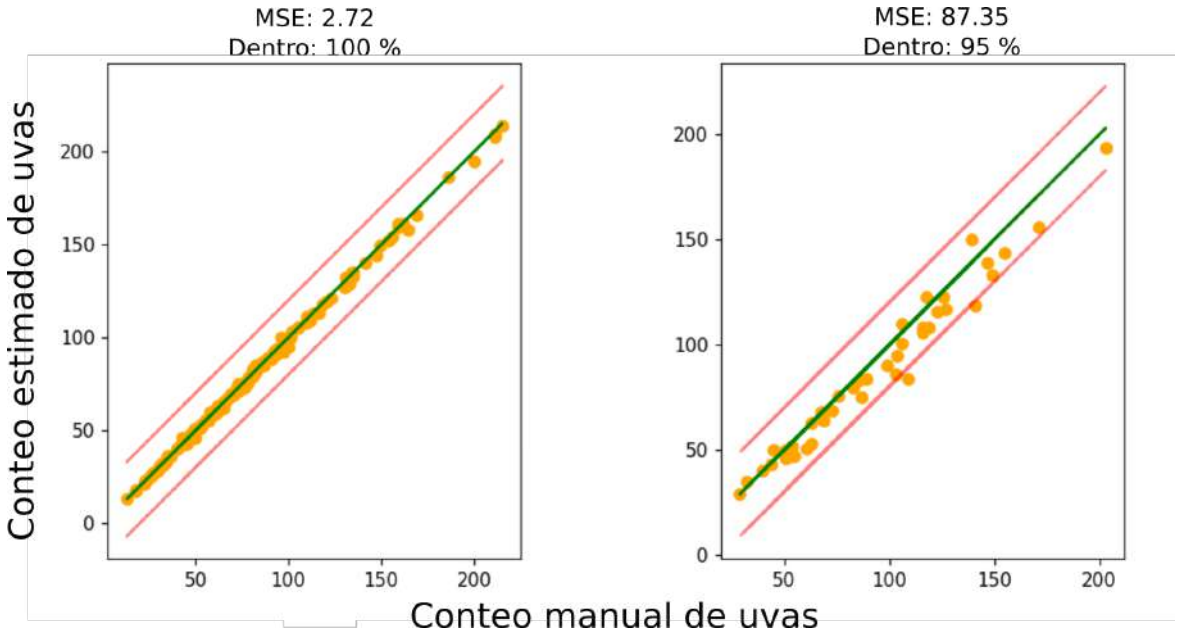


Figura 4.31: Gráfico-R sobre el conjunto **ZabawaDS** [47]. Resultados de la arquitectura UNet usando como entrenamiento con el conjunto híbrido CosmeDS+ZabawaDS. La gráfica izquierda representa el conteo sobre el conjunto de entrenamiento, mientras que la gráfica derecha al conjunto de validación. El eje horizontal corresponde al conteo manual, mientras el vertical a la predicción. La línea verde representa el conteo perfecto ($R^2 = 1$), mientras que la línea roja marca la tolerancia permitida para el conteo ideal. El MSE y el porcentaje próximo al conteo ideal son incluidos en la parte superior de cada gráfico.

De los resultados reportados por trabajo de Zabawa et al. [9] se observaron dos datos relevantes. El primero es que reportan una $R^2 = 0.97$, sin embargo, solo incluyeron los conteos de una de las tres variedades de uvas con las que entrenaron a Zabnet (Riesling). El segundo es que agregaron en el conteo sub-imágenes que no contienen uvas y su estimación fue incorrecta hasta por más de 10 unidades (uvas). Por un lado, el conjunto híbrido (CosmeDS+ZabawaDS) incluye únicamente imágenes que si contienen racimos dentro; por otro lado, en vez de ajustar una recta a la estimación de conteo, se incluyó el porcentaje estimaciones que cayó cerca del conteo ideal. Cuando se evaluó la versión de UNet entrenada en esta sección, sobre el conjunto de ZabawaDS (sección 3.1.2) se obtuvo un desempeño de conteo de 100 % y 95.5 % para los conjuntos de entrenamiento y validación, respectivamente.

Ahora se probó esta versión de UNet sobre los conjuntos BIVcolor [45] y Kicherer [51]. Para el primer conjunto se observó una disminución en el MSE de 453.25 a 347.16 (Figura 4.32-izquierda); mientras que para el segundo ocurrió el mismo comportamiento, pasando de un MSE igual a 6274.52 a 1456.88 (Figura 4.32-derecha).

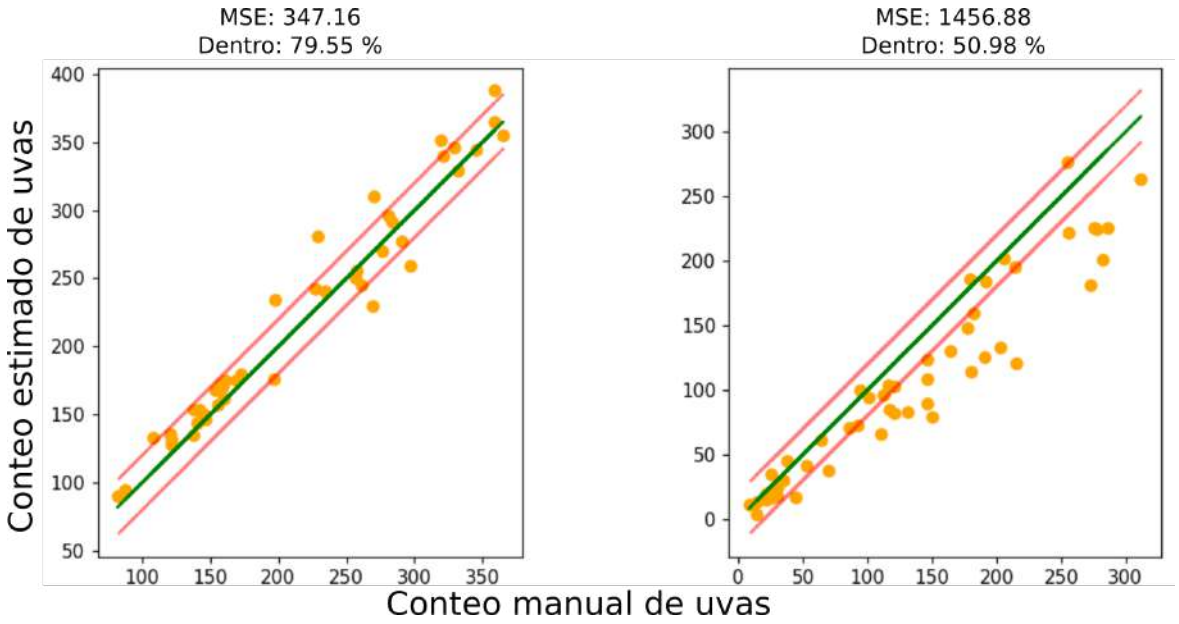


Figura 4.32: Gráfico-R de la arquitectura UNet [56] usando como entrenamiento con el conjunto híbrido CosmeDS+ZabawaDS. La gráfica izquierda representa el conteo sobre el conjunto **BIVcolor** [45], mientras que la gráfica derecha al conjunto **Kicherer** [51]. El eje horizontal corresponde al conteo manual, mientras el vertical a la predicción. La línea verde representa el conteo perfecto ($R^2 = 1$), mientras que la línea roja marca la tolerancia permitida para el conteo ideal. El MSE y el porcentaje próximo al conteo ideal son incluidos en la parte superior de cada gráfico.

El haber incluido uvas blancas en el entrenamiento de UNet mejoró el desempeño de conteo con respecto a imágenes con este tipo de uvas. Esto se pudo observar al evaluarla sobre el conjunto de Kicherer [51], ya que aumentó el desempeño de conteo de 19.61 % (Figura 4.28) a 50.98 % (Figura 4.32-derecha).

4.6. Análisis de conteo en racimos individuales

Para este último experimento se usaron las imágenes de la última serie de capturas en San Cosme Viñedos (sección 3.1.1). Este conjunto contiene imágenes de 30 racimos individuales de cuatro variedades de uvas. De cada racimo se capturaron cuatro imágenes que corresponden a cuatro ángulos distintos, para tener un total de 120 imágenes

por variedad.

Se tomó a UNet entrenada con el conjunto híbrido (CosmeDS + ZabawaDS) y se procesó cada una de las 120 imágenes. A cada imagen procesada, se aplicó el proceso de detección de área ocluida basada en arcos (sección 3.6) a fin de estimar la distribución de tallas y tomar en esta estimación el área parcialmente oculta de los frutos.

En esta sección se muestran los resultados en el siguiente formato: un gráfico-R con el conteo de las 120 capturas por variedad, que incluye el conteo manual y la estimación de frutos. Se incluyó también, una muestra de las cuatro capturas de un racimo con el conteo de círculos detectados, en la parte superior de cada imagen. Además, debajo de cada una se agregó su respectiva distribución de tallas en un gráfico de barras.

Se presentaron los resultados sobre cada variedad capturada: Merlot (Figura 4.33), Cabernet Sauvignon (Figura 4.34), Cabernet Franc (Figura 4.35) y Syrah (Figura 4.36). Se resalta que el desempeño de conteo sobre las cuatro variedades supera al 90 %. Además, la detección de arcos no solo sirvió para estimar la cantidad de frutos, sino también para calcular el área oculta y ser capaz de presentar una distribución de tallas mas cercana a la distribución real de los frutos individuales en un racimo.

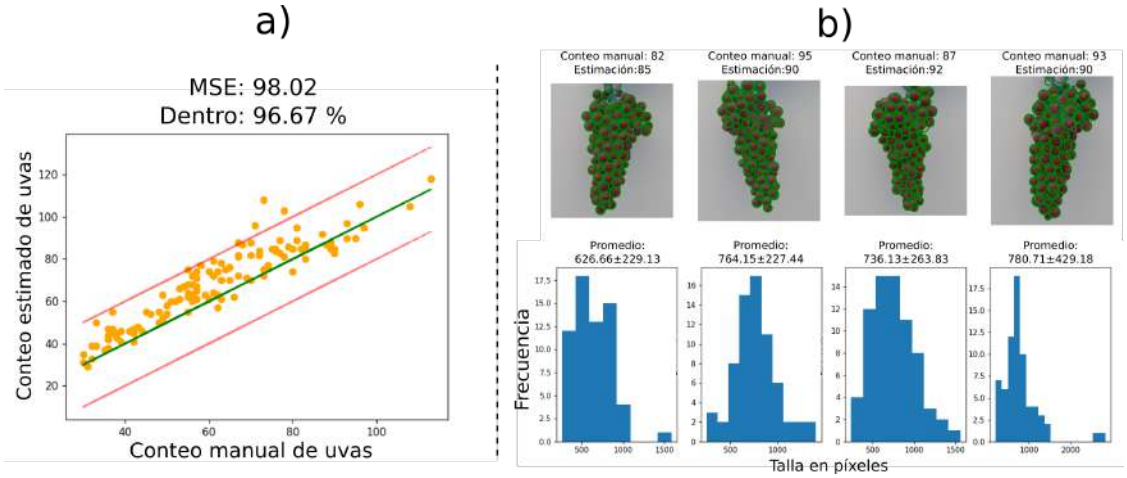


Figura 4.33: Racimos individuales, variedad: **Merlot**. La imagen (a) presenta un gráfico-R donde el eje horizontal corresponde al conteo manual, mientras que el eje vertical corresponde a la predicción, la línea verde representa el conteo perfecto ($R^2 = 1$), mientras que la línea roja marca la tolerancia para este conteo ideal. El MSE y el porcentaje próximo al conteo ideal son incluidos en la parte superior del gráfico. La imagen (b) presenta cuatro capturas realizadas a un racimo muestra; y sobrepuestos se pueden observar los círculos correspondientes a los arcos detectados para estimar el área oculta. En la parte inferior de cada imagen se incluyó la distribución de tallas en píxeles.

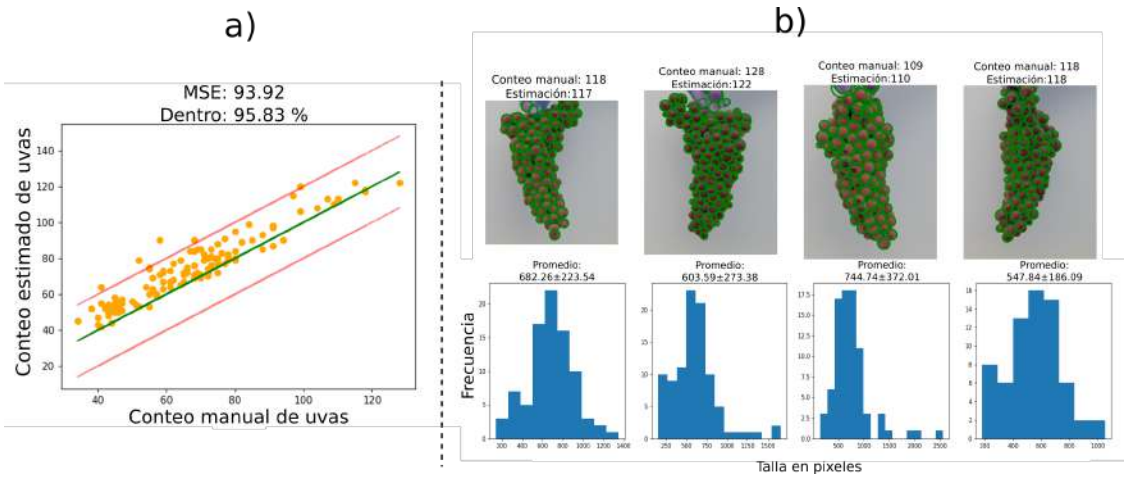


Figura 4.34: Racimos individuales, variedad: **Cabernet Sauvignon**. La imagen (a) presenta un gráfico-R donde el eje horizontal corresponde al conteo manual, mientras que el eje vertical corresponde a la predicción, la línea verde representa el conteo perfecto ($R^2 = 1$), mientras que la línea roja marca la tolerancia para este conteo ideal. El MSE y el porcentaje próximo al conteo ideal son incluidos en la parte superior del gráfico. La imagen (b) presenta cuatro capturas realizadas a un racimo muestra; y sobrepuestos se pueden observar los círculos correspondientes a los arcos detectados para estimar el área oculta. En la parte inferior de cada imagen se incluyó la distribución de tallas en píxeles.

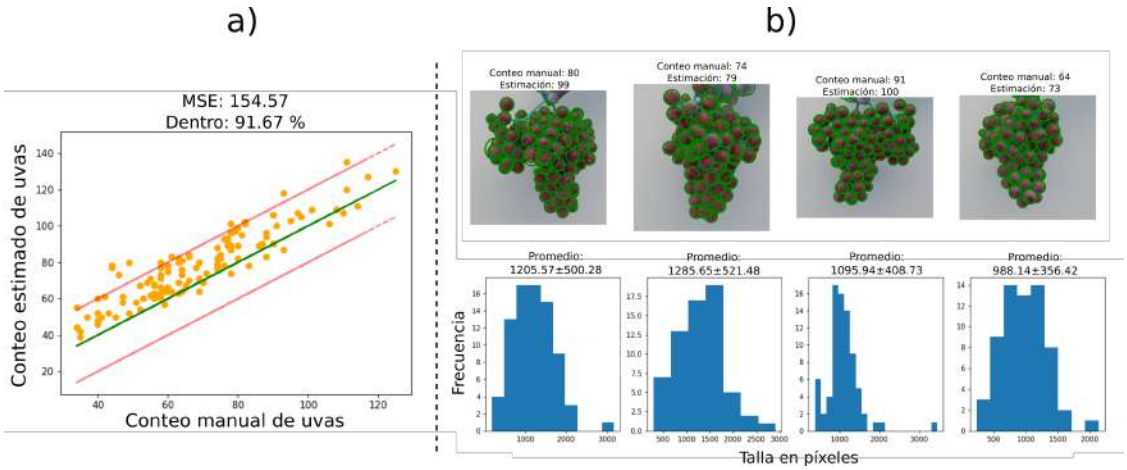


Figura 4.35: Racimos individuales, variedad: **Cabernet Franc**. La imagen (a) presenta un gráfico-R donde el eje horizontal corresponde al conteo manual, mientras que el eje vertical corresponde a la predicción, la línea verde representa el conteo perfecto ($R^2 = 1$), mientras que la línea roja marca la tolerancia para este conteo ideal. El MSE y el porcentaje próximo al conteo ideal son incluidos en la parte superior del gráfico. La imagen (b) presenta cuatro capturas realizadas a un racimo muestra; y sobrepuestos se pueden observar los círculos correspondientes a los arcos detectados para estimar el área oculta. En la parte inferior de cada imagen se incluyó la distribución de tallas en píxeles.

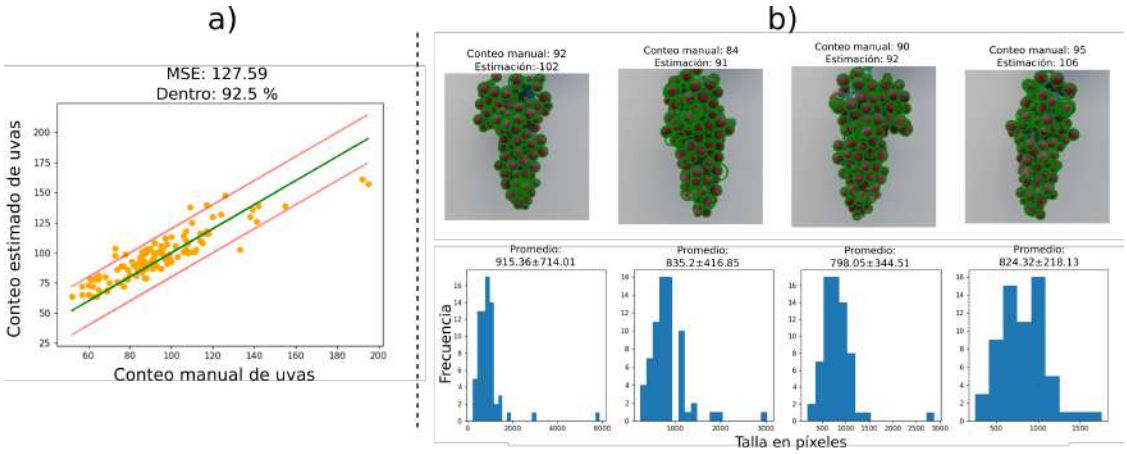


Figura 4.36: Racimos individuales, variedad: **Syrah**. La imagen (a) presenta un gráfico-R donde el eje horizontal corresponde al conteo manual, mientras que el eje vertical corresponde a la predicción, la línea verde representa el conteo perfecto ($R^2 = 1$), mientras que la línea roja marca la tolerancia para este conteo ideal. El MSE y el porcentaje próximo al conteo ideal son incluidos en la parte superior del gráfico. La imagen (b) presenta cuatro capturas realizadas a un racimo muestra; y sobrepuestos se pueden observar los círculos correspondientes a los arcos detectados para estimar el área oculta. En la parte inferior de cada imagen se incluyó la distribución de tallas en píxeles.

Como se mencionó anteriormente, el trabajo de Zabawa et al. [9] reportó un $R^2 = 0.97$, sin embargo, este resultado no se puede interpretar como un desempeño de conteo de 97 %. Además su modelo no fue probado fuera del dominio de las imágenes de entrenamiento, y solo hicieron una evaluación subjetiva con uvas tintas sin medir su desempeño. Por su parte, Millan et al. [13] reportaron una $R^2 = 0.79$ para conteo en imágenes de racimos capturados en campo. Nótese que UNet obtuvo una eficiencia de conteo de 97 % y 100 % para los conjuntos de entrenamiento de CosmeDS y ZabawaDS, respectivamente; mientras que para los de validación se obtuvo 82.5 % y 95 %, respectivamente. Para BIVcolor [45] y Kicherer [51] UNet obtuvo 79.55 % y 50.98 %, respectivamente. Este desempeño superó al reportado por Millan et al. [13]. Para los reportados por Zabawa et al. [9] no se pudo comparar directamente, ya que UNet solo fue evaluada sobre un subconjunto de los datos de Zabawa que si incluyeran racimos en las imágenes.

Por último, se hizo un experimento para determinar, por variedad, una relación lineal entre el peso real en gramos con el área catalogada como "uva" y la estimación de frutos individuales.

4.6.1. Peso de los racimos

Esta sección tiene como objetivos relacionar el número de frutos estimados y el área segmentada con el peso real de cada racimo. Usando las imágenes del experimento anterior, se almacenaron el conteo estimado y la cantidad de píxeles catalogados como "uva". Se utilizaron ocho valores de entrada, los cuales corresponden a las dos cantidades estimadas de las cuatro imágenes por cada racimo.

El experimento consistió en hacer una regresión lineal a las variables de entrada ($\mathbf{x}$) y usar como variable de referencia el peso real de cada racimo (y). El proceso de regresión consiste en encontrar los valores m y b que mejor se ajusten a $\mathbf{x}$ para estimar y (ecuación 4.2). Se hizo la suposición que las variables estimadas en las imágenes se relacionan linealmente con el peso de cada racimo. Es decir que, si se tiene un mayor área y/o conteo de uvas se debe tener un mayor peso estimado (Tabla 4.12). Se observó que la variedad Merlot presenta el peso promedio menor y Syrah el mayor; además, que el peso real es proporcional al número de frutos.

$$y = m \cdot \mathbf{x} + b \tag{4.2}$$

Se entrenaron cinco modelos de regresión lineal para relacionar el peso de los racimos con las variables previamente explicadas. Cuatro de ellos se ajustaron con los datos de cada variedad por separado; mientras que el quinto consistió en un híbrido al incluir todas las variedades para ajustar el modelo de regresión lineal. Para evaluar el desempeño de cada uno, se usó la métrica del error promedio absoluto (MAE) (Tabla 4.13).

Tabla 4.12: Estadística por variedad. El peso fue medido el día de la cosecha, mientras que el conteo se refiere al etiquetado manual en las imágenes.

Medición	**Merlot**	**Cabernet Sauvignon**	**Cabernet Franc**	**Syrah**
Peso (gr)	141.47 ± 46.6	158.53 ± 66.83	203.57 ± 91.45	235.6 ± 82.53
Conteo	61.36 ± 18.71	67 ± 19.61	68.07 ± 19.9	92.88 ± 23.98

Tabla 4.13: Error absoluto promedio al ajustar un modelo de regresión lineal por variedad.

Modelo/Variedad	**Merlot**	**Cabernet Sauvignon**	**Cabernet Franc**	**Syrah**
Merlot	**15.58**	30.02	49.93	33.81
Cabernet Sauvignon	26.99	**22.09**	43.6	32.29
Cabernet Franc	33.06	37.93	**32.81**	60.58
Syrah	35.3	32.86	60.82	**22.41**
Híbrido	23.23	24.01	39.8	30.1

Ya que el MAE no tiene una cota superior, solo se buscó el modelo que logró el valor más bajo. El modelo híbrido fue el que presentó el MAE más bajo, al evaluar todas las variedades al mismo tiempo (Tabla 4.13); con un error promedio de 29.5 gramos. Nótese que el MAE aumenta cuando hay mucha diferencia entre los frutos visibles en la imagen de cada racimo, es decir, si uno de los ángulos de captura oculta muchos de los frutos, el desempeño del modelo de regresión lineal disminuye (Figura 4.38). Se observó en dos capturas de la muestra de Cabernet Franc (Figura 4.39) ocurrió esta situación, debido a la forma del racimo; mientras que en los demás casos el MAE es menor (figuras 4.37 y 4.40).

Figura 4.37: Predicción del peso sobre la variedad **Merlot**. Se muestran cuatro capturas sobre el mismo racimo, en donde se incluye el conteo manual y el conteo de frutos estimado; también el peso real y el peso estimado por el modelo de regresión lineal. Por otro lado, se incluyó un rectángulo verde con las medidas en píxeles de alto y ancho.

Figura 4.38: Predicción del peso sobre la variedad **Cabernet Sauvignon**. Se muestran cuatro capturas sobre el mismo racimo, en donde se incluye el conteo manual y el conteo de frutos estimado; también el peso real y el peso estimado por el modelo de regresión lineal. Por otro lado, se incluyó un rectángulo verde con las medidas en píxeles de alto y ancho.

Figura 4.39: Predicción del peso sobre la variedad **Cabernet Franc**. Se muestran cuatro capturas sobre el mismo racimo, en donde se incluye el conteo manual y el conteo de frutos estimado; también el peso real y el peso estimado por el modelo de regresión lineal. Por otro lado, se incluyó un rectángulo verde con las medidas en píxeles de alto y ancho.

Figura 4.40: Predicción del peso sobre la variedad **Syrah**. Se muestran cuatro capturas sobre el mismo racimo, en donde se incluye el conteo manual y el conteo de frutos estimado; también el peso real y el peso estimado por el modelo de regresión lineal. Por otro lado, se incluyó un rectángulo verde con las medidas en píxeles de alto y ancho.

Por un lado, Nuske et al. [17] estimaron el peso de los racimos detectados basado en el conteo de frutos y obtuvieron una $R^2 = 0.74$ al comparar el conteo automático contra el peso real en libras. Además, con un modelo de regresión lineal estimaron el peso de los frutos en 8 surcos de siembra con un error de 9.8 %. Por otro lado, Millan et al. [13] obtuvieron un $R^2 = 0.81$ con un RMSE (raíz del MSE) de 310.2 gramos.

Para poder realizar una comparación directa con lo reportado en el trabajo de Nuske et al. [17] se calculó el porcentaje de error del modelo de regresión lineal, explicado anteriormente, con la siguiente ecuación:

$$\% \text{ Error} = \frac{|\text{Real} - \text{Predicción}|}{\text{Real}} \cdot 100 \tag{4.3}$$

y además, se calculó el promedio de todas las variedades, obteniendo un error de 5.65 %. Notar que estos resultados no son directamente comparables, pero da una idea de qué tan acertada es la predicción del modelo de regresión lineal evaluado en esta investigación. Para la comparación con Millan et al. [13], se obtuvo un MAE promedio de 29.5 gramos en todas las variedades, mientras que ellos un RMSE promedio 310.2 gramos; la diferencia principal es que el RMSE siempre será mayor al MAE cuando se evalúan las mismas muestras. Si se traduce el MAE obtenido por el modelo de regresión lineal a RMSE sería de 37.51 gramos, significativamente menor que el reportado por Millan et al. [13].

Para finalizar con este trabajo, se presenta un marco de referencia integrado en una aplicación web.

4.7. Marco de referencia

Para concluir esta investigación se desarrollaron dos aplicaciones web, las cuales contienen un demo de los algoritmos de procesamiento de imágenes descritos en este documento. El primero abarcó los algoritmos de segmentación no supervisada, mientras que el segundo es una herramienta que puede ser usada para analizar imágenes de racimos de uvas.

4.7.1. Segmentación no supervisada

La primer aplicación web incluye dos variantes para reconocer regiones de interés. Una se basa solamente en índices de color, mientras que la otra solo en espacios de color. Nótese que este demo tiene la capacidad de analizar imágenes de cualquier tipo sin importar su dominio (Figura 4.41). El motor de procesamiento que utiliza esta aplicación web incluye los algoritmos de agrupamiento: K-medias y FC-medias. Esta aplicación se encuentra disponible en el siguiente enlace: https://ricglez90-unsupervised-segmentation-deploy-streamlit-r955qd.streamlitapp.com/

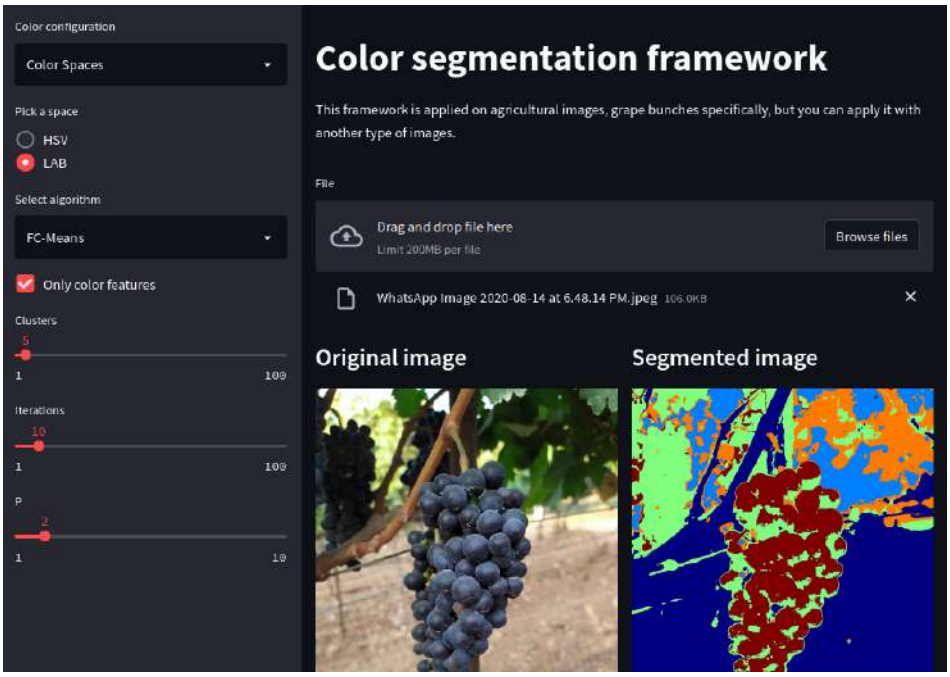


Figura 4.41: Captura de pantalla del demo Color segmentation framework.

4.7.2. Análisis de imágenes de racimos de uvas

Esta aplicación web consta de tres secciones principales: Localización del racimo, racimo individual y estimación del peso. La primera usa como motor de procesamiento el modelo basado en bosque aleatorio (Figura 4.42); la segunda usa el modelo de aprendizaje profundo y permite hacer un análisis por fruto a nivel de color y talla (Figura 4.43); mientras que la tercera requiere de entrada cuatro imágenes que correspondan al mismo racimo tal como se explicó en la sección 4.6.1 (Figura 4.44). Esta herramienta se puede encontrar en el siguiente enlace: `http://148.231.215.200:8501/`.

La primer ventana mostrada en la Figura 4.42, tiene como objetivo localizar el área que corresponde a los racimos y despliega el tamaño en píxeles que corresponden a esta área. La segunda, el localizar los frutos individuales, mostrar la distribución de tallas y desplegar en el espacio CIELab el color medido en cada fruto. La tercer ventana despliega una muestra para predecir el peso (Figura 4.44). Además se incluyó, un conjunto de imágenes muestra[8], capturadas con un teléfono inteligente, también en San Cosme Viñedos.

[8]https://github.com/ricglez90/grape_learning/tree/main/Images

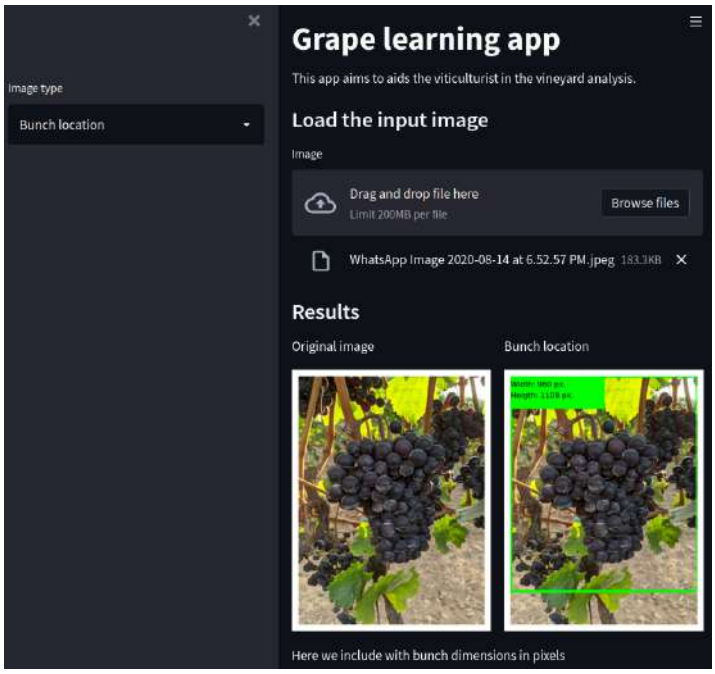


Figura 4.42: Captura de pantalla de la aplicación web Grape Learning: localización del racimo.

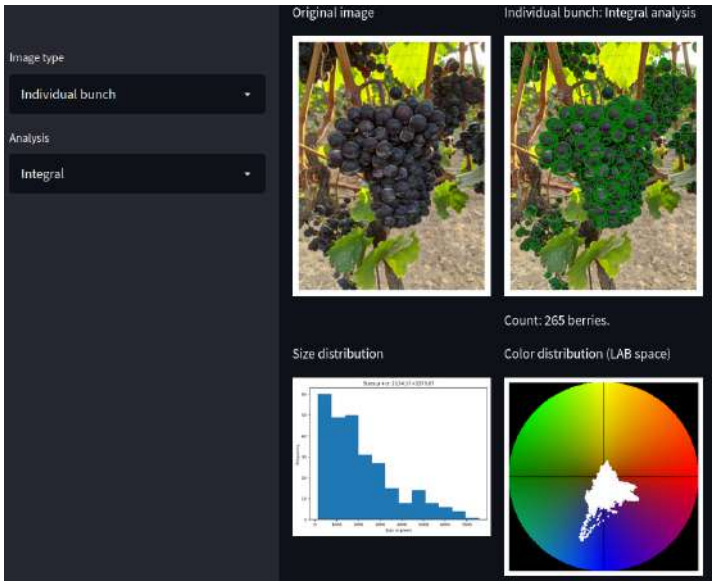


Figura 4.43: Captura de pantalla de la aplicación web Grape Learning sección: racimo individual.

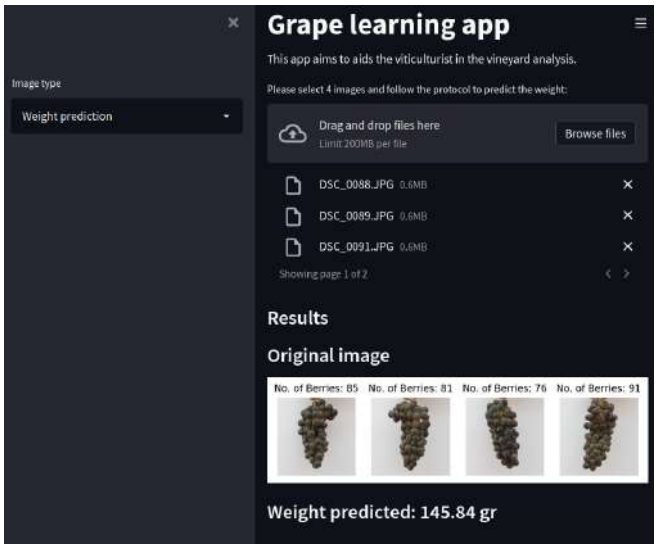


Figura 4.44: Captura de pantalla de la aplicación web Grape Learning sección: predicción del peso.

4.8. Limitaciones

Como cualquier trabajo de investigación, este también presenta limitaciones y se listan a continuación:

1. Robustez limitada ante uvas blancas: aunque se agregaron imágenes con uvas blancas al entrenamiento, estás se encontraban en una etapa de maduración temprana. Al hacer las pruebas con el conjunto Kicherer [51], se presentó una mejora en la detección, pero se requieren hacer más pruebas. Es importante recordar que cuando se hicieron las capturas, solo se tuvo acceso a variedades tintas.

2. Predicción del peso: aunque el modelo de regresión lineal estima el peso con un margen de error aceptable, este se vio afectado por la forma de los racimos y la cantidad de uvas ocultas en la captura.

4.9. Sumario

Para intentar resolver el problema núcleo de esta investigación se recorrió el siguiente camino:

1. Se hizo uso de la **segmentación no supervisada** y las características de color para obtener resultados preliminares de segmentación, sin embargo, este enfoque resultó en una segmentación de bajo desempeño y por tal motivo se decidió usar aprendizaje supervisado para abordar esta tarea.

2. La **segmentación supervisada**, es decir, un enfoque píxel-a-píxel consistió en asignar a cada píxel una clase y en este se propuso usar un vector de 22 características de color. Este tipo de enfoque es usado en [42] y [13]; en el primero reportaron un F1-Score de 0.89 para segmentación de sus imágenes, mientras que, el segundo no reportaron sus resultados de segmentación. En esta etapa se evaluaron cinco clasificadores y el que presentó el mejor desempeño fue el algoritmo bosque aleatorio (RF), resultando en un F1-Score promedio de **0.91** para el conjunto de CosmeDS y superando los reportados en [42], aunque no son directamente comparables, ya que se usaron diferentes conjuntos de datos para su evaluación. Además, se incluyó un análisis subjetivo al probar el modelo en el conjunto de BIVcolor [45]. Este enfoque solo definió el área que corresponde al racimo dentro de la imagen y no nos indicó el número de frutos contenidos dentro de cada imagen evaluada.

3. Para predecir el **número de frutos** existen diversas propuestas, algunas de ellas toman ventaja de la forma circular de las uvas. Por un lado, Millan et al. [13] propusieron una estimación basada en el área segmentada como "racimo", esta depende de catalogar correctamente la región de interés, y además de conocer el área circular promedio de las uvas; ellos reportaron un 80 % de eficiencia en el conteo, sin embargo, cuando se aplicó su método a imágenes segmentadas por el modelo basado en RF, se obtuvo una eficiencia de 31.68 % para el conjunto de CosmeDS. Esta diferencia se debió a que ellos usaron imágenes de entrada con fondo homogéneo detrás del racimo, mientras que RondoDS y CosmeDS solo incluyeron imágenes de campo donde el fondo de la escena es heterogéneo; concluyendo que este método no es adecuado para este tipo de imágenes. Por otro lado, Rudolph et al. [37] aplicaron detección de círculos y obtuvieron una eficiencia de conteo del 93 %, pero usan imágenes de uvas capturadas en etapa temprana de crecimiento y ese porcentaje reportado es de una muestra aleatoria de 10 imágenes de las 108 que tienen a disposición. Adicionalmente, se compararon tres detectores de círculos, pero su desempeño para localizar uvas dentro de imágenes digitales fue más bajo que el método de Millan et al. [13], y por esta razón se determinó no usar este enfoque. Cabe mencionar que se entrenaron seis algoritmos de clasificación que servirían para catalogar las sub-imágenes extraídas por los círculos detectados. Estos modelos fueron entrenados con las uvas anotadas manualmente en el conjunto de RondoDS, aquí de nueva cuenta RF fue el que presentó el mejor desempeño. Ya que el enfoque píxel-a-píxel solo sirvió para segmentar racimos y la detección de círculos tuvo un desempeño bajo, se decidió explorar un enfoque de aprendizaje profundo (DL).

4. **El enfoque de DL**, es el más complejo que esta investigación abordó. Entrenando a UNet, una arquitectura de CNN que aborda la tarea segmentación, con el conjunto CosmeDS, se logró alcanzar un F1-Score para la clase "uva" de 0.89.

Este resultado es mayor en comparación al reportado por Zabawa et al. [9] de 0.75 también para la clase "uva". El conjunto de CosmeDS incluyó solamente imágenes de uvas tintas, mientras que el de Zabawa [47] incluye mayormente uvas blancas. Al combinar ambos conjuntos de datos (CosmeDS + ZabawaDS) en el entrenamiento de UNet, se logró aumentar el conteo al evaluar sobre el conjunto de Kicherer [51] de 19.61 % a 50.98 %; recordar que este último conjunto solo incluyó una variedad de uvas blancas (Riesling). La predicción de UNet arrojó como resultado cada fruto segmentado y, en conjunto con un método de predicción de área ocluida, se pudo extraer datos estadísticos de cada racimo y de los frutos individuales, tales como la talla y su distribución.

5. **La estimación del peso**. Este experimento se hizo sobre un conjunto especial de imágenes. Este incluyó cuatro ángulos distintos por cada racimo. Calculando el área en píxeles y la cantidad de frutos de cada imagen, se entrenó un modelo de regresión lineal con esos valores y se estimó el peso en gramos. Al evaluarlo se obtuvo un MAE (error) promedio de 29.5 gramos, sobre todos las variedades y esto corresponde a 5.65 % de error. En el trabajo de Nuske et al. [17] reportaron un error promedio de 9.8 %, sin embargo, ellos lo miden basado en las diferencias de su estimación de conteo en cada surco que evalúan, por lo que sus resultados y los obtenidos del modelo de regresión lineal de esta investigación, no son directamente comparables. La propuesta de esta investigación, también consintió en usar un modelo de regresión lineal para estimar el peso, pero incluyó además del conteo de frutos, el valor en píxeles del área segmentada como racimo.

6. **Marco de referencia**. Se diseñaron dos aplicaciones web para procesar imágenes de racimos de uvas. La primera usa como motor de procesamiento los algoritmos de aprendizaje no supervisado y puede usarse con cualquier tipo de imágenes como entrada ya que solo cataloga píxeles basado en características de color. La segunda es específica para imágenes de racimos de uva y usa los algoritmos de ML tradicional (enfoque píxel-a-píxel) y los de DL (UNet) como motor para predecir del racimo, el área de los frutos y el conteo. Además, se incluyó un gráfico de dispersión del espacio de color CIELab para describir a los frutos en cuestión.

Como lo menciona Nuske et al. [17] la predicción del rendimiento en un viñedo depende de tres variables: el número de racimos por planta, el número de frutos por racimo y el tamaño de las uvas. Siendo las últimas dos las de mayor importancia para esta investigación; la primera es parcialmente resuelta con un enfoque de DL, mientras que la segunda se resuelve parcialmente con la etapa de detección de arcos, la cual estima el área ocluida en los frutos. Además, se tienen resultados preliminares con respecto a la estimación del peso, con un error del 5.65 %. En el siguiente capítulo se presentan las conclusiones y unas ideas para trabajo a futuro que se propone para esta investigación.

Capítulo 5

Conclusiones

Este capítulo presenta las conclusiones a las que se llegaron al final de esta investigación, se incluye también un resumen de las aportaciones puntuales de este trabajo. Se cierra con perspectivas de investigación derivadas de este proyecto.

5.1. Conclusiones y aportaciones

5.1.1. Identificación de racimos y estimación de frutos

El primer objetivo fue identificar si los píxeles contenidos en una imagen correspondían a dos categorías establecidas: "fondo" o "racimo". Se concluyó que, de los cinco algoritmos evaluados, el basado en bosque aleatorio (RF) presentó el mejor desempeño en la tarea de identificar píxeles que corresponden a la categoría "racimo". Además, la configuración de características de color fue la mejor configuración de entrada, para identificar racimos de uvas tintas. Se observó también, que el desempeño del modelo RF disminuye cuando se evalúan imágenes que contienen frutos con coloración distinta a las de entrenamiento, como las que contiene el conjunto BIVcolor [45]. Después de una inspección visual de los resultados del modelo RF sobre este conjunto, se observó que en el 20 % de las imágenes, dicho modelo clasificó incorrectamente los píxeles evaluados.

Para el objetivo de estimar la cantidad de frutos en una imagen, se observó que el método evaluado es sensible a píxeles mal categorizados, ya que no incluye una manera de manejarlos y esto resulta en un sobre-conteo en cada imagen. Debido a estos resultados se optó por explorar un enfoque de detección de círculos como herramienta principal para el conteo.

5.1.2. El problema de localizar frutos individuales

El siguiente objetivo, fue evaluar si las uvas individuales contenidas en una imagen podían localizarse como círculos. Además de evaluar si estos círculos realmente

contenían una uva.

Se propuso un detector de círculos para imágenes genéricas [67], el cuál superó a dos detectores evaluados bajo dos escenarios: imágenes sintéticas y dibujadas a mano; estos detectores son ampliamente utilizados en la literatura: EDCirles [66] y la transformada de Hough para círculos (CHT). Después, se evaluaron estos tres detectores en el dominio de imágenes de racimos de uvas. Se hizo una inspección visual sobre los resultados de detección, y se observó que solo EDCircles [66] identificó a las uvas sin oclusión como círculos, mientras que el resto de las uvas que no fueron detectadas. Esto hace suponer, que mientras persista mayor aglomeración de uvas dentro del racimo, el desempeño de un detector de círculos disminuye. Aunque Rudolph et al. [37] usaron la CHT y reportaron un 93 % de eficiencia de conteo, este se obtuvo en imágenes de racimos en una etapa temprana de crecimiento, es decir, cuando la aglomeración de frutos está menos presente. Por un lado, la aglomeración de las uvas presenta un reto al detectarlas como círculos, ya que los bordes de los frutos que forman un círculo se pierden. Por otro lado, los tres métodos evaluados, requieren bordes bien definidos para detectar correctamente los círculos, por lo que, si la imagen de entrada presenta bordes difuminados o con ruido esto disminuirá el desempeño de detección. Con base en el pobre desempeño obtenido de los detectores de círculos no se profundizó más en este enfoque.

De manera paralela y para determinar si los círculos detectados eran uvas, se evaluaron seis algoritmos de clasificación y de nueva cuenta, el clasificador bosque aleatorio (RF), fue el mejor entre ellos. Las características usadas para evaluar las imágenes fueron: textura (LBP), gradientes (HOG) y el color (FCH). Perez-Zavala et al. [12] propusieron un esquema similar, pero usaron una máquina de vectores de soporte (SVM) como clasificador y no incluyeron características de color. Ellos obtuvieron una sensibilidad y precisión de 0.8 y 0.88, respectivamente, evaluado sobre su conjunto de datos; mientras que el clasificador RF, aquí estudiado, alcanzó una exactitud promedio de **0.94** sobre las sub-imágenes de RondoDS. Ya que no se tuvo acceso a sus datos no se pudo realizar una comparación directa.

5.1.3. La elección del enfoque de aprendizaje de máquina

Basados en los resultados de esta investigación, se observó que el enfoque de aprendizaje profundo (DL) es el que mejor se desempeñó al segmentar frutos individuales en imágenes de racimos de uvas. En este trabajo se evaluaron las siguientes arquitecturas: Zabnet [9], Segnet [58] y UNet [56]. La red convolucional UNet presentó el mejor desempeño en la tarea de segmentar frutos individuales, con un F1-Score de **0.92** promedio en los píxeles de la clase "uva", superando a Segnet y Zabnet con un F1-Score de 0.87 y 0.88, respectivamente. Además, UNet presentó el porcentaje de conteo más alto con **98.77 %**, mientras que Segnet y Zabnet lograron solo 67.73 % y 46.91 %, respectivamente.

Se demostró que hacer transferencia de conocimiento en DL, en el problema parti-

cular de segmentación de frutos en racimos, genera un desempeño mayor que entrenar desde cero. Se obtuvo un aumento en la exactitud de 5 % y 3 % al evaluarlo con las particiones de entrenamiento y validación, respectivamente. Esto es consistente con lo reportado por Yosinski et al. [89]. Además, al incluir sobre-muestreo de datos, se obtuvo un incremento adicional de 5 % y 2 % a la exactitud de la partición de entrenamiento y validación, respectivamente.

Por último, se demostró que incluir mayor variabilidad en el conjunto de entrenamiento genera un desempeño de conteo más alto. La inclusión de imágenes de uvas blancas (ZabawaDS [47]) en el entrenamiento de UNet produjo un incremento en el porcentaje de conteo, pasando del 19.61 % al 50.98 %, al evaluar sobre un conjunto de datos que solo incluye uvas blancas (Kicherer [51]).

5.1.4. Un modelo para la estimación del peso

Se propuso el uso de un modelo de regresión lineal para estimar el peso en racimos individuales, y este arrojó un error promedio de 29.5 gramos por variedad de uvas, esto se traduce a un error de 5.65 %. Los resultados son una mejora en comparación a los de Nuske et al. [17] que obtuvieron un error de 9.8 %, sin embargo, estos resultados no son directamente comparables, ya que ellos calculan el peso en todo un surco y en esta investigación se hizo por racimo. A pesar de la mejoría observada, el esquema propuesto para la estimación del peso en imágenes de racimos, requiere ser evaluada en su capacidad de generalización.

5.2. Contribuciones

Con base en los resultados descritos previamente se concluyó que se cumplieron los objetivos planteados inicialmente. Como resultado de esta investigación se obtuvieron las siguientes aportaciones:

1. Se crearon dos bancos de imágenes de racimos de uvas, que se suman a los ya disponibles en la literatura, para el entrenamiento de algoritmos de aprendizaje de máquina.

2. Se propuso un vector de características basado en espacios e índices de color [97] junto con un algoritmo de clasificación, que localizaron de manera eficiente los píxeles que corresponden a uvas en imágenes de uvas tintas.

3. Se propuso un método para detectar círculos en imágenes genéricas [67] y se evaluó su desempeño en imágenes de racimos de uva.

4. Se determinó que un esquema de aprendizaje profundo permite localizar y contar el número de frutos dentro de imágenes de racimos de uvas, de manera más eficiente, comparado con un esquema clásico.

5. Se propuso una fase post-procesamiento de imágenes que permite estimar el área de uvas parcialmente ocluidas por medio de la detección de arcos.

6. Se estimó la relación lineal entre las variables calculadas por un algoritmo de visión por computadora y el peso real de un racimo. Esto se hizo por cada variedad de uva presentada en esta investigación.

Además se desarrollaron dos aplicaciones web potenciadas con algoritmos de aprendizaje de máquina. La primera reconoce regiones de interés dentro de imágenes, analizando el color de las mismas y admite como entrada cualquier tipo de imágenes. Mientras que la segunda, es una herramienta para el análisis de imágenes de racimos individuales, y arroja resultados de la localización, talla y color tanto de los racimos como de cada fruto. En lo siguiente se propone el trabajo a futuro que pudiera dar continuación a esta investigación.

5.3. Trabajo a futuro

1. Incluir imágenes de variedades de uvas blancas para balancear el conjunto de datos a fin de alcanzar la característica de representatividad que debe tener un conjunto de datos. Incrementar el número de elementos etiquetados en CosmeDS.

2. Utilizar imágenes multi/hiperespectrales en vez de usar un vector de características basado en el color.

3. Sintonizar el detector de círculos propuesto al problema específico. Lo que se propuso en [67], se puede sintonizar para detectar círculos incompletos y agregar una fase de segmentos de arco continuos que pudiera servir para detectar uvas parcialmente ocluidas. Además, se propone optimizar la implementación *in-house* que se hizo para este método.

4. Evaluar el desempeño de los detectores de círculos en imágenes de racimos de uvas en etapas tempranas de crecimiento.

5. Destilar[1] a UNet a una versión más ligera *i.e.* capas separables en profundidad, a fin de poderlo llevar a una arquitectura de *hardware* más limitada.

[1]En el contexto de redes neuronales es transferir de una red con N parámetros a una con M, tal que $M \ll N$.

6. Incluir el procedimiento de detección de área ocluida en la etapa de aprendizaje. Etiquetar a CosmeDS incluyendo el porcentaje de oclusión de cada uva y entrenar a UNet con este conjunto.

7. Probar el esquema de predicción de peso en nuevas capturas para replicar o mejorar resultados. Además incluir más características u otro modelo más complejo.

5.4. Productos derivados de esta investigación

De esta investigación se desprenden las siguientes publicaciones:

1. Trabajo presentado en la ROPEC 2019[2]: González-Márquez, M. R., Brizuela, C. A., Martínez-Rosas, M. E., Cervantes, H. (2020, November). Grape Bunch Detection Using A Pixel-Wise Classification In Image Processing. In 2020 IEEE International Autumn Meeting on Power, Electronics and Computing (ROPEC) (Vol. 4, pp. 1-6). IEEE. Se incluyó en él, una parte de los resultados explicados de la sección 4.3.

2. Trabajo publicado en la revista PAAA[3]: González, M. R., Martínez, M. E., Cosío-León, M., Cervantes, H., Brizuela, C. A. (2021). Multiple circle detection in images: a simple evolutionary algorithm approach and a new benchmark of images. Pattern Analysis and Applications, 24(4), 1583-1603. Se propuso un detector de círculos para imágenes genéricas.

3. Artículo de arquitecturas: González-Márquez, M. R., Brizuela, C. A., Martínez-Rosas, M. E. "*Deep learning for berry segmentation: comparison of convolutional network architectures in the single grape detection problem*". En preparación.

[2]https://2019.ropec.org/, accesado el 31 de Julio de 2022.
[3]https://www.springer.com/journal/10044, accesado el 31 de Julio de 2022.

Apéndice A

Conceptos

A.1. Representaciones de las imágenes

Una imagen digital es una proyección 2D capturada por un dispositivo (Figura A.1). Esta proyección es extraída de una escena originalmente en 3D y de manera formal se representa en forma matricial [18]. Sea $\mathbf{A}$ una matriz de imagen formada por píxeles en posiciones discretas (i, j); donde i se refiere al renglón y j a la columna; $\mathbf{A}$ es muestreada a un espacio finito con M renglones y N columnas y cuantificada en k valores, es decir, los valores de $\mathbf{A}(i, j) \in \{0, k-1\}$. El valor de k dependerá de la sensibilidad del sensor de captura y es conocido como la profundidad de *luminiscencia*.

Es necesario transformar una imagen digital a una representación en menor dimensión, pero conservando las características principales dentro de la escena capturada. En lo siguiente se explica cómo llevar a cabo este proceso.

A.1.1. Extracción de características

El color es una característica distintiva en frutos y vegetales, y generalmente, un consumidor define la calidad relativa por la apariencia de los frutos antes de consumirlos [101]. Estas características pueden ser: tamaño, forma y por su puesto el color. Para esta última, cambios detectados en la uniformidad podrían sugerir signos de enfermedades. Así como un consumidor elige los frutos que llevará al hogar, por sus características visuales, el problema de esta investigación puede ser abordado para distinguir entre frutos y el resto de la escena dentro de una imagen digital.

El color es un indicador de madurez y se relaciona con el contenido químico de los frutos pero ¿cómo medimos el color en una imagen digital? Para entender esto, primero se requiere tener una estructura que lo represente. Como se mencionó hace un momento, una imagen puede ser representada por una matriz en 2D, sea $\mathbf{A} \rightarrow \mathbb{Z}^{M \times N}$ una matriz de imagen. Esta tiene un solo valor por píxel y representa solo los cambios de luz de la escena capturada [18]. Sin embargo, se trabaja generalmente con imágenes a color que

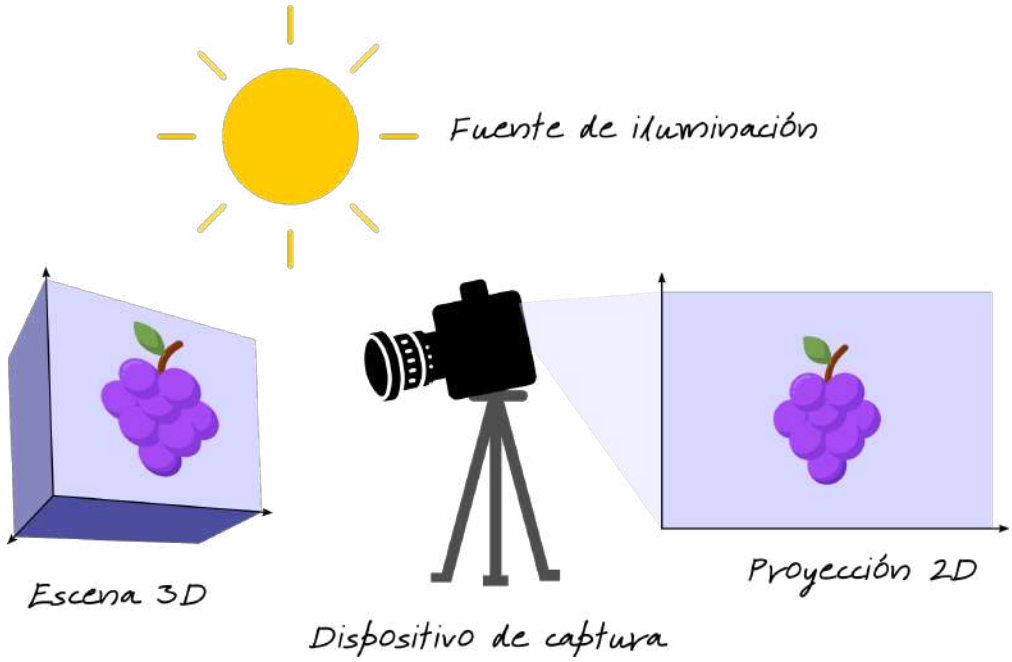


Figura A.1: Ilustración de la formación de imágenes digitales.

son representadas por una matriz 3D, sea $\mathbf{A} \rightarrow \mathbb{Z}^{M \times N \times C}$ una matriz de imagen para representar el color; donde C representa cada canal en algún espacio de color. Antes de explicar a este tema es pertinente conocer ¿cómo se capta el color?.

De manera formal, el color es una consecuencia de tres factores: una fuente de luz, un objeto que refleja una parte de esa luz y un sistema de visión que la capta. La fuente de iluminación puede ser natural o artificial; el objeto de interés provoca tres fenómenos sobre la luz: absorción, reflexión y transmisión; y por último el sistema de visión que capta esa luz reflejada, pueden ser un sistema de visión natural o artificial.

El color visible captado por el ojo humano va desde los 400 hasta los 750 nm de longitud de onda (λ) y por esa razón la mayoría de los sensores digitales están diseñados para captar estas longitudes de onda [18]. Sin embargo, la visión humana capta tres componentes principales de λ: rojo (700 nm), verde (546.1 nm) y azul (435.8 nm), y con la combinación de ellos se forman el resto de los colores (Figura A.2). Nótese que los sistemas de visión son más sensibles al componente verde, que al resto. De la misma manera, los sensores de la mayoría de las cámaras digitales captan la luz de estos mismos componentes de color.

Ya que es importante representar al color en un modelo matemático, en lo siguiente se explican algunos de los espacios de color desarrollados a través de la historia.

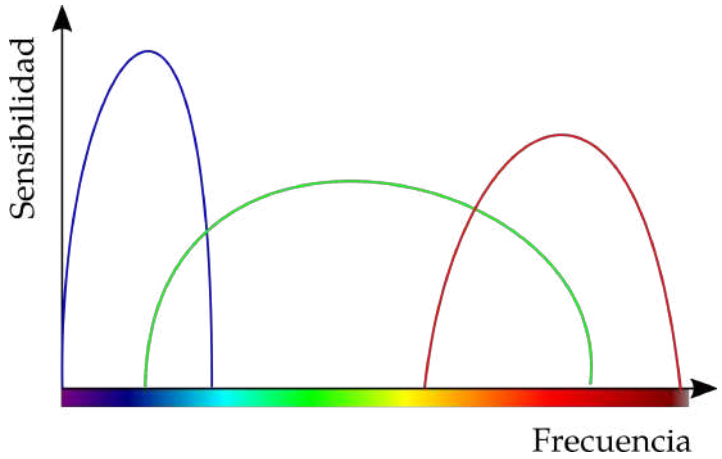


Figura A.2: Curvas de sensibilidad espectral. Los componentes RGB captados por un foto-sensor: azul (izquierda), verde (centro) y rojo (derecha). Figura tomada de [18].

A.1.2. Espacios de color

Matemáticamente el color es modelado por dos componentes principales: cromaticidad e intensidad. El primero corresponde al color puro, es decir, cuando hablamos del verde, rojo o azul; mientras que el segundo, a la claridad del color (Figura A.3).

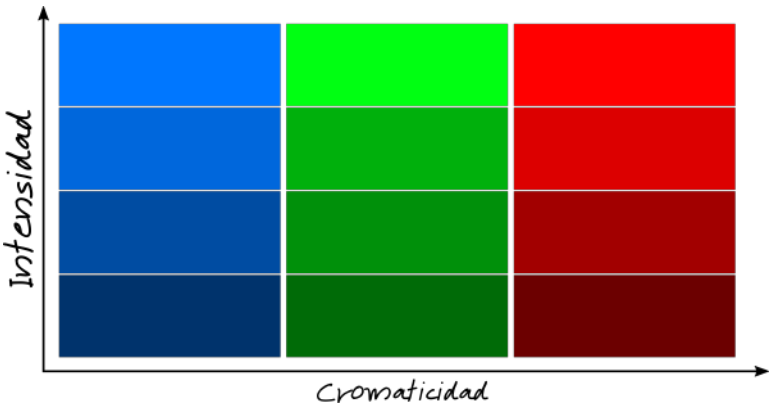


Figura A.3: Componentes básicos del color. Ilustración de cómo se modela el color para representarlo de manera computacional [86].

El objetivo de loa espacios es facilitar su representación dentro de espacio 3D, donde cada dimensión corresponde a un único punto dentro del espacio [86]. Pero primero se requiere entender la representación más simple presente en imágenes digitales.

Espacio RGB

Este es el espacio más simple y está compuesto por tres componentes básicos: rojo (R), verde (G) y azul (B); tiene forma de cubo –un eje para cada componente– y se muestra en un intervalo dinámico de $[0, k]$ (Figura A.4).

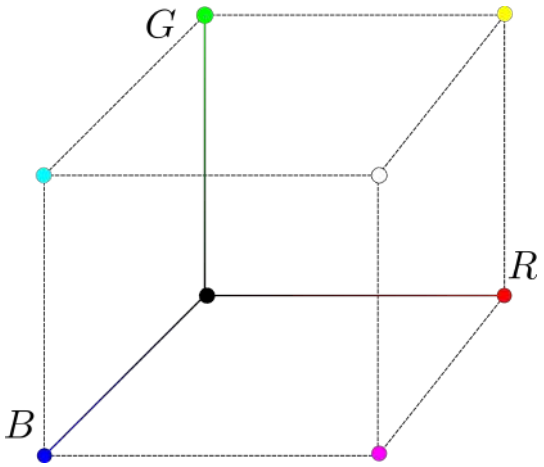


Figura A.4: Espacio RGB. Ilustración de la forma geométrica del espacio RBG: rojo, verde y azul [86]. El complemento de cada componente también es incluido en cada ilustración: cían, magenta y amarillo, respectivamente.

El espacio RGB no tiene la capacidad de separar el color en sus dos componentes principales (cromaticidad e intensidad), ya que sus componentes están altamente relacionados entre ellos. Para lograr esto es necesario transformar a alguno de las siguientes representaciones.

Espacio HSV

Este espacio es representado por tres componentes: el matiz (H, *hue* en inglés), la saturación (S) y la intensidad (V). La cromaticidad es representada por H y S, mientras que la intensidad por V. A diferencia de RGB, la forma geométrica de HSV es cónica (Figura A.5), y en consecuencia se modela en coordenadas polares.

El intervalo dinámico para cada componente es de la siguiente manera:

- Matiz: $H : [0, 2\pi] \in \mathbb{R}$
- Saturación: $S : [0, 1] \in \mathbb{R}$
- Intensidad: $V : [0, k] \in \mathbb{Z}$

Una de las propiedades que permite visualizar el espacio HSV es el cambio color a través de un ángulo. Para pasar del espacio RGB a HSV seguimos los paso mostrados en el Algoritmo 1.

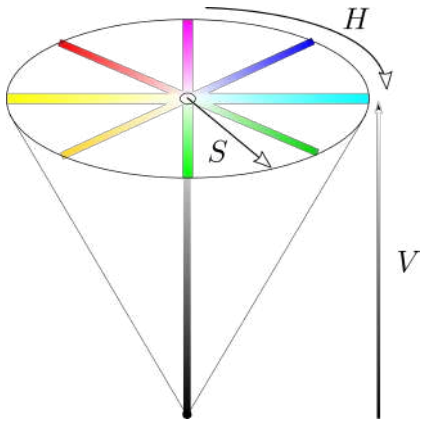


Figura A.5: Espacio HSV. Ilustración de la forma geométrica del espacio HSV: matiz (H, *hue* en inglés), saturación (S) e intensidad (V) [86].

Algoritmo 1 Proceso para pasar del espacio RBG a HSV

Datos: r, g, b
Resultado: h, s, v
si $r = g = b$ **entonces**
| θ se indetermina
en otro caso
| $\theta = \cos^{-1}\left(\frac{(r-g)+(r-b)}{2\sqrt{(r-g)^2+(r-b)(g-b)}}\right)$
| **si** $b \leq g$ **entonces**
| | $h = \theta$
| **en otro caso**
| | $h = 2\pi - \theta$
| **fin**
fin
si $\text{máx}(r, g, b) = 0$ **entonces**
| $s = 0$
en otro caso
| $s = 1 - \frac{\text{mín}(r,g,b)}{\text{máx}(r,g,b)}$
fin
$v = \text{máx}(r, g, b)$

Donde r, g y b son los valores de cada píxel en los componentes R, G y B, respectivamente. Notar que el espacio HSV no tiene manera de manejar los grises, es decir, cuando $r = g = b$ y el componente h se indetermina para ellos (Figura A.6).

Este no es el único espacio donde se puede representar al color en 3D, a continuación se describen otros espacios.

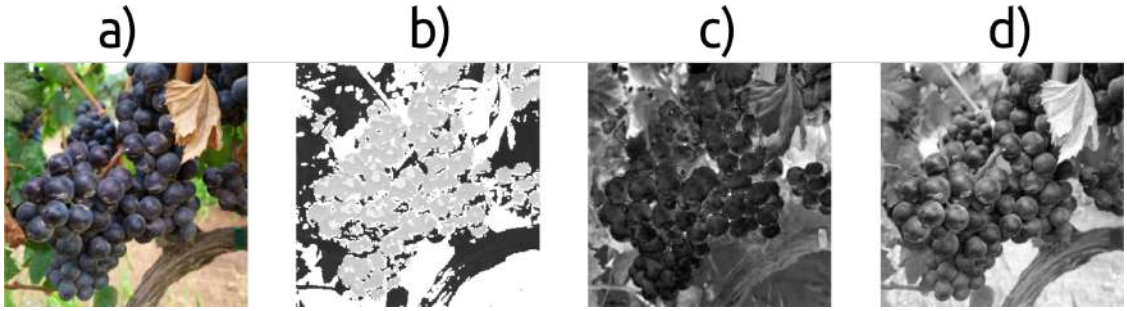


Figura A.6: Visualización del espacio HSV. Captura en Rondo del Valle (a) mostrada en el espacio RGB; (b) componente de matiz H, (c) componente de saturación S y (d) componente de intensidad V. Los últimos tres presentados en escala de grises.

Espacio HSI

Este espacio es similar a HSV, la diferencia radica en el cálculo del componente S, la intensidad tenemos un componente I (en vez de V) y la forma geométrica de este espacio son dos conos HSV superpuestos como se aprecia en la Figura A.7.

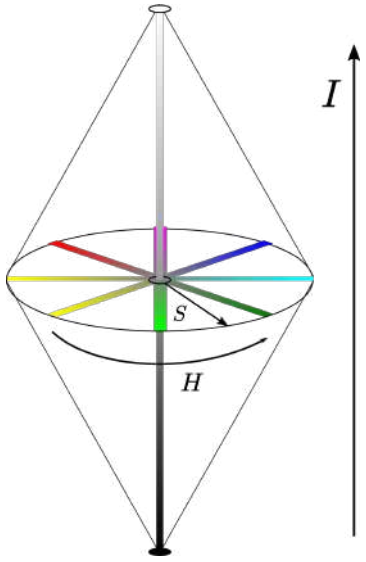


Figura A.7: Espacio HSI. Ilustración de la forma geométrica del espacio HSI: matiz, saturación e intensidad [86].

El cálculo para el componente H es idéntico a HSV al igual que sus correspondientes intervalos dinámicos. El componente S se calcula de la siguiente forma:

$$s = 1 - \frac{\text{mín}(r, g, b)}{i} \tag{A.1}$$

y para calcular el componente I se utiliza la siguiente ecuación:

$$i = \frac{r + g + b}{3} \tag{A.2}$$

Nótese que a diferencia de HSV, este espacio calcula la intensidad como un promedio y no como el valor máximo entre los canales de RGB. Una muestra de cada componete del espacio HSI se puede apreciar en la Figura A.8.

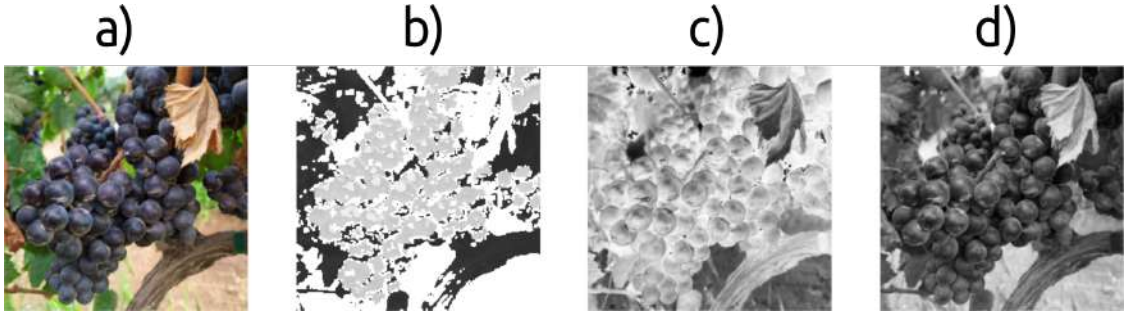


Figura A.8: Visualización del espacio HSI. Captura en Rondo del Valle (a) mostrada en el espacio RGB; (b) componente de matiz H, (c) componente de saturación S y (d) componente de intensidad I. Los últimos tres presentados en escala de grises.

Los siguientes espacios de color son una variante del modelo estándar XYZ. Este modelo fue desarrollado por la Comisión Internacional de la Iluminación (CIE, por su sigla en francés *Comission Internationale de I'Eclairage*) con el objetivo de reproducir el color real y que sea independiente al sensor de captura, o a la pantalla de despliegue. Según la descripción de [86] estos espacios imitan la percepción del color de los humanos, y por lo tanto, son ampliamente usados en procesamiento de imágenes.

Espacio CIELab

Este espacio define la intensidad del color por su componente L^* y la cromaticidad por sus componentes a^* y b^*. CIELab es representado geométricamente en una esfera (Figura A.9).

A pesar de la forma, sus componentes no se expresan en coordenadas polares. Nótese que el eje a va de un verde a rojo, mientras que el eje b de azul a amarillo. Los intervalos dinámicos se establecen de la siguiente manera:

- Intensidad $L^* : [0, 100] \in \mathbb{R}$
- Cromaticidad a^* y $b^* : [-127, 128] \in \mathbb{R}$

Para transformar los datos del espacio RBG a CIELab se hace con las siguientes ecuaciones:

$$L^* = 116 f\left(\frac{Y}{Y_{ref}}\right) - 16 \tag{A.3}$$

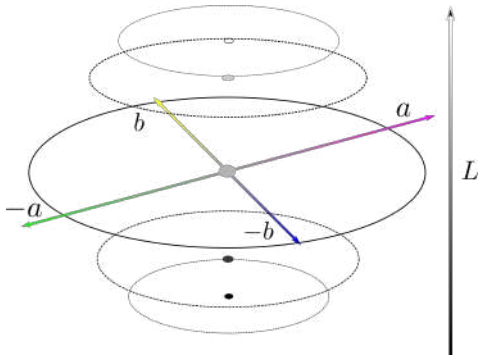


Figura A.9: Ilustración de la forma geométrica del espacio CIELab [86].

$$a^* = 500\left[f\left(\frac{X}{X_{ref}}\right) - f\left(\frac{Y}{Y_{ref}}\right)\right] \tag{A.4}$$

$$b^* = 200\left[f\left(\frac{Y}{Y_{ref}}\right) - f\left(\frac{Z}{Z_{ref}}\right)\right] \tag{A.5}$$

donde:

- $$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} 0.4124 & 0.3575 & 0.1804 \\ 0.2126 & 0.7151 & 0.0721 \\ 0.0193 & 0.1191 & 0.9502 \end{bmatrix} \begin{bmatrix} r \\ g \\ b \end{bmatrix} \tag{A.6}$$

- $$f(t) = \begin{cases} \sqrt[3]{t}, & t > \delta^3 \\ \frac{t}{3\delta^2} + \frac{4}{29}, & t \leq \delta^3 \end{cases} \tag{A.7}$$

- $$\delta = \frac{6}{29}$$

  y

- $X_{ref}, Y_{ref}, Z_{ref}$ son las coordenadas XYZ de un blanco de referencia capturado con el mismo sensor que la imagen a procesar.

Nótese que la ecuación A.6 es una multiplicación matricial y define el método para convertir del espacio RGB al CIE XYZ. La Figura A.10 ilustra en escala de grises cómo lucen cada componente del espacio CIELab.

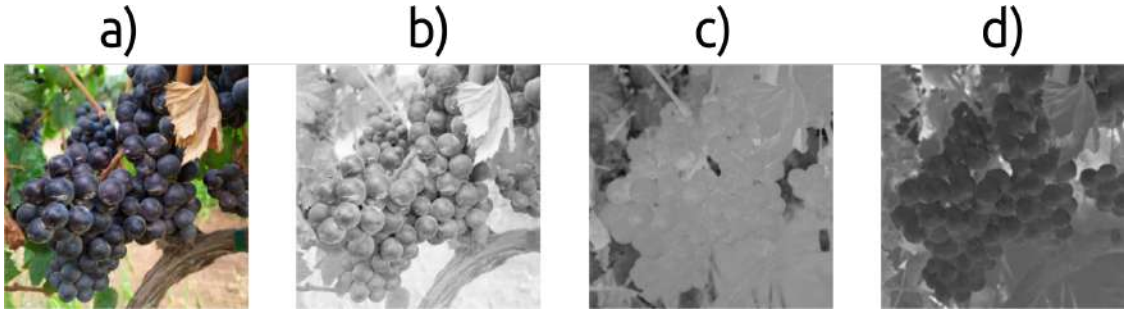


Figura A.10: Visualización del espacio CIELab. Captura en Rondo del Valle (a) mostrada en el espacio RGB; (b) componente de intensidad L, (c) componentes de cromaticidad a y (d) b. Los últimos tres presentados en escala de grises.

Espacio CIELuv

Este espacio es similar a CIELab. Su componente de intensidad es también L^* y los de cromaticidad u^* y v^*. Sus intervalos dinámicos se presentan en:

- Intensidad L^*: $[0, 100] \in \mathbb{R}$.
- Cromaticidad u^* y v^* : $[-175, 175] \in \mathbb{R}$

Para pasar del espacio RGB a CIELuv se siguen las siguientes ecuaciones:

$$L^* = \begin{cases} \left(\frac{29}{3}^3\right)\left(\frac{Y}{Y_{ref}}\right), & \left(\frac{Y}{Y_{ref}}\right) \leq \delta^3 \\ 116\left(\frac{Y}{Y_{ref}}\right)^{1/3} - 16, & \left(\frac{Y}{Y_{ref}}\right) > \delta^3 \end{cases} \tag{A.8}$$

$$u^* = 13L^*(u' - u'_{ref}) \tag{A.9}$$

$$v^* = 13L^*(v' - v'_{ref}) \tag{A.10}$$

donde:

- u'_{ref} y v'_{ref} los valores del blanco de referencia.
- $$u' = \frac{4X}{X + 15Y + 3Z} \tag{A.11}$$
- $$v' = \frac{9Y}{X + 15Y + 3Z} \tag{A.12}$$
- X, Y y Z son calculados de la misma manera que el espacio CIELab.

La Figura A.11 ilustra cómo luce cada componente del espacio CIELuv, en escala de grises.

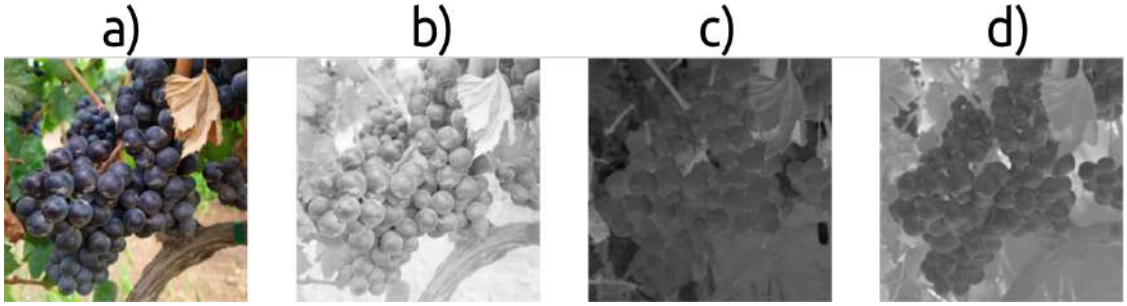


Figura A.11: Visualización del espacio CIELuv. Captura en Rondo del Valle (a) mostrada en el espacio RGB; (b) componente de intensidad L, (c) componentes de cromaticidad u y (d) v. Los últimos tres presentados en escala de grises.

Espacios YUV y YCbCr

Ambos espacios son utilizados como estándar para las imágenes de televisión. La intensidad del color es representada por el componente Y, mientras que la cromaticidad por U, V y Cb, Cr, respectivamente para cada espacio. Su intervalo dinámico dependerá del intervalo del espacio RGB y para transformarlos se utilizan las siguientes ecuaciones:

$$\begin{bmatrix} Y \\ U \\ V \end{bmatrix} = \begin{bmatrix} 0.299 & 0.587 & 0.114 \\ -0.147 & -0.289 & 0.436 \\ 0.615 & 0.515 & -0.1 \end{bmatrix} \begin{bmatrix} r \\ g \\ b \end{bmatrix} \tag{A.13}$$

$$\begin{bmatrix} Y \\ C_b \\ C_r \end{bmatrix} = \begin{bmatrix} 0.299 & 0.587 & 0.114 \\ -0.169 & -0.331 & 0.5 \\ 0.5 & -0.419 & -0.081 \end{bmatrix} \begin{bmatrix} r \\ g \\ b \end{bmatrix} \tag{A.14}$$

Las figuras A.12 y A.13 ilustran en escala de grises, cómo luce cada componente de los espacios YUV y YCrCb, respectivamente.

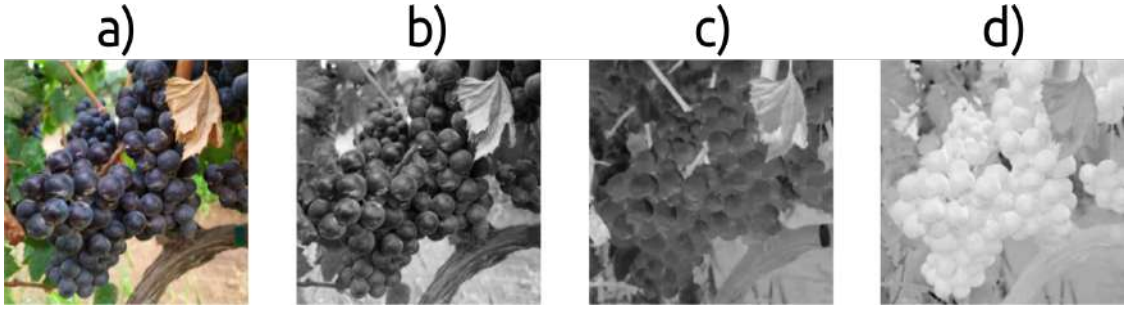


Figura A.12: Visualización del espacio YUV. Captura en Rondo del Valle (a) mostrada en el espacio RGB; (b) componente de intensidad Y, (c) componentes de cromaticidad U y (d) V. Los últimos tres presentados en escala de grises.

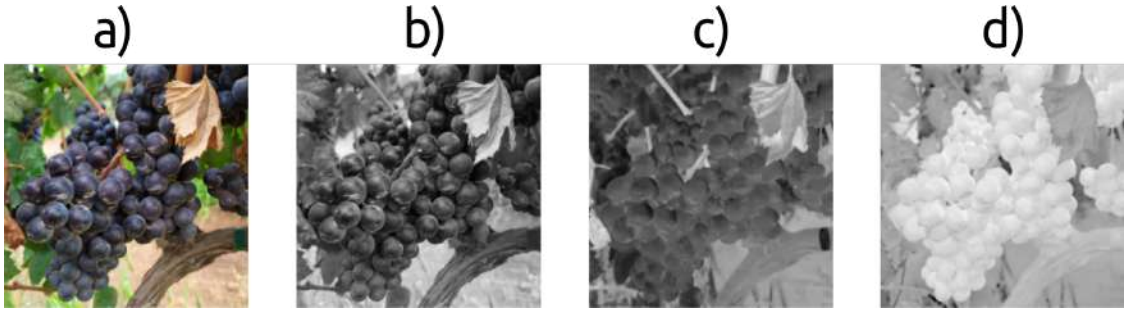


Figura A.13: Visualización del espacio YCbCr. Captura en Rondo del Valle (a) mostrada en el espacio RGB; (b) componente de intensidad Y, (c) componentes de cromaticidad Cb y (d) Cr. Los últimos tres presentados en escala de grises.

Los espacios de color previamente explicados requieren una representación matricial en 3D, sea $\mathbf{A}_{cs} \in \mathbb{R}^{M \times N \times C}$ una imagen transformada del espacio RGB a cualquier espacio de color, donde $C = 3$. Aunque los espacios de color son representaciones de las imágenes de entrada, se pueden encontrar algunas más simples para conservar las características del color. Ahora se explican los índices de color, ampliamente utilizados en imágenes de agricultura.

A.1.3. Índices de color

Los índices de color fueron usados en imágenes de agricultura para distinguir plantas del suelo [40]. Todos los índices son transformados del espacio RGB y pueden ser representados por una imagen 2D, sea $\mathbf{A}ci \in \mathbb{R}^{M \times N}$ una imagen índice de color transformado del espacio RGB. En [40] explicaron 10 de ellos:

1. Índice de diferencia normalizada (NDI *Normalized Difference Index*): propuesto por [102] y se calcula con la siguiente ecuación:

$$NDI = 128 \cdot \left(\left(\frac{G - R}{G + R} \right) + 1 \right) \tag{A.15}$$

2. Índice de exceso de verde (ExG, *Exess Green Index*): propuesto por [103] y se calcula de la siguiente manera:

$$ExG = 2G' - R' - B' \tag{A.16}$$

donde R', G' y B' son:

$$R' = \frac{R}{R + G + B}, G' = \frac{G}{R + G + B}, B' = \frac{B}{R + G + B} \tag{A.17}$$

3. Índice de exceso de rojo (ExR, *Excess Red Index*): Propuesto por [104] y definido por:

$$ExR = 1.3R - G \tag{A.18}$$

4. Índice de color para extracción de vegetación (CIVE, *Color index of vegetation extraction*): propuesto por [105] y formalmente definido por:

$$CIVE = 0.441R - 0.811G + 0.385B + 18.78745 \tag{A.19}$$

5. Índice de exceso de verde menos exceso de rojo (ExGR, *Excess Green minus excess red index*): propuesto por [106] y definido por:

$$ExGR = ExG - ExR \tag{A.20}$$

6. Índice de la diferencia verde-roja normalizada (NGRDI, *Normalized Green-Red difference index*): propuesto por [107] y dado por:

$$NGRDI = \frac{G - R}{G + R} \tag{A.21}$$

7. Índice vegetativo (VEG, *Vegetative Index*): propuesto por [108] y definido por:

$$VEG = \frac{G}{R^a B^{1-a}} \tag{A.22}$$

donde a es una constante igual a 0.667.

8. Índice modificado de exceso de verde (MExG, *Modified Excess Green Index*): propuesto por [109] y definido por:

$$MExG = 1.262G - 0.884R - 0.311B \tag{A.23}$$

9. Combinación de índices 1 (COM1): seleccionando algunos índices [110] propone el siguiente índice:

$$COM1 = ExG + CIVE + ExGR + VEG \tag{A.24}$$

10. Combinación de índices 2 (COM2): ponderando algunos índices [111] propone el siguiente índice:

$$COM2 = 0.36ExG + 0.47CIVE + 0.17VEG \tag{A.25}$$

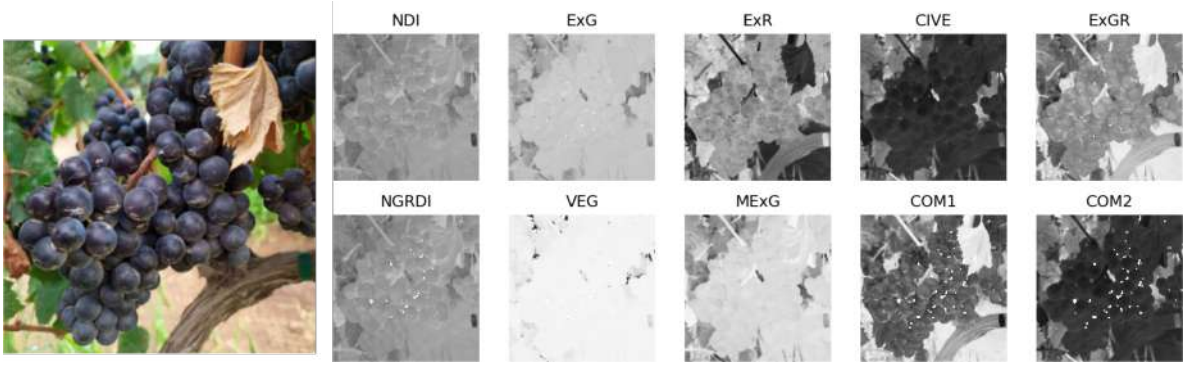


Figura A.14: Los índices de color. Muestra en escala de grises sobre una imagen capturada en Rondo del Valle.

Nótese que por cada índice de color solo se requiere una imagen 2D (Figura A.14).

Hasta este momento se ha atendido la información correspondiente a características de color, pero no es la única representación para captar la información esencial de las imágenes de entrada. Además, el transformar de un espacio de color a otro no reduce la dimensión de la imagen y en lo siguiente se explica cómo se calculan representaciones de las imágenes que tienen una dimensión menor.

A.1.4. Descriptores visuales

Una parte esencial para entrenar a los algoritmos de aprendizaje de máquina, es el vector de características. Este es la representación numérica de las imágenes, que vive en un espacio de dimensión d finita y –de alguna manera– es más simple que la imagen original. También se pueden ver como una representación traducida a un lenguaje entendible para un algoritmo computacional. Para obtener un vector que describa a cada imagen se requiere una función para transformarlo; sea $\mathbf{D}$ una vector que almacene al descriptor de la imagen $\mathbf{A}$ formalmente definido por:

$$\mathbf{D} = \phi(\mathbf{A}) \tag{A.26}$$

donde $\phi : \mathbf{A}^{M\times N\times C} \rightarrow \mathbf{D}^{1\times d}$. En visión por computadora a estos vectores se les conoce como: *descriptores visuales*; en lo siguiente se explican tres de ellos.

Patrones locales binarios

Este vector (LBP, *Local Binary Pattern*) es un descriptor de textura y tiene como objetivo encontrar micro patrones dentro de una imagen [112]. Primero, LBP calcula una derivada circular de primer orden y concatena las direcciones de las gradientes binarias, dando como resultado una imagen de textura. Por último, se calcula un histograma de esta imagen y este histograma corresponde al descriptor LBP. Este extrae la

información de las variaciones locales en una imagen en escala de grises [112]; se calcula píxel-a-píxel considerando la vecindad en un radio R alrededor del píxel q_c y lo define la siguiente ecuación:

$$LBP(P,R) = \sum_{p=0}^{P-1} s(q_p - q_c)2^p, \tag{A.27}$$

donde P es el número de píxeles en la vecindad, q_p el píxel vecino comparándose con q_c y:

$$s(x) = \begin{cases} 1, & \text{Si } x \geq 0 \\ 0, & \text{cualquier otro caso} \end{cases}, \tag{A.28}$$

Nótese que la función $s(x)$ arroja un vector binario que usa como valor de umbral el píxel q_c; todos los valores en el radio vecino R mayores al q_c serán 0 y los menores o iguales 1, dando un vector binario de P elementos. El factor 2^p es para tomar en cuenta la posición del elemento para convertirlo de binario a un valor decimal. En la Figura A.15 se puede apreciar cómo luce este descriptor.

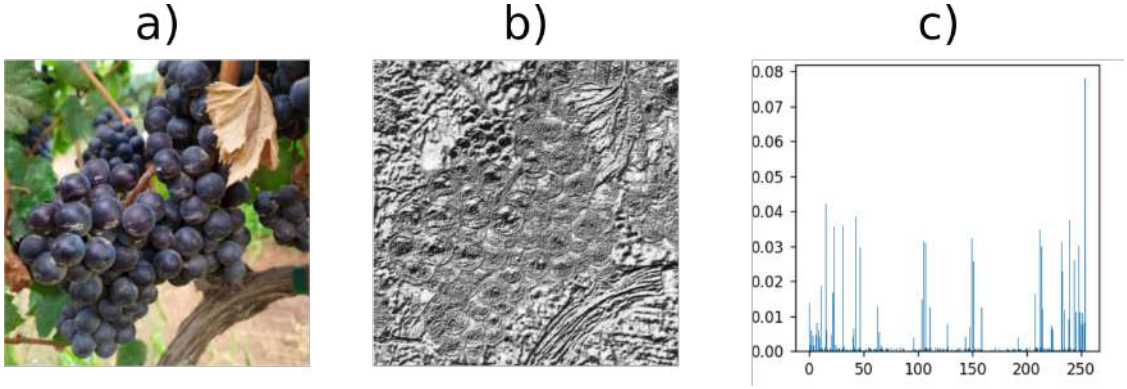


Figura A.15: Descriptor LBP, (a) La imagen de entrada, (b) la imagen de textura y (c) el histograma de los patrones locales binarios.

Histograma de gradientes orientados

Este descriptor fue usado de manera formal por Lowe [113]. HOG (*Histogram of Oriented Gradients*) es una representación de la acumulación de direcciones de los gradientes sobre cada píxel en un región llamada "celda"; luego se construye un histograma 1D compuesto por una concatenación de todos los histogramas calculados.

Sea L una función con valor de intensidad en escala de grises con respecto a la imagen a analizar. La imagen es dividida en celdas de $N' \times N'$ píxeles y la orientación $\theta_{x,y}$ del gradiente en cada píxel es calculado como:

$$\theta_{x,y} = \tan^{-1} \frac{L(x, y+1) - L(x, y-1)}{L(x+1, y) - L(x-1, y)} \tag{A.29}$$

Así mismo, las orientaciones $\theta_i^j \quad i = 1, ..., N'^2$, donde cada celda j es cuantificada y acumulada en un histograma de M-valores (Figura A.16).

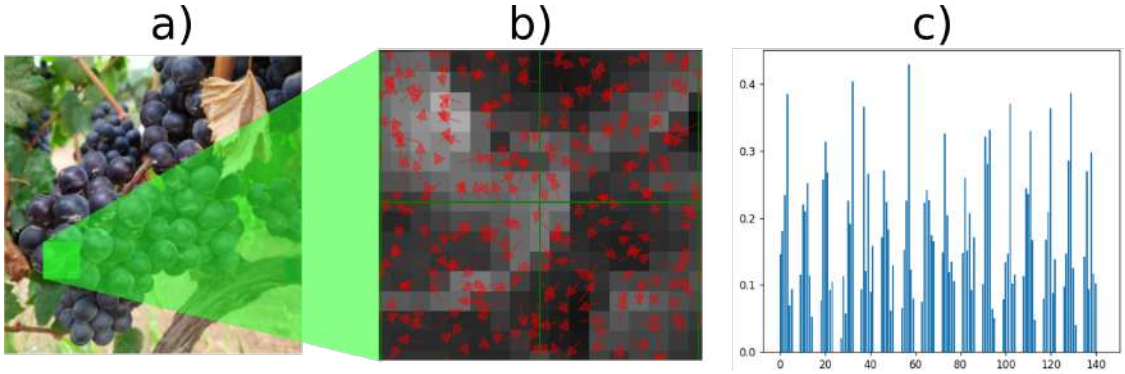


Figura A.16: El descrptor HOG; (a) la imagen de entrada, (b) porción de las direcciones de los gradientes de cada píxeles representados por flechas y (c) el histograma de gradientes orientados.

Histograma de color difuso

Para generar los histogramas de color difuso (FCH, *Fuzzy Color Histogram*), la imagen necesitar estar en el espacio HSV y esta es modelado en 36 colores usando lógica difusa [114]. Por un lado, si se toman en cuenta pocos colores del componente de Matiz (Hue) del espacio de color para disminuir el tiempo de procesamiento, pudiéramos desviarnos del color original. Por otro lado, si se consideran muchos colores, entonces el tiempo de procesamiento incrementaría, así como también, que pudiera haber más colores de los que el ojo humano puede diferenciar. Para esto, son considerados 12 matices de color, los mismos usados en *Google Images*. Entonces usando diferentes combinaciones de matiz, saturación e intensidad (valor) 36 colores son generados, y los cuales son usados como una representación de características de color.

Al usar el espacio HSV, 3 situaciones pueden presentarse, donde los colores no son representativos:

1. El matiz del color con "baja" saturación S y "bajo" valor V.
2. La saturación del color con un intervalo "pequeño" del valor V.
3. El matiz del color con saturación S "alta" valor V "alto".

Considerando estos factores, el espacio de color insaturado, fue divido en tres regiones: blanco, gris y negro. Aplicando las siguientes reglas:

$$c = \begin{cases} \text{Negro}, & \text{Si } V < T\_V\_min \\ \text{Gris}, & \text{Si } T\_V\_max < V < T\_V\_min \text{ y } S < T\_S \\ \text{Blanco}, & \text{Si } V > T\_V\_min \text{ y } S < T\_S \end{cases} \tag{A.30}$$

Donde los valores fueron fijados a:$T\_V\_min = 10$, $T\_V\_max = 85$ y $T_S = 15$. Recordar que al transformar del espacio RGB a HSV, en los tonos grises el componente H (matiz) se indetermina y por esta razón se aplican las reglas.

Los colores son constantes y el tamaño de los intervalos son fijos. Una forma trapezoidal determina la función de membresía y de manera formal se define como:

$$\mu(x) = \begin{cases} 0, & (x > a) \text{ o } (x > d) \\ \frac{x-a}{b-a}, & a \leq x \leq b \\ 1, & b \leq x \leq c \\ \frac{d-x}{d-c}, & c \leq x \leq d \end{cases} \tag{A.31}$$

tal que $a < b < c < d$ y a y d son los límites inferior y superior respectivamente, y b y c son los límites de soporte superior e inferior, respectivamente (Figura A.17).

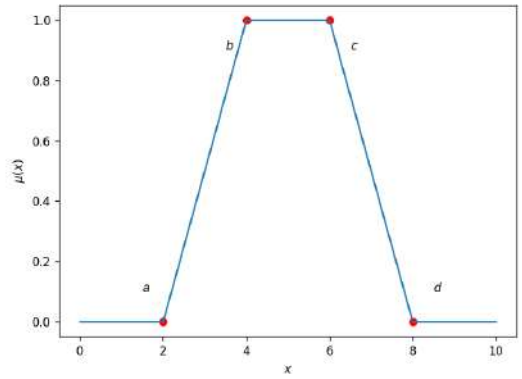


Figura A.17: Ilustración de la Ecuación A.31.

Para definir el matiz de las áreas difusas se aplican las siguientes reglas:

1. El centro de cada trapecio es el valor que define cada color en el modelo de matices.
2. El punto de cruce de dos funciones de membresía es a 0.5.
3. La pendiente izquierda de la función de membresía es igual a la pendiente derecha de la función adyacente y viceversa.

4. Más de dos diferentes funciones de membresías nunca se traslaparán.

La Figura A.18 muestra las 12 funciones que representan todos los matices en el espacio del descriptor de colores difusos.

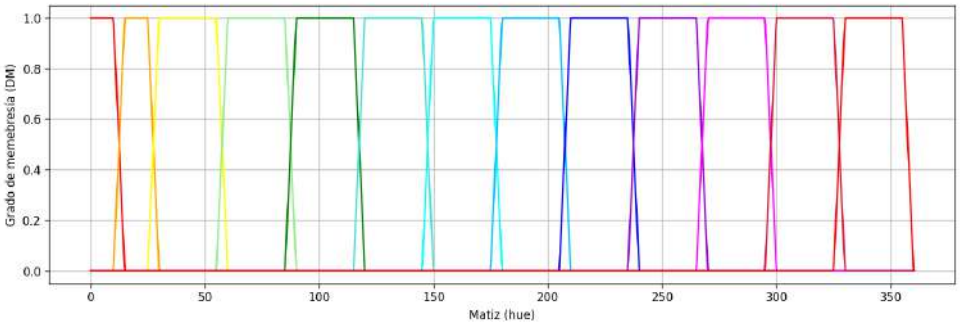


Figura A.18: El color difuso. Las 12 funciones trapezoidales que modelan al color de manera difusa [114].

Las reglas para generar la función de membresía difusa para la saturación S y el valor V son:

1.
$$DM \leftarrow \begin{cases} low, & S < 25 \text{ o } V < 25 \\ mid, & 25 < S < 75 \text{ o } 25 < V < 75 \\ high, & S > 75 \text{ o } V > 75 \end{cases} \tag{A.32}$$

2. El punto de cruce de dos funciones de membresías es igual a 0.5.

3. Más de dos funciones de membresía nunca se traslaparán.

La Figura A.19 muestra el modelo de las funciones trapezoidales del DM para saturación e intensidad.

Sin embargo, la definición de un color en particular usando solo el componente de matiz, no es posible. Entonces, los componentes de saturación y de valor, deben ser considerados. Por ejemplo, si el matiz de un píxel particular es naranja, y tiene una cantidad baja de saturación y valor, entonces el color de ese píxel es naranja oscuro; por el contrario si son altos mostrarán un naranja claro. Así, para matiz, se tienen tres variaciones de color: matiz-oscuro, matiz y el matiz-claro.

Para construir el vector de características, primero se calcula el grado de membresía (DM) para cada valor de color nítido de entrada basados en la función de membresía difusa. Luego, para cada píxel los colores insaturados son procesados. Si el píxel es un color saturado, el DM es calculado para su matiz H usando las reglas difusas y

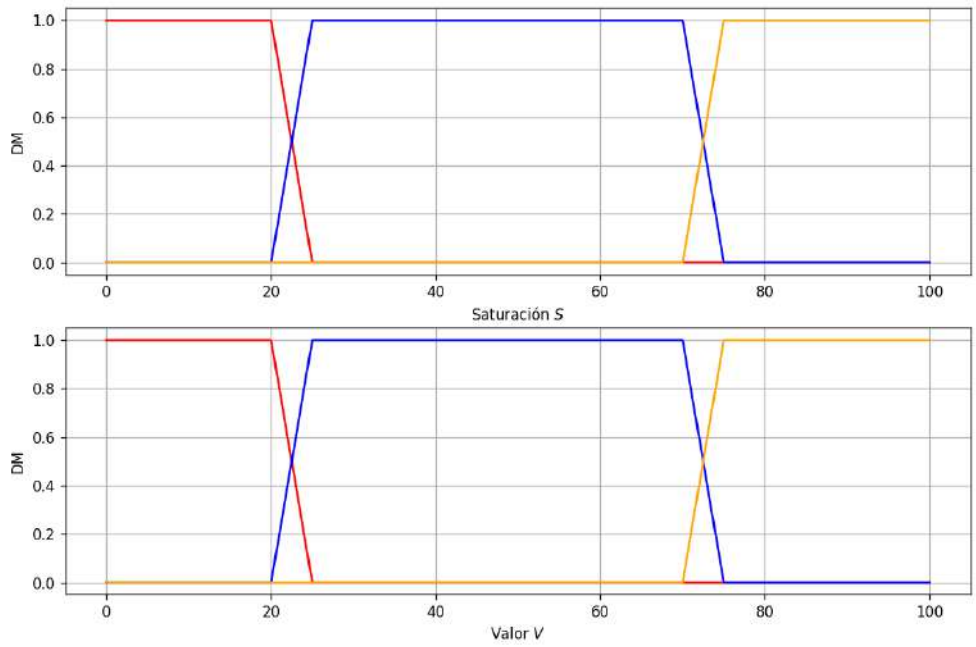


Figura A.19: Saturación e intensidad difusas. Las tres funciones trapezoidales que modelan a la saturación y la intensidad de manera difusa [114].

asignándolo a cada trapecio. Si el píxel es asignado a la función de membresía de matiz, entonces, para cada píxel, la función de membresía es calculada y asignada para cada trapecio de matiz. Luego de eso, considerando las diferentes variaciones de S y V, para cada matiz, tres diferentes colores pueden generarse. Después, la salida es defuzzificada usando el operador de agregación MAX. Por último, el valor nítido de decisión es producido, el cual será la salida final del sistema difuso de colores. Siendo así, un vector difuso de características de color de 36 componentes es calculado y almacenada como un vector de características de color, dando como resultado este descriptor de color. El FCH de $\mathbf{A}_{hsv}$, puede expresarse como una función $F(\mathbf{A}_{hsv}) = [f_1, f_2, ..., f_n]$ donde:

$$f_i = \sum_{j=1}^{N} \mu_{ij} P_j = \frac{1}{N} \sum_{j=1}^{N} \mu_{ij} \tag{A.33}$$

donde:

- $$P_j = \begin{cases} 1, & \text{si el } j\text{-ésimo píxel es cuantificado en el } i\text{-ésimo bin de color} \\ 0, & \text{cualquier otro caso} \end{cases} \tag{A.34}$$

- La función de membresía se calcula:

$$\mu_{ij} = \frac{(1/dist(x_i, c_j)^2)^{\frac{1}{p-1}}}{\sum_{q=1}^{k} (1/dist(x_i, c_q)^2)^{\frac{1}{p-1}}} \tag{A.35}$$

Y además:

- x_i: es el valor de cada píxel transformado al espacio HSV.
- c_i: es el valor del centro en el modelo utilizado.
- p: un valor auxiliar que ayuda a decidir el grado de membresía normalmente igual a 2.
- $dist(a, b)$: es la distancia euclidiana entre los puntos a y b.

Nótese que los c_i son los centros calculados del modelo estático de las 12 funciones trapezoidales por cada color y las tres funciones de cada componente (Figura A.20).

Una vez que se tiene el vector de características, el siguiente paso es definir si podemos disminuir su dimensión. En la siguiente sección se explican dos métodos para lograrlo.

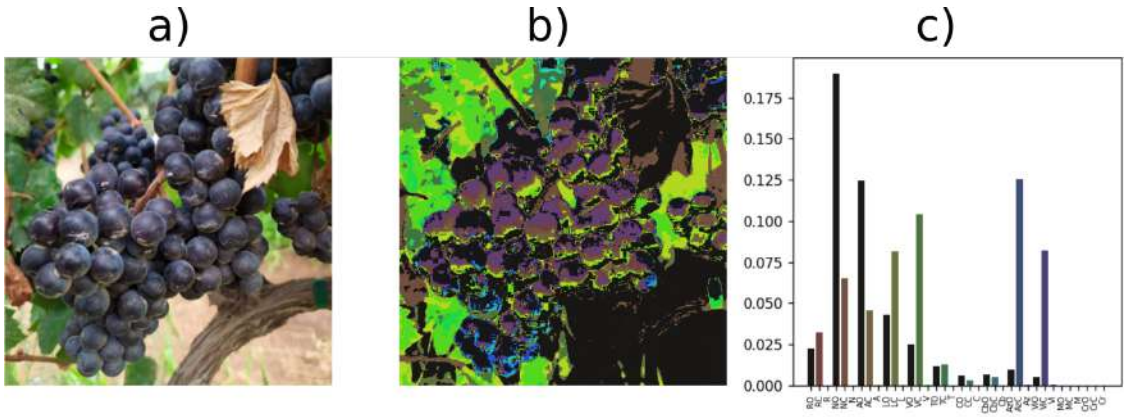


Figura A.20: El descriptor FCH. (a) La imagen de entrada, (b) imagen cuantificada por colores difusos y (c) el histograma de colores difusos.

A.1.5. Reducción de dimensión

En este trabajo se enfocó en dos técnicas para reducir el número de elementos en un vector de características, a continuación se explican a detalle.

Análisis por componentes principales

El análisis por componentes principales (PCA, *Principal Components Analysis*) se basa en el hecho de que los vectores de característica en el conjunto de datos, presentan una correlación alta. Este análisis transforma los datos originales para eliminar esta correlación e identifica la combinación lineal óptima de las características empleando sus propiedades estadísticas de cada muestra para examinar la dependencia. Usa el principio matemático de descomposición de los valores y vectores propios de la matriz de covarianza, calculada de los vectores de características; para lo cual, se lleva a cabo el siguiente procedimiento [115]:

Teniendo una entrada de datos $\mathbf{X} = \{x_1, x_2, ..., x_l\} \in \mathbb{R}^n$

- Se obtiene el vector media de todas las características:

$$\mu = \frac{1}{l} \sum_i x_i$$

- Covarianza:

$$C = \frac{1}{l} \sum_i (x - \mu)(x - \mu)^T$$

- Valores y vectores propios:

$$[U, \Lambda] = eig(lC)$$

- Proyectando los datos de entrada para obtener:

$$\hat{x} = U_k^T x_i$$

Y los datos transformados: $\hat{\mathbf{X}} = \{\hat{x_1}, \hat{x_2}, ..., \hat{x_l}\} \in \mathbb{R}^n$

Incrustación vecina estocástica en distribución-t

Esta técnica (TSNE, *T-distributed Stochatstic Neigbor Embedding*) fue propuesta por L. van der Maaten y G. Hinton [116]. Este método convierte distancias entre puntos de alta dimensión en probabilidades condicionales que representen similitudes. La similitud entre el punto x_i y el punto x_j es la probabilidad condicional $p_{i|j}$, tal que x_i elegiría a x_j como su vecino, si ambos caen en una porción a su probabilidad de densidad bajo una curva Gaussiana centrada en x_i. Para puntos cercanos, $p_{i|j}$ es relativamente alta, mientras que para puntos separados, $p_{i|j}$ será casi infinitesimal, para valores razonables de la varianza de la Gaussiana, σ_i^2. Matemáticamente, la probabilidad condicional $p_{i|j}$ está dada por:

$$p_{i|j} = \frac{exp(-\left\|x_i - x_j\right\|^2 / 2\sigma_i^2)}{\sum_{k \neq i} exp(-\left\|x_i - x_k\right\|^2 / 2\sigma_i^2)} \tag{A.36}$$

Donde σ_i^2 es la varianza de la curva Gaussiana que está centrada en el punto x_i. Para la contra-parte de baja dimensión y_i y y_j de los puntos x_i y x_j de alta dimensión, es posible calcular una probabilidad condicional similar, la cual se denota por $q_{i|j}$, la varianza de la Gaussiana que se emplea para calcular $q_{i|j}$ es fijada a $\frac{1}{\sqrt{2}}$. Por lo tanto se modela la similitud de transformar el punto y_j a y_i dado por:

$$q_{i|j} = \frac{exp(-\left\|y_i - y_j\right\|^2)}{\sum_{k \neq i} exp(-\left\|x_i - x_k\right\|^2)} \tag{A.37}$$

TSNE minimiza la suma de la divergencia de Kullback-Leibler sobre todos los puntos usando el método descenso de gradiente. La función de costo C está dada por:

$$C = \sum_i KL(P_i||Q_i) = \sum_i \sum_j p_{i|j} log \frac{q_{i|j}}{p_{i|j}} \tag{A.38}$$

Dentro de lo más relevante de TSNE, es que ocupa un parámetro de entrada llamado **perplejidad**, el cual consiste en definir la densidad de la distribución considerada como vecindad; y los otros parámetros importantes son el número de **iteraciones** en el descenso de gradiente y la **taza de aprendizaje**, para más detalles revisar [116].

Nótese que ambos métodos (PCA y TSNE) transforman los datos a una dimensión menor a la original. Otra manera para reducir la dimensión consiste en seleccionar las características más relevantes al conjunto de datos.

Selección de características

Los métodos mencionados anteriormente, son utilizados para reducir el número de características a través de una transformación. En lo siguiente se explican dos métodos para seleccionar las características más relevantes, un diagrama de estos métodos se observa en la Figura A.21.

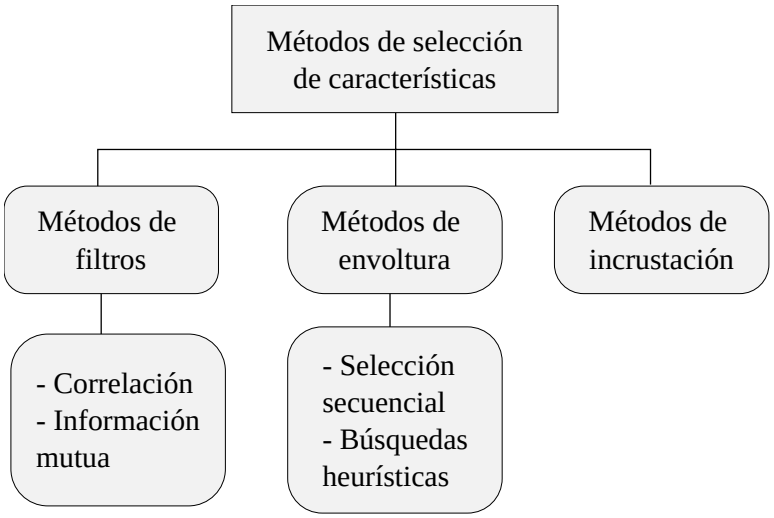


Figura A.21: Selección de características. Clasificación de los métodos empleados para selección de características [117].

Este trabajo se enfocó solo en los métodos de filtro. Estos utilizan como auxiliar una variable de categoría como principal criterio de selección, y un valor de umbral que remueve todas las variables que estén por debajo de él. Para evaluar este criterio tenemos los datos de entrada $[x_{ij}, y_k]$, los cuales consisten en N muestras, i va de 1 hasta N, con D variables (dimensiones o características), mientras que j va de 1 hasta D, donde x_i es la i-ésima muestra y y_k es la k-ésima clase correspondiente a Y.

- **Correlación**:
  Uno de los criterios más simples es, el coeficiente de correlación de Pearson y es definido de la siguiente manera:

$$R(i) = \frac{\text{cov}(x_i, Y)}{\sqrt{\text{var}(x_i) * \text{var}(Y)}} \tag{A.39}$$

- **Información mutua**:
  Este criterio mide la dependencia entre dos variables, usando la fórmula de la entropía de Shannon:

$$H(Y) = -\sum_{Y} p(y) log(p(y)) \tag{A.40}$$

Y su expresión condicional:

$$H(Y|X) = -\sum_{X}\sum_{Y} p(x,y)log(p(y|x)) \tag{A.41}$$

Esto implica que observando la variable X, la incertidumbre en la salida Y es reducida. La reducción de la incertidumbre es dada por:

$$I(Y,X) = H(Y) - H(Y|X) \tag{A.42}$$

Estos procedimientos son comúnmente usados para dos objetivos: visualizar en dimensiones menores y reducir la cantidad de características con las que un algoritmo de ML será entrenado. En la siguiente sección se describen los conceptos básicos de los algoritmos de ML más utilizados en este trabajo.

A.2. Aprendizaje de máquina

El aprendizaje de máquina (ML) es la capacidad que tiene un sistema computacional para aprender de manera automática a través de la experiencia aportada por un conjunto datos de [118]. Este enfoque construye modelos que asignan un resultado a un dato de entrada nunca antes analizado. La facilidad al acceso de recurso computacional de la época, ha hecho posible el entrenar modelos complejos con grandes volúmenes de datos, dando como resultados predicciones con mayor certeza.

Ahora bien, para entender el núcleo de ML se muestra un ejemplo sencillo adaptado de [52]. Suponiendo que se requiere pesar sacos de naranjas y la báscula disponible no funciona, pero, se cuenta con registros de días anteriores que incluyen los pesos y la cantidad de naranjas que había en cada saco. Se puede observar una relación entre estas cantidades (Figura A.22).

Ya que no se tiene conocimiento previo de la tarea, la suposición más simple es que ambas mediciones están linealmente relacionadas, es decir, tratar de ajustar una línea recta en el gráfico de dispersión a través de la ecuación de la recta:

$$y = m \cdot x + b \tag{A.43}$$

donde m y b corresponden a un factor y un valor de adición, respectivamente. a la cantidad de naranjas (x), mientras que y es el peso en kilogramos. El reto consiste en conocer cuáles son los valores de m y b que mejor se ajustan a este problema; para ello se requiere una manera de evaluar qué tan bien se ajusta esta recta. A este método de evaluación se le conoce como: *función de pérdidas*. Esta corresponde a un solo valor para cada par m, b y los pares que resulten en la función de pérdidas con menor valor,

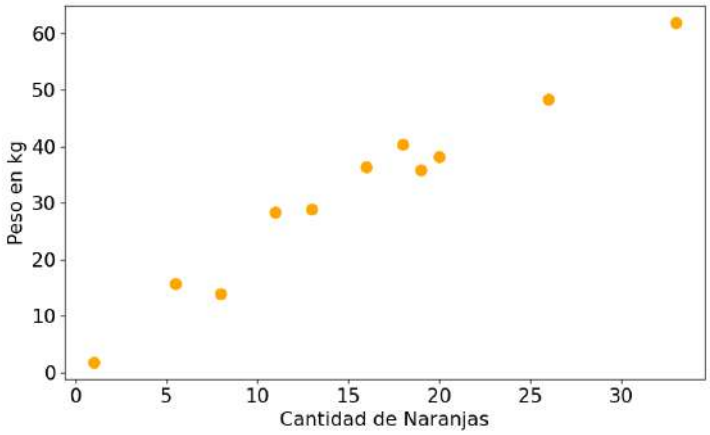


Figura A.22: Gráfico de dispersión que muestra la relación del peso con la cantidad de naranjas.

representarán una solución a este problema. La función más simple para calcular las pérdidas, es el error cuadrático medio (MSE) y está dado por la siguiente ecuación:

$$F(m,b) = \frac{1}{n}\sum_{i=1}^{n}\sqrt{(x_i - y_i)^2} \tag{A.44}$$

donde n es la cantidad de registros, x_i corresponde a la i-ésima cantidad de naranjas registrada e y_i al i-ésimo peso registrado. la Figura A.23 muestra un ejemplo de cuatro valores de m, b con su respectivo valor de $F()$.

El par $m = 1.8$ y $b = 4$ es el más adecuado a este problema, ya que es el que da como resultado un MSE menor. De tal manera que, si se tienen tres sacos con 18, 22 y 27 naranjas, sus pesos estimados serán: 36.28, 43.45 y 52.41, respectivamente (Figura A.24). Al procedimiento empleado en este ejemplo se le conoce como *Regresión lineal*.

Ahora bien, se necesita separar naranjas para dos tipos de venta: exportación y locales. En los registros de otros días se encuentra un listado de cada naranja con dos características y al final la venta objetivo (Figura A.25).

Al no conocer la naturaleza del problema, es lógico suponer que los datos se pueden separar con una línea recta. La línea verde en la Figura A.25 corresponda a la frontera de separación y las "x" representan datos nuevos a identificar. Este método se conoce como *clasificación* y tiene por objetivo asignar una clase a las observaciones basándose en sus características medibles. Como en el método de regresión, la tarea de clasificación necesita encontrar los valores de la dupla m, b para separar a los datos, y se puede usar también el MSE como función de pérdidas, pero comparando la etiqueta real contra la predicción. El MSE no es la única función para evaluar los resultados pero es ampliamente utilizada.

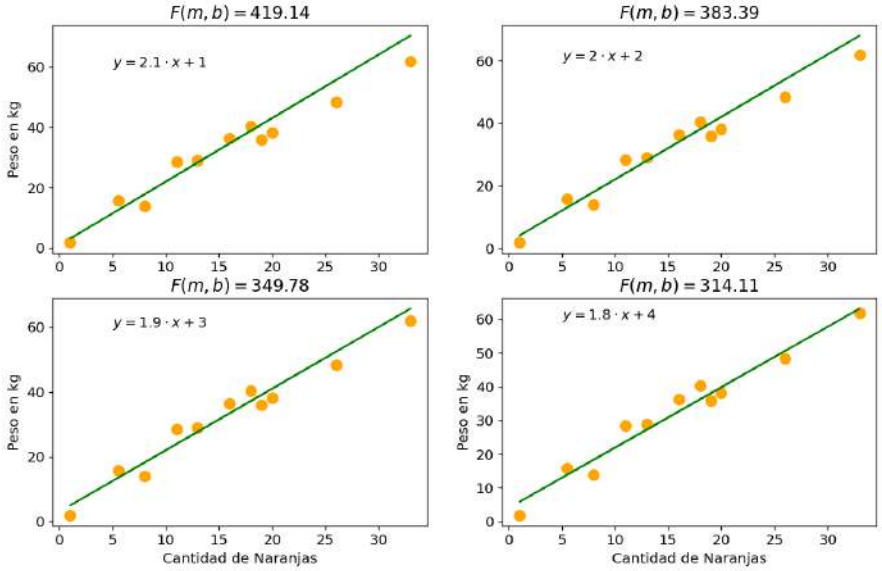


Figura A.23: Cuatro gráficos de dispersión que corresponden a una solución distinta de la Figura A.22 variando los valores de de m y b de cada recta verde.

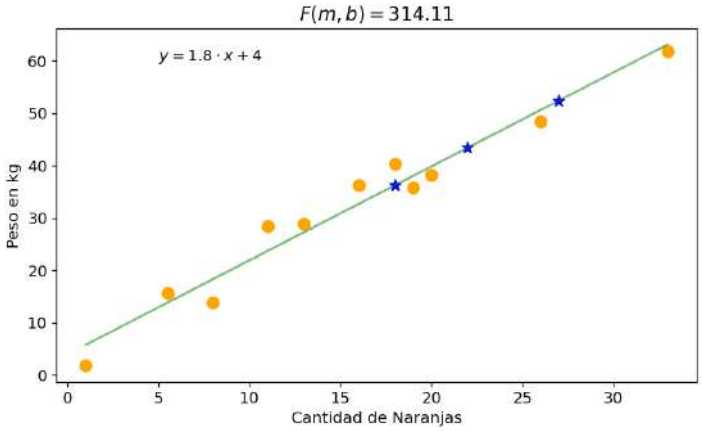


Figura A.24: Ejemplo de juguete resuelto. Gráficos de dispersión donde mostramos la predicción a tres valores nuevos.

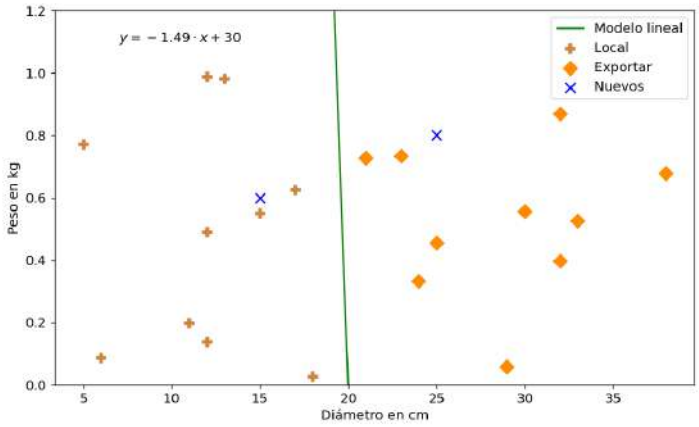


Figura A.25: Segundo ejemplo de juguete. Gráficos de dispersión donde los datos de otro ejemplo de juguete con naranjas. El eje x corresponde al diámetro en cm de cada naranja y el eje y al peso de cada naranja; mientras que la línea verde representa un modelo lineal que separa los tipo de venta y las "x" azules los datos nuevos.

A.2.1. Aprendizaje no supervisado

La tarea de segmentación está relacionada con este tipo de aprendizaje. Esta es la base de diversas soluciones a problemas de visión por computadora, y por lo tanto es ampliamente utilizada en aplicaciones de la vida real. La capacidad de un sistema de visión artificial de discernir fácilmente entre objetos, es similar al proceso de segmentación de imágenes. Este proceso se define como la unión de píxeles con características cercanas [86]; sea $\mathbf{A} = \cup_{l=1}^{n} R_l$ una imagen segmentada con n regiones tal que cada región $R_l = \{(i,j) \in \mathbb{N}^2 | \mathbf{A}(i,j) = \delta_l\}$ [119].

La técnica básica para lograr segmentación es crear una imagen $\mathbf{B}$ binaria:

$$\mathbf{B}(i,j) = \begin{cases} 1, & \mathbf{A}(i,j) > \delta \\ 0, & \text{Cualquier otro caso} \end{cases} \tag{A.45}$$

Donde δ es un umbral en el intervalo dinámico $[0, k-1]$, y $\mathbf{A} \in \mathbb{R}^{M \times N}$, es decir, una imagen 2D. Si $\mathbf{A} \in \mathbb{R}^{M \times N \times C}$ se aplica para cada C a través del siguiente procedimiento:

$$\mathbf{B}(i,j) = \begin{cases} 1, & d(i,j) \geq d_{max} \\ 0, & d(i,j) < d_{max} \end{cases} \tag{A.46}$$

donde:

$$d = \|\mathbf{A} - \delta_C\|^2 \tag{A.47}$$

d_{max} es la distancia umbral, δ_C es un valor de umbral para cada C y $\|.\|$ es una métrica de distancia. Nótese que estas técnicas no miden cercanía entre píxeles; para medir cercanía entre se recurre a los algoritmos de ML no supervisados. En lo siguiente, se explican dos de los algoritmos de agrupamiento de datos, utilizados para realizar la tarea de segmentación.

K-medias

Este algoritmo de aprendizaje no supervisado, toma la idea básica de clasificación, es decir, asigna una clase o grupo a cada píxel y se basa en la medición de la cercanía entre píxeles [86]. Sea $\mathbf{X} \in \mathbb{R}^{n \times d}$ una imagen 2D que almacene por renglón cada píxel y por columna cada componente de color en C. Las particiones se hacen en k grupos tal que $k \leq n$ con el objetivo de minimizar la suma de los cuadrados en cada grupo $\mathbf{s} = \{s_1, s_2, ..., s_k\}$:

$$\underset{s}{\operatorname{argmin}} \sum_{i=1}^{k} \sum_{j \in S_i} \|x_j - \mu_i\|^2 \tag{A.48}$$

donde μ_i es la media de S_i. Este valor se inicia de manera aleatoria acotado al intervalo dinámico de la imagen $\mathbf{X}$. Este proceso se realiza de manera iterativa I veces o hasta que los valores de μ_i no cambien (Figura A.26).

FC-Medias

Este algoritmo es similar a K-medias pero incluye la base de lógica difusa. Las muestras x_i pertenecen a todos los grupos en una proporción ponderada, en vez de solo pertenecer a un solo grupo. Se tienen n muestras $\mathbf{X} = \{x_1, x_2, ..., x_n\}$ y cada x_i es un punto de dimensión d. Un grupo difuso es una colección de k grupos, sea $\mathbf{S} = \{s_1, s_2, ..., s_k\}$ los grupos que incluyen una matriz de partición $\mathbf{W} = w_{ij} \in [0, 1]$ para $i = 1, .., n$ y $j = 1, ..., k$. Donde cada elemento w_{ij} es un valor ponderado que corresponde al grado de membresía del i-ésimo objeto x_i al j-ésimo grupo s_j. Esta matriz $\mathbf{W}$ incorpora dos restricciones:

1. $\sum_{j=1}^{k} w_{ij} = 1$
2. $0 < \sum_{i=1}^{n} w_{ij} < 1$

La primera se asegura que la ponderación esté normalizada, mientras que la segunda, que al menos exista un elemento que no sea cero para cada s_j. El Algoritmo 2 muestra como se lleva a cabo este proceso.

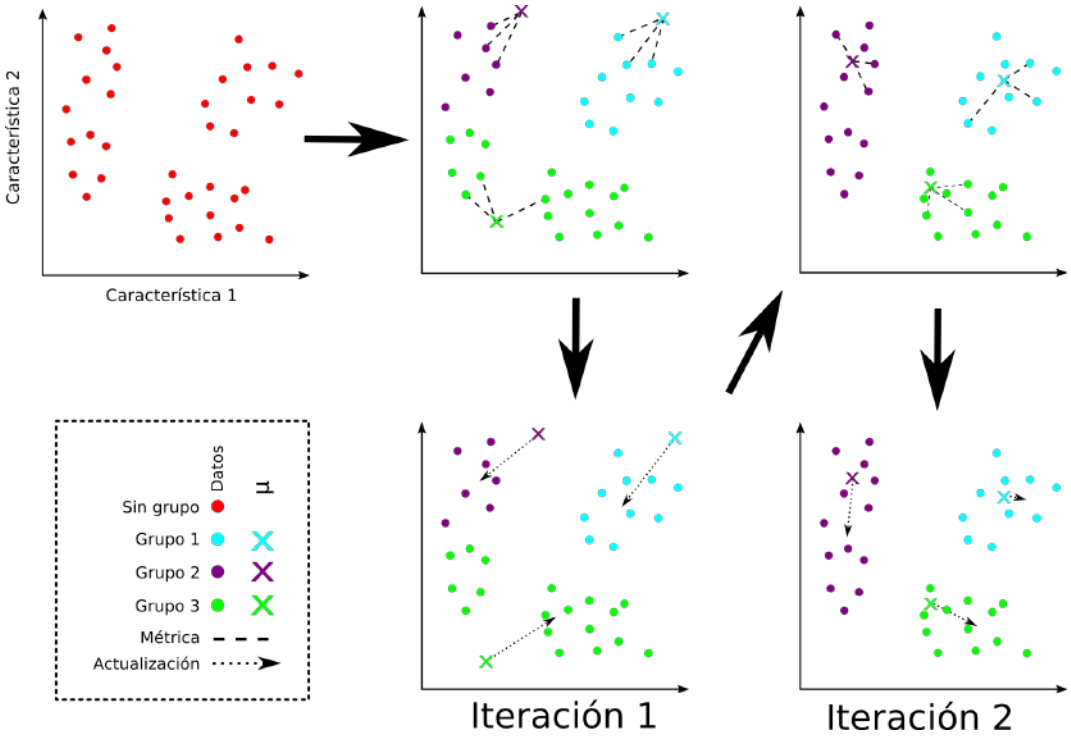


Figura A.26: Ilustración del proceso de agrupamiento del algoritmo K-medias. Nótese que el único dato que debemos conocer es el número de grupos a identificar.

Algoritmo 2 Proceso para agrupar datos usando FC-medias

Datos: X

Resultado: S

Iniciar aleatoriamente **W** con las restricciones anteriores

mientras *Se alcance el criterio de parada* **hacer**

Calcular el centro: $c_j = \frac{\sum_{i=1}^{n} w_{i,j}^{p} x_i}{\sum_{i=1}^{n} w_{i,j}^{p}}$

Actualizar $w_{ij} = \frac{(1/\|x_i - c_j\|^2)^{\frac{1}{p-1}}}{\sum_{q=1}^{k} (1/\|x_i - c_q\|^2)^{\frac{1}{p-1}}}$

fin

$s_i = \text{argmin}_i \, w_{ij}$

Donde $p \in [1, \infty]$ es un parámetro que determina la influencia de la ponderación [86]. Nótese que FC-Medias intenta minimizar la suma de errores cuadráticos SEE:

$$SSE = \sum_{j=1}^{k} \sum_{x \in C_i} w_{i,j}^{p} \left\| x_i - c_j \right\|^2 \tag{A.49}$$

Este algoritmo también calcula la cercanía entre cada muestra y los grupos, pero pondera esta distancia a partir de la matriz **W** (Figura A.27).

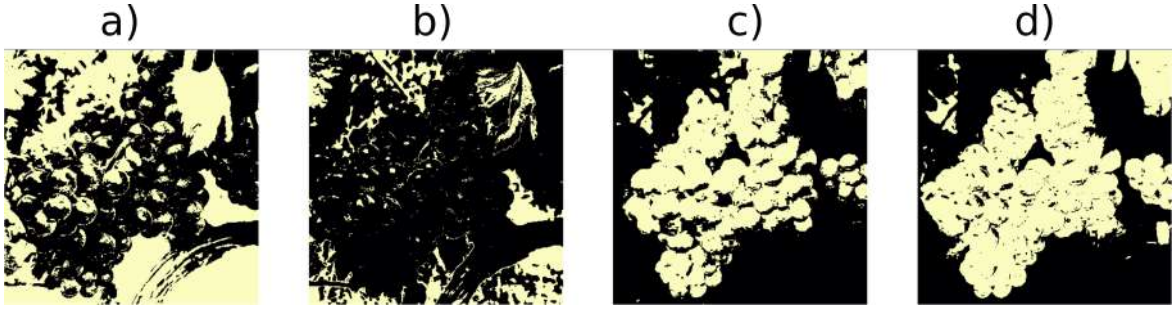


Figura A.27: Algoritmos de agrupamiento en acción. Resultados de las imágenes segmentadas por cada algoritmo de segmentación; (a) usando un solo umbral ($\delta = 100$) y la imagen en escala de grises, (b) usando tres umbrales y uno de distancias ($\delta_C = [50, 143, 100]$ y $d_{max} = 180$) sobre el espacio RGB, (c) usando K-medias en el espacio HSV con $k = 2$ e $I = 50$ y (d) usando FC-medias y el espacio HSV con $k = 2$, $I = 50$ y $p = 2$.

Se observa que al agregar características de color, la segmentación cobra sentido para problema de esta investigación. La Figura A.28 muestra los mismo resultados la Figura A.27 sobre-puesta a la imagen original.

Al observar las figuras A.27 y A.28 percibimos que FC-medias es el que segmenta la mayor cantidad de píxeles que corresponden a uvas. Hasta aquí termina la explicación del aprendizaje no supervisado, y en lo siguiente se describe el aprendizaje supervisado.

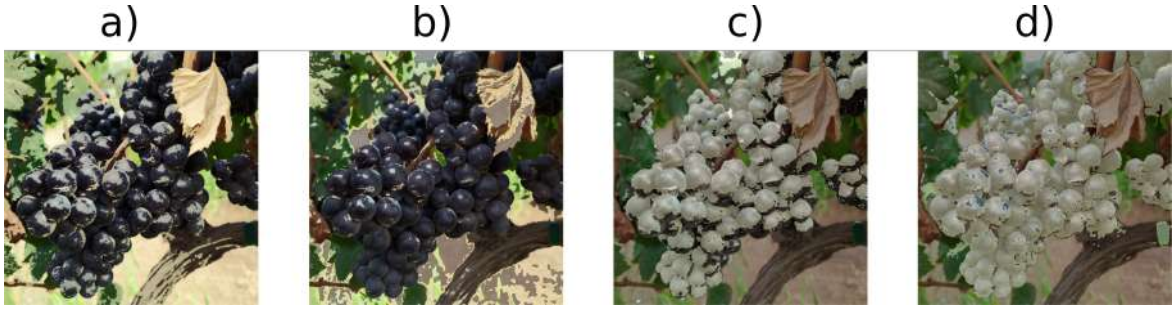


Figura A.28: Algoritmos de agrupamiento en acción. Resultados de las imágenes segmentadas por cada algoritmo de segmentación sobre-puestas a la imagen original; (a) usando un solo umbral ($\delta = 100$) y la imagen en escala de grises, (b) usando tres umbrales y uno de distancias ($\delta_C = [50, 143, 100]$ y $d_{max} = 180$) sobre el espacio RGB, (c) usando K-medias en el espacio HSV con $k = 2$ e $I = 50$ y (d) usando FC-medias y el espacio HSV con $k = 2$, $I = 50$ y $p = 2$.

A.2.2. Aprendizaje supervisado

Consiste en aprender de datos etiquetados, es decir, cada muestra u observación tienen asignado un dato objetivo. Hay múltiples alternativas, y en lo siguiente se explican las más comunes.

Regresión logística multinomial

La Regresión logística multinomial (MLR, por su sigla en inglés *multinomial logistic regression*) generaliza el método de Regresión Logística para el problema de clasificación multiclase, es decir, con más de dos posibles resultados discretos. MLR se trata de un modelo que predice las probabilidades de los resultados posibles de una distribución categórica como variable dependiente, dado un conjunto de variables independientes. Su ecuación en forma general es [21]:

$$g_k(x) = log\frac{P(\omega_{k-1}|x)}{P(\omega_k|x)} = w_{(k-1)0} + w_{k-1}^t x \tag{A.50}$$

Donde:

- w es el vector para la nueva proyección,
- $P(\omega_k|x)$ es la probabilidad a posterior de cada clase ω_k y se calcula por la siguiente ecuación:

$$P(\omega_k|x) = \frac{1}{1 + \sum_{j=1}^{k-1} e^{w_{j0} + w_j^T x}} \tag{A.51}$$

.

Aquel valor $g_k(x)$ que sea mayor, pertenecerá a la k-ésima clase.

K Vecinos cercanos

Este método no es un método de clasificación multi-instancia, sin embargo, J. Wang et al. [120] proponen una adaptación para lograrlo. K-vecinos cercanos (KNN, por su sigla en inglés *K nearest neighbors*) describe que: sea $\mathbf{X}$ un conjunto de observaciones $\{x_1, x_2, .., x_n\}$ con d dimensiones cada observación x_i. Esta es asignada a la clase C, si esta clase es la más frecuente entre las k observaciones de entrenamiento más cercanas. Para definir la frecuencia entre los vecinos, se utiliza una métrica, generalmente la distancia euclidiana:

$$d(x_i, x_j) = \sqrt{\sum_{r=1}^{d}(x_{ri} - x_{rj})^2} \tag{A.52}$$

Sin embargo, se pueden utilizar otras métricas, como la distancia Manhattan, la cual consiste en la suma de las longitudes de las proyecciones del segmento de línea entre los puntos sobre el sistema de ejes coordenados:

$$dm(x_i, x_j) = \|x_i - x_j\|_1 = \sum_{r=1}^{d} |x_{ri} - x_{rj}| \tag{A.53}$$

Para realizar el **entrenamiento**, cada observación x_i y su correspondiente etiqueta y_i, representaran los ejemplos de aprendizaje. Para **clasificar**, teniendo una observación $\hat{x}$ debe ser clasificado en $\mathbf{X}$ a los k vecinos más cercanos a $\hat{x}$, regresando:

$$\hat{y} \leftarrow \text{argmax}_{\hat{x}\in\hat{X}} \sum_{i=1}^{k} \delta(\hat{x}, y_i) \tag{A.54}$$

, donde:

$$\delta(a, b) = \begin{cases} 1 & a = b \\ 0 & \text{cualquier otro caso} \end{cases} \tag{A.55}$$

Este método es sensible al valor de k y la métrica empleada.

Máxima verosimilitud

El método de Máxima Verosimilitud (ML, *Maximum Likelyhood*) describe a la población base de la que ha sido extraída la muestra, como los procesos de selección que han dado lugar a dicha muestra. Este método se define formalmente con el siguiente conjunto de ecuaciones $g_i(x)$ [121]:

$$g_i(x) = -\frac{1}{2}(x - \mu_i)^T \Sigma_i^{-1}(x - \mu_i) + P(\omega_i), \tag{A.56}$$

donde:

- μ_i es la media de cada clase,
- Σ_i es la matriz de covarianza de cada clase,
- $P(\omega_i)$ es la probabilidad a *priori* de cada clase.

Al evaluar el dato de entrada x en $g_i(x)$ pertenecerá a la i-ésima clase en la g_i con el valor mayor. Nótese que en la ecuación A.56 se necesita la inversa de la matriz de convarianza Σ_i. Sin embargo, para poder calcularla debe cumplir con ciertas características: debe ser cuadrada y no debe ser singular. La Ecuación A.57 muestra el proceso de cálculo

$$A^{-1} = \frac{1}{\det(A)}\text{adj.}(A^T) \tag{A.57}$$

La matriz de covarianza Σ_i, es cuadrada, pero en ocasiones es singular, es decir, su determinante es nulo. Cuando esto sucede es necesario hacer el cálculo de la pseudo-inversa de Moore-Penrose [122] la cual consiste en:

$$A^{+} = (A^T A)^{-1} A^T \tag{A.58}$$

Máquinas de vectores de soportes

Las máquinas de vectores de soportes (SVM, *Support Vector Machine*) son utilizadas como clasificadores e intentan maximizar las regiones de los márgenes con respecto a la zona (hiper-plano) que separa una clase de otra. Suponiendo que se tiene un conjunto de datos $(\mathbf{x}_i, y_i)...(\mathbf{x}_m, y_m)$ donde $\mathbf{x}_i \in \mathcal{H}$ y $y_i \in \{\pm 1\}$, se busca una función de decisión [121]:

$$f_{\mathbf{w},b} = sgn\left(\langle \mathbf{w}, \mathbf{x} \rangle + b\right) \tag{A.59}$$

Donde:

- $\mathbf{w}$ es el vector de pesos que se requiere encontrar.
- $$sgn(x) = \begin{cases} 1, & \text{si } x > 0 \\ -1, & \text{si } x \leq 0 \end{cases} \tag{A.60}$$
- $\langle \cdot \rangle$ es el productor interior,
- b es el margen entre el hiperplano y los vectores de soporte (Figura A.29).

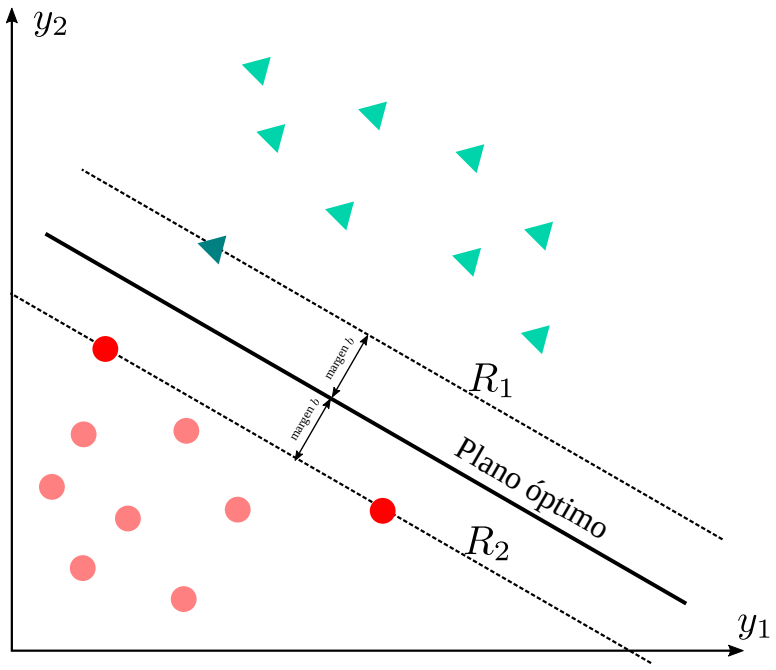


Figura A.29: Ilustración de los vectores de soporte [121].

Tal que:

$$f_{\mathbf{w},b} = y_i \tag{A.61}$$

Al hacer que $y_i \pm 1$ el patrón i corresponde a la clase ω_1 o ω_2, y se busca un discriminante lineal dado por:

$$g(\mathbf{x}) = \mathbf{w}^{\mathbf{t}}\mathbf{x} + b \tag{A.62}$$

el hiperplano separador:

$$y_i g(\mathbf{x}_i) \geq 1 \tag{A.63}$$

y el vector $\mathbf{w}$ que maximice el margen b:

$$\frac{y_i g(\mathbf{x}_i)}{\|\mathbf{w}\|} \geq 1 \tag{A.64}$$

Solucionando este problema de optimización, se tiene la función objetivo:

$$\phi(\mathbf{w}) = \frac{1}{2}\|\mathbf{w}\|^2 \tag{A.65}$$

Sujeto a:

$$y_i \cdot \langle \mathbf{w}, \mathbf{x} \rangle + b \geq 1 \quad \forall i = 1, ..., m \tag{A.66}$$

Para resolverlo se utiliza el método de los multiplicadores de Lagrange $\mathcal{L}(x, \lambda)$ dado por:

$$\mathcal{L}(x, \lambda) = f(x) - \lambda h(x) \tag{A.67}$$

Donde:

- x es la variable independiente de interés.
- $f(x)$ es la función objetivo.
- λ son los valores del multiplicador de Lagrange.
- $h(x)$ son las restricciones

Para este problema específico el multiplicador de Lagrange es calculado de la siguiente manera:

$$\mathcal{L}(\mathbf{w}, b, \lambda) = \frac{1}{2} \|\mathbf{w}\|^2 - \sum_{i=1}^{m} \lambda_i \left[(y_i \cdot \langle \mathbf{w}, \mathbf{x_i} \rangle + b) - 1 \right] \tag{A.68}$$

Para encontrar la solución de la ecuación A.68 derivamos parcialmente con respecto a b y a $\mathbf{w}$:

$$\frac{\partial}{\partial b} \mathcal{L}(\mathbf{w}, b, \lambda) = \sum_{i=1}^{m} \lambda_i y_i \tag{A.69}$$

$$\frac{\partial}{\partial \mathbf{w}} \mathcal{L}(\mathbf{w}, b, \lambda) = \mathbf{w} - \sum_{i=1}^{m} \lambda_i y_i \mathbf{x}_i \tag{A.70}$$

Al igualar a cero cada una de las ecuaciones A.69 y A.70 se tienen:

$$\sum_{i=1}^{m} \lambda_i y_i = 0 \tag{A.71}$$

$$\mathbf{w} = \sum_{i=1}^{m} \lambda_i y_i \mathbf{x}_i \tag{A.72}$$

Al regresar a la ecuación A.68 y sustituir el valor de $\mathbf{w}$ para depender solo de λ_i, nuestra nueva función objetivo es:

$$\max_{\lambda \in \mathbf{R}^m} \Phi(\lambda) = \mathcal{L}(\lambda) = \sum_{i=1}^{m} \lambda_i - \frac{1}{2} \sum_{i,j=1}^{m} \lambda_i \lambda_j y_i y_j \langle \mathbf{x}_i \mathbf{x}_j \rangle \tag{A.73}$$

Sujeto a:

$$\sum_{i=1}^{m} \lambda_i y_i = 0 \tag{A.74}$$

$$\lambda_i \geq 0 \quad \forall i = 1, ..., m \tag{A.75}$$

Al sustituir en la ecuación A.59 el valor de $\mathbf{w}$ se tiene:

$$f_{\mathbf{w},b} = sgn\left(\sum_{i=1}^{m} \lambda_i y_i \langle \mathbf{x}, \mathbf{x_i} \rangle + b\right) \tag{A.76}$$

Nótese que el producto interior $\langle \mathbf{x}, \mathbf{x_i} \rangle$ puede ser representado por una operación de kernel [123] $k(\mathbf{x}, \mathbf{x_i})$, permitiendo así introducir diferentes operaciones. Teniendo al final la siguiente ecuación que representa la operación que hace una SVM:

$$f_{w,b} = sgn\left(\sum_{i=1}^{m} \lambda_i y_i k(\mathbf{x}, \mathbf{x_i}) + b\right) \tag{A.77}$$

Donde:

- $k(\mathbf{x}, \mathbf{x_i})$ es el kernel (se usó uno de base radial en la implementación)
- y_i es la función lineal que separa cada clase
- λ_i son los coeficientes del multiplicador de Lagrange que maximizan el margen b

Para este caso también se usó la versión c-SVM, la cual incluye un parámetro C que permite suavizar el margen b. Teniendo como función objetivo la misma que la ecuación A.73 pero sujeta a las siguientes restricciones:

$$\sum_{i=1}^{m} \lambda_i y_i = 0 \tag{A.78}$$

$$0 \leq \lambda_i \leq \frac{C}{m} \quad \forall i = 1, ..., m \tag{A.79}$$

El valor de C es un valor arbitrario, conocido como el compromiso entre el error y el margen. Los detalles de todos los desarrollos se pueden encontrar en [123] y [121].

Perceptrón multicapa

El perceptrón multicapa (MLP, *Multi-Layer Perceptron*) es un método que intenta solucionar el problema de clasificación errónea, iterando las soluciones a través del algoritmo del perceptrón. Suponiendo que se tiene un conjunto de datos $(\mathbf{x}_i, y_i)...(\mathbf{x}_m, y_m)$, $\mathbf{X} \in \mathbf{R^{m \times d}}$, la función del perceptrón se describe formalmente de la siguiente manera [21]:

$$z(\mathbf{x}) = f(\mathbf{w}^t \phi(\mathbf{x})) \tag{A.80}$$

Donde:

- $f(\cdot)$ es la función de activación,
-

$$\mathbf{w} = \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_d \end{bmatrix} \tag{A.81}$$

  Son los pesos que elijen la dirección de la frontera de decisión,

- El vector $\phi(\mathbf{x})$ contendrá el valor de sesgo $\phi_0(\mathbf{x}) = 1$

Los objetivos t de $z(\cdot)$, resultado de la función de activación $f(\cdot)$ serán $t \in \{-1, 1\}$, para las clases ω_1 y ω_2, respectivamente.

El algoritmo usado para determinar el vector $\mathbf{w}$ del perceptrón, puede ser calculado minimizando una función de error, sin embargo, esto no es tan sencillo. Para esto se usa una alternativa conocida como el *criterio del perceptrón*. Esto nos lleva a, buscar un vector $\mathbf{w}$ tal que los datos $\mathbf{x}_i$ en la clase ω_1 tendrán $\mathbf{w}^t\phi(\mathbf{x}_i) > 0$, mientras que los datos $\mathbf{x}_i$ en la clase ω_2, tendrán $\mathbf{w}^t\phi(\mathbf{x}_i) < 0$. Usando el esquema de los $t \in \{-1, 1\}$ objetivos, todos los datos deberán satisfacer: $\mathbf{w}^t\phi(\mathbf{x}_i)t_i > 0$. El criterio del perceptrón asocia el error cero, con cualquier dato correctamente clasificado, mientras que para los datos $\mathbf{x}_i$ trata de minimizar la cantidad $-\mathbf{w}^t\phi(\mathbf{x}_i)t_i$; de manera formal el criterio del perceptrón está dado por:

$$J_p(\mathbf{w}) = -\sum_{i \in \mathcal{M}} \mathbf{w}^t\phi(\mathbf{x}_i)t_i \tag{A.82}$$

donde $\mathcal{M}$ denota el conjunto de datos clasificados erróneamente. La contribución al error asociado al dato clasificado erróneamente es una función lineal a $\mathbf{w}$ en las regiones del espacio $\mathbf{w}$ donde el dato clasificado erróneamente y el cero en las regiones donde es clasificado correctamente. El total de la función de error es entonces lineal a trozos. Ahora se aplica el procedimiento de descenso de gradiente estocástico, a esta función de error. El cambio en el vector $\mathbf{w}$ está dado por:

$$\mathbf{w}^{(\tau+1)} = \mathbf{w}^{(\tau)} - \eta\nabla J_p(\mathbf{w}) = \mathbf{w}^{(\tau)} + \eta\nabla\phi_i t_i \tag{A.83}$$

donde:

- $\eta \in \mathbf{R}_+$ es la taza de aprendizaje,
- τ es un valor entero que indexa los pasos del algoritmo.

Debido a que la función del perceptrón $z(\mathbf{x}, \mathbf{w})$ no cambia si multiplicamos $\mathbf{w}$ por una constante, de tal manera que, podemos establecer $\eta = 1$. Nótese que, mientras el vector de pesos evoluciona durante el entrenamiento, el conjunto de datos que están clasificados erróneamente también cambia.

La Figura A.30 muestra la convergencia del algoritmo de aprendizaje del perceptrón, mostrando dos clases de puntos (rojo y verde), en un espacio de características bidimensional (ϕ_1, ϕ_2). El gráfico superior izquierdo muestra el vector $\mathbf{w}$ mostrado como una flecha negra y su respectiva frontera (línea rayada negra), tal que la flecha que apunta hacia la región de decisión clasificó como perteneciente a la clase roja. Los puntos rodeados con un círculo verde claro, es un punto erróneamente clasificado, y entonces su vector de características es añadido al vector de pesos actual, dando la nueva decisión de frontera mostrada en el gráfico superior derecho. El mostrado en la parte inferior izquierda, muestra el siguiente dato clasificado erróneamente a ser considerado, indicado también con un círculo verde claro, de nuevo su vector de características es añadido al vector de pesos, dando la decisión de frontera mostrada en el gráfico inferior derecho, en el cual todos los datos están clasificados correctamente.

El valor de la tasa de aprendizaje η, define el tamaño del paso en el algoritmo del descenso de gradiente, por lo que el valor óptimo de η debe buscarse. También la función de activación $f(\cdot)$, puede ser alguna de las siguientes:

- Lineal: $f(x) = x$
- Sigmoide o logística: $f(x) = \frac{1}{1+\exp^{-x}}$
- Tangente hiperbólica: $f(x) = \tanh(x)$
- ReLU: $f(x) = \max(0, x)$

MLP es una extensión del algoritmo del perceptrón donde se tienen múltiples capas: capa de entrada, capa oculta y capa de salida. Con respecto a la capa oculta puede tener diversas capas y diversos nodos (conexiones) definidos por el usuario; al número de nodos y capas oculta, se le conoce también como arquitectura de la red. Los detalles de esta teoría fue obtenida en [121] y [21].

Bosque aleatorio

Suponiendo que se tiene un conjunto de datos entrenamiento $\mathbf{X} = \{(\mathbf{x}_m, y_m), m = 1, ..., M\}$, donde $\mathbf{x}_m$ es una observación, y y_m, es el predictor de salida. Un **clasificador débil** puede ser creado usando $\mathbf{X}$, siendo este un predictor $f(\mathbf{x}_m, \mathbf{X})$ con un sesgo bajo y una varianza alta. Podemos crear, muestreando aleatoriamente del conjunto $\mathbf{X}$, una colección de clasificadores débiles $f(\mathbf{x}_m, \mathbf{X}, \theta_k)$, siendo θ_k el k-ésimo vector de selección aleatoria para el k-ésimo clasificador débil. Por ejemplo, aplicando **muestreo**

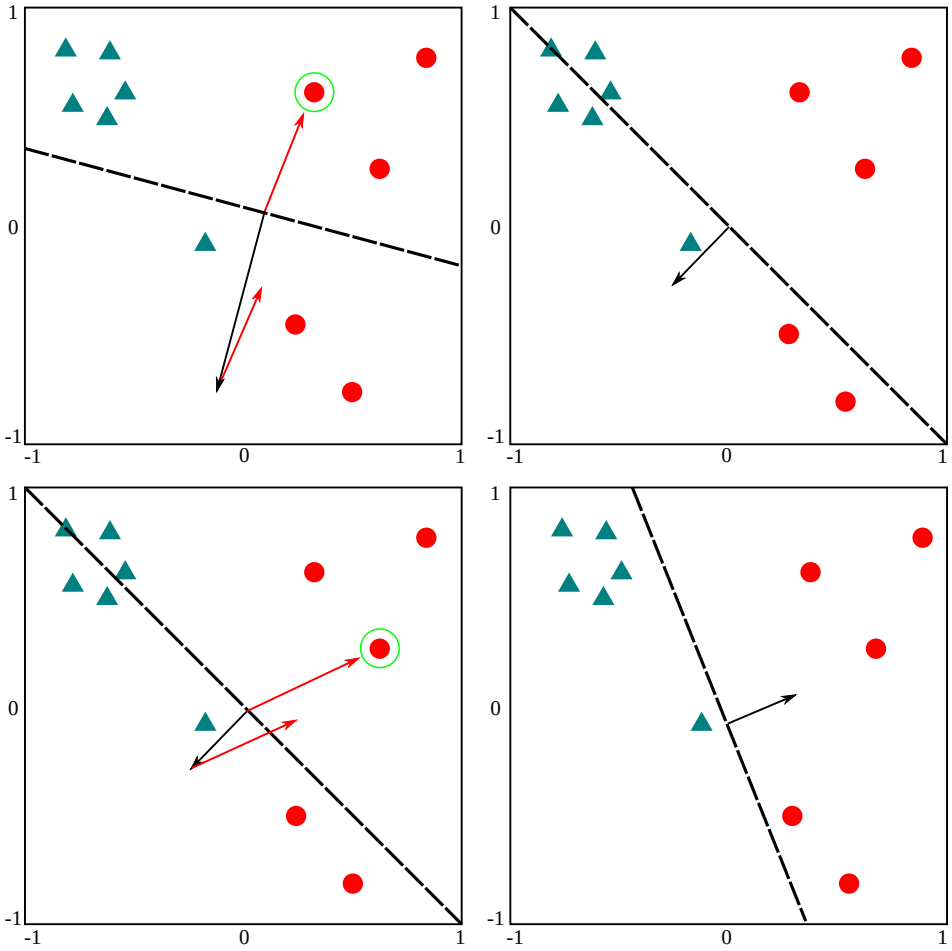


Figura A.30: Ilustración de la convergencia proceso del algoritmo del perceptrón [21].

de arranque (*bootstrap sampling*), en dos tercios de los datos del conjunto de entrenamiento, para generar cada θ_k. Siendo así, que un tercio de las observaciones quedan fuera (OOB), y los θ_k son independientes e idénticamente distribuidos (i.i.d.).

Puede mostrarse que, combinando clasificadores débiles aleatorios i.i.d. dentro de un comité, deja el sesgo sin cambios, mientras que la varianza se reduce por un factor $\overline{\rho}-$ la media de la correlación entre los clasificadores débiles. Entonces, si la correlación y el sesgo de estos clasificadores débiles se mantiene baja, se puede reducir el error en el conjunto de prueba.

Un Árbol Aleatorio (RF, *Random Forest*) es un comité de clasificadores débiles para predicción. La predicción de RF, en clasificación es, la mayoría de votos en la clase, no ponderados (Figura A.31). Sea B es el número de árboles de decisión en RF, y $k_1, k_2, ..., k_B$ son las etiquetas de clase. Mientras el número de árboles crece, la tasa del error en el conjunto de prueba, converge a un límite, es decir, no hay sobre-ajuste en RF grandes [124].

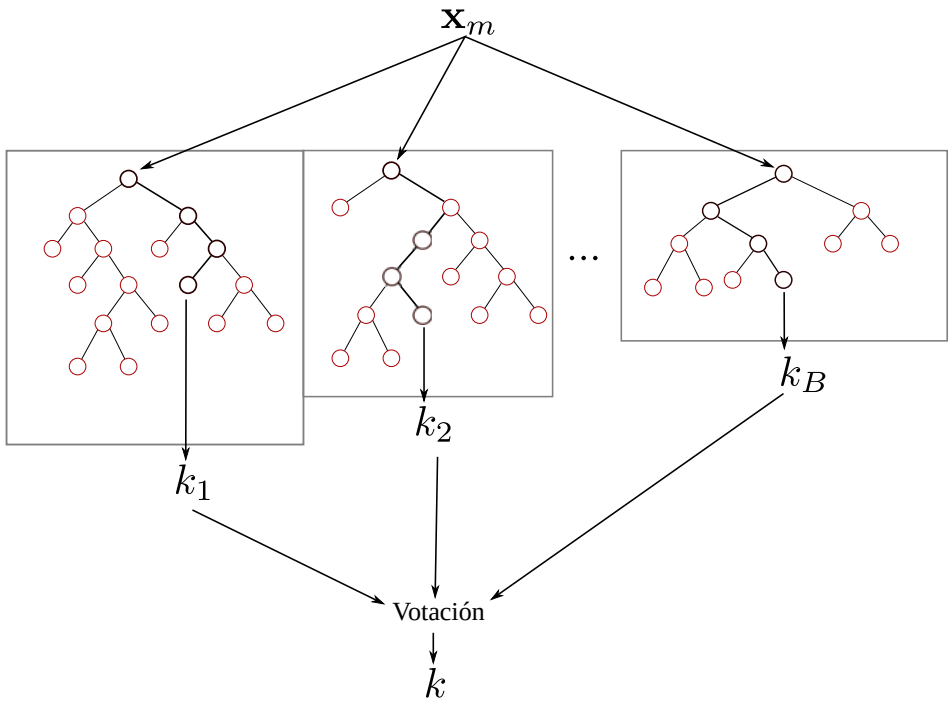


Figura A.31: Arquitectura general de un bosque aleatorio.

Logar un sesgo bajo sesgo y una correlación baja, son esenciales para un mejor desempeño. Para obtener el primero, los árboles crecen a su máxima profundidad. Para alcanzar la segunda se aplica aleatoriedad:

1. Cada árbol del RF, se crea en un muestreo de arranque del conjunto de entrenamiento.

2. Cuando se crea el árbol, en cada nodo, n variables son seleccionadas aleatoriamente de N variables.

3. Por lo general, $n \ll N$ y se sugiere empezar con $n = \lfloor \log_2(N) + 1 \rfloor$ o $n = \sqrt{N}$ y entonces disminuir e incrementar n hasta que el error mínimo del OOB del conjunto de datos es obtenido. En cada nodo, solo una variable, provee la mejor división, usando el mejor n seleccionado.

En RF, n es el único parámetro a seleccionar experimentalmente. También RF puede manejar miles de variables con diferentes tipos con muchos valores ausentes. Para la creación de un árbol de un conjunto de datos de arranque, el conjunto OOB puede ser usado como conjunto de prueba para ese árbol. Mientras crece el número de árboles, RF provee un conjunto de datos OOB insesgado, estimado del error de prueba. El conjunto OOB es usado también, para estimar la importancia de las variables, ambos son ampliamente usados en RF.

Importancia de variables: Hay cuatro medidas de importancia de variables implementadas para RF. Dos medidas basadas en el índice Gini [124], de impureza del nodo y la precisión de clasificación de los datos OOB.

Teniendo un nodo t y estimando su probabilidad de clase $p(k|t), k = 1, ..., Q$, el índice Gini se define como:

$$G(t) = 1 - \sum_{k=1}^{Q} p^2(k|t) \tag{A.84}$$

donde Q es el número de clases. Para calcular el índice basado en Gini, en cada nodo el decremento del índice Gini, es calculado por la variable x_j usada para hacer la división. La medida del índice basado en Gini de la importancia de variable $\overline{\Delta}_j$, está dado por el promedio de decremento del índice Gini en el bosque, donde la variable x_j es usada para dividir un nodo.

El estimador basado en la exactitud de clasificación de la importancia de variable prevalece en varios estudios. Este valor calcula el decremento en la precisión media de clasificación del conjunto OOB. Teniendo un muestreo de arranque con muestras $b = 1, ..., B$, la medida de importancia $\overline{D}_j$ para la variable x_j es calculada de la siguiente manera:

1. Fijar $b = 1$ y encontrar el conjunto OOB $\mathcal{L}_b^{oob}$.

2. Clasificar $\mathcal{L}_b^{oob}$ usando ábol T_b y contar el número de clasificaciones correctas, R_b^{oob}.

3. Para las variables $x_j, j = 1, ..., N$:

   a) permutar los valores de x_j en $\mathcal{L}_b^{oob}$, la permutación resultante va en $\mathcal{L}_{b,j}^{oob}$.

b) Usar T_b para clasificar $\mathcal{L}_{b,j}^{oob}$ y contar el número de clasificaciones correctas, $R_{b,j}^{oob}$.

4. Repetir los pasos 1 al 3, para $b = 2, ..., B$.

5. La medida de importancia $\overline{D}_j$ para la variable x_j está dada por:

$$\overline{D}_j = \frac{1}{B}\sum_{b=1}^{B}(R_b^{oob} - R_{b,j}^{oob}) \tag{A.85}$$

6. Calcular al desviación estándar s_j del decremento en la clasificación correcta y un *z-score*:

$$z_j = \frac{\overline{D}_j}{s_j\sqrt{B}} \tag{A.86}$$

7. Suponiendo que tienen una distribución Gaussiana, convertir z_j a un valor de significancia.

Matriz de proximidad de datos: Esta matriz disponible del RF, es una fuente de información ampliamente usada. Para calcularla en cada árbol creado, los datos recorren cada árbol, si dos observaciones $\mathbf{x}_i$ y $\mathbf{x}_j$ ocupan el mismo nodo terminal en el árbol, $prox(i, j)$ se incrementa en uno. Cuando RF es creado, las proximidades son divididas en el número de árboles en el RF. La proximidad de los datos, puede ser usada para reemplazar los datos ausentes, para encontrar datos atípicos ó datos mal etiquetados y para visualizar datos aplicando escalamiento multidimensional a la matriz de proximidad.

Siglas

ANN red neuronal artificial (*artificial neural network*). 11

CHT transformada de Hough para círculos (*circular Hough transform*). 18

CNN red neuronal convolucional (*convolutional neural network*. 12, 51, 52

CV validación cruzada (*cross-validation*). 57

DL aprendizaje profundo (*deep learning*). 1, 6, 12, 20, 21

HOG histogramas de gradientes orientados (*histogram oriented gradients*). 11

KNN k vecinos cercanos (k *nearest neighbors*). 66, 70

LBP patrones locales binarios (*local binary patterns*). 11

MaL máxima verosimilitud (*maximum likelihood*). 66, 70

ML aprendizaje de máquina (*machine learning*). 1, 4, 6, 8, 11, 16, 20, 30

MLP perceptrón multicapa (*Multi-layer perceptron*). 66, 70

MLR regresión logística multinomial (*multinomial logistic regression*). 66, 70

RF bosque aletorio (*random forest*). 66, 70

SVM máquina de vectores de soporte (*support vector machine*). 8

Bibliografía

[1] Alexander L Fradkov. Early history of machine learning. *IFAC-PapersOnLine*, 53(2):1385–1390, 2020.

[2] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Communications of the ACM*, 60(6): 84–90, 2017.

[3] Thorsten Wuest, Daniel Weimer, Christopher Irgens, and Klaus-Dieter Thoben. Machine learning in manufacturing: advantages, challenges, and applications. *Production & Manufacturing Research*, 4(1):23–45, 2016.

[4] Chih-Pu Dai and Fengfeng Ke. Educational applications of artificial intelligence in simulation-based learning: A systematic mapping review. *Computers and Education: Artificial Intelligence*, page 100087, 2022.

[5] Cecilia S Lee and Aaron Y Lee. Clinical applications of continual learning machine learning. *The Lancet Digital Health*, 2(6):e279–e281, 2020.

[6] Christopher J Fluke and Colin Jacobs. Surveying the reach and maturity of machine learning and artificial intelligence in astronomy. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 10(2):e1349, 2020.

[7] Diego Inácio Patrício and Rafael Rieder. Computer vision and artificial intelligence in precision agriculture for grain crops: A systematic review. *Computers and electronics in agriculture*, 153:69–81, 2018.

[8] Alessandro Matese and Salvatore Filippo Di Gennaro. Technology in precision viticulture: A state of the art review. *International journal of wine research*, 7: 69–81, 2015.

[9] Laura Zabawa, Anna Kicherer, Lasse Klingbeil, Reinhard Töpfer, Heiner Kuhlmann, and Ribana Roscher. Counting of grapevine berries in images via semantic segmentation using convolutional neural networks. *ISPRS Journal of Photogrammetry and Remote Sensing*, 164:73–83, 2020.

[10] Roberto Oberti, Massimo Marchi, Paolo Tirelli, Aldo Calcante, Marcello Iriti, Emanuele Tona, Marko Hočevar, Joerg Baur, Julian Pfaff, Christoph Schütz, et al. Selective spraying of grapevines for disease control using a modular agricultural robot. *Biosystems engineering*, 146:203–215, 2016.

[11] Julio Nogales-Bueno, José Miguel Hernández-Hierro, Francisco José Rodríguez-Pulido, and Francisco José Heredia. Determination of technological maturity of grapes and total phenolic compounds of grape skins in red and white cultivars during ripening by near infrared hyperspectral image: A preliminary approach. *Food Chemistry*, 152:586–591, 2014.

[12] Rodrigo Pérez-Zavala, Miguel Torres-Torriti, Fernando Auat Cheein, and Giancarlo Troni. A pattern recognition strategy for visual grape bunch detection in vineyards. *Computers and Electronics in Agriculture*, 151:136–149, 2018.

[13] Borja Millan, Santiago Velasco-Forero, Arturo Aquino, and Javier Tardaguila. On-the-go grapevine yield estimation using image analysis and boolean model. *Journal of Sensors*, 2018, 2018.

[14] Arturo Aquino, Maria P Diago, Borja Millán, and Javier Tardáguila. A new methodology for estimating the grapevine-berry number per cluster using image analysis. *Biosystems engineering*, 156:80–95, 2017.

[15] Arturo Aquino, Ignacio Barrio, Maria-Paz Diago, Borja Millan, and Javier Tardaguila. vitisberry: An android-smartphone application to early evaluate the number of grapevine berries by means of image analysis. *Computers and Electronics in Agriculture*, 148:19–28, 2018.

[16] Laura Zabawa, Anna Kicherer, Lasse Klingbeil, Andres Milioto, Reinhard Topfer, Heiner Kuhlmann, and Ribana Roscher. Detection of single grapevine berries in images using fully convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 0–0, 2019.

[17] Stephen Nuske, Kyle Wilshusen, Supreeth Achar, Luke Yoder, Srinivasa Narasimhan, and Sanjiv Singh. Automated visual yield estimation in vineyards. *Journal of Field Robotics*, 31(5):837–860, 2014.

[18] Kenneth Dawson-Howe. *A practical introduction to computer vision with opencv*. John Wiley & Sons, 2014.

[19] Jesús R Martínez-Sandoval, Eduardo A Murillo-Bracamontes, Miguel E Martínez-Rosas, Manuel M Miranda Velasco, and Humberto Cervantes De Ávila. Image

processing applied in agriculture. In *Embedded Systems and Wireless Technology*, pages 223–248. CRC Press, 2012.

[20] Rafael C Gonzalez, Richard Eugene Woods, and Steven L Eddins. *Digital image processing using MATLAB*. Pearson Education India, 2004.

[21] Christopher M Bishop. *Pattern recognition and machine learning*. springer, 2006.

[22] Danai Khemasuwan, Jeffrey S Sorensen, and Henri G Colt. Artificial intelligence in pulmonary medicine: computer vision, predictive model and covid-19. *European respiratory review*, 29(157), 2020.

[23] RGV Bramley and APB Proffitt. Managing variability in viticultural production. *Grapegrower and Winemaker*, 427:11–16, 1999.

[24] Alishba Adeel, Muhammad Attique Khan, Muhammad Sharif, Faisal Azam, Jamal Hussain Shah, Tariq Umer, and Shaohua Wan. Diagnosis and recognition of grape leaf diseases: An automated system based on a novel saliency approach and canonical correlation analysis based multiple features fusion. *Sustainable Computing: Informatics and Systems*, 24:100349, 2019.

[25] Scarlett Liu, Mark Whitty, and Stephen Cossell. Automatic grape bunch detection in vineyards for precise yield estimation. In *2015 14th IAPR International Conference on Machine Vision Applications (MVA)*, pages 238–241. IEEE, 2015.

[26] Kiran R Gavhale, Ujwalla Gawande, and Kamal O Hajari. Unhealthy region of citrus leaf detection using image processing techniques. In *International Conference for Convergence for Technology-2014*, pages 1–6. IEEE, 2014.

[27] Suhaili Beeran Kutty, Noor Ezan Abdullah, Hadzli Hashim, Aida Sulinda Kusim, Tuan Norjihan Tuan Yaakub, Puteri Nor Ashikin Megat Yunus, Mohd Fauzi Abd Rahman, et al. Classification of watermelon leaf diseases using neural network analysis. In *2013 IEEE Business Engineering and Industrial Applications Colloquium (BEIAC)*, pages 459–464. IEEE, 2013.

[28] A Camargo and JS Smith. Image pattern classification for the identification of disease causing agents in plants. *Computers and electronics in agriculture*, 66(2): 121–125, 2009.

[29] Sanjeev S Sannakki, Vijay S Rajpurohit, VB Nargund, and Pallavi Kulkarni. Diagnosis and classification of grape leaf diseases using neural networks. In *2013 Fourth International Conference on Computing, Communications and Networking Technologies (ICCCNT)*, pages 1–5. IEEE, 2013.

[30] Manisha Bhange and HA Hingoliwala. Smart farming: Pomegranate disease detection using image processing. *Procedia computer science*, 58:280–288, 2015.

[31] Edna Chebet Too, Li Yujian, Sam Njuki, and Liu Yingchun. A comparative study of fine-tuning deep learning models for plant disease identification. *Computers and Electronics in Agriculture*, 161:272–279, 2019.

[32] Vijai Singh and Ak K Misra. Detection of plant leaf diseases using image segmentation and soft computing techniques. *Information processing in Agriculture*, 4(1):41–49, 2017.

[33] Malay Kishore Dutta, Namita Sengar, Navroj Minhas, Biplab Sarkar, Arnab Goon, and Kaushik Banerjee. Image processing based classification of grapes after pesticide exposure. *LWT-Food Science and Technology*, 72:368–376, 2016.

[34] Armando M Fernandes, Camilo Franco, Ana Mendes-Ferreira, Arlete Mendes-Faia, Pedro Leal da Costa, and Pedro Melo-Pinto. Brix, ph and anthocyanin content determination in whole port wine grape berries by hyperspectral imaging and neural networks. *Computers and Electronics in Agriculture*, 115:88–96, 2015.

[35] Francisco J Rodríguez-Pulido, Luis Gómez-Robledo, Manuel Melgosa, Belén Gordillo, M Lourdes González-Miret, and Francisco J Heredia. Ripeness estimation of grape berries and seeds by image analysis. *Computers and electronics in agriculture*, 82:128–133, 2012.

[36] Eduardo A Murillo-Bracamontes, Miguel E Martinez-Rosas, Manuel M Miranda-Velasco, Horacio L Martinez-Reyes, Jesus R Martinez-Sandoval, and Humberto Cervantes-de Avila. Implementation of hough transform for fruit image segmentation. *Procedia Engineering*, 35:230–239, 2012.

[37] Robert Rudolph, Katja Herzog, Reinhard Töpfer, and Volker Steinhage. Efficient identification, localization and quantification of grapevine inflorescences in unprepared field images using fully convolutional networks. *arXiv preprint arXiv:1807.03770*, 2018.

[38] Pavel Škrabánek and Thomas Philip Runarsson. Detection of grapes in natural environment using support vector machine classifier. In *Proceedings of the 21st International Conference on Soft Computing MENDEL*, pages 143–150, 2015.

[39] Pavel Škrabánek. Deepgrapes: Precise detection of grapes in low-resolution images. *IFAC-PapersOnLine*, 51(6):185–189, 2018.

[40] Esmael Hamuda, Martin Glavin, and Edward Jones. A survey of image processing techniques for plant extraction and segmentation in the field. *Computers and Electronics in Agriculture*, 125:184–199, 2016.

[41] Dong Liu, Yongtao Wang, Zhi Tang, and Xiaoqing Lu. A robust circle detection algorithm based on top-down least-square fitting analysis. *Computers & Electrical Engineering*, 40(4):1415–1428, 2014.

[42] Florent Abdelghafour, Roxana Rosu, Barna Keresztes, Christian Germain, and Jean-Pierre Da Costa. A bayesian framework for joint structure and colour based pixel-wise classification of grapevine proximal images. *Computers and Electronics in Agriculture*, 158:345–357, 2019.

[43] Anna Kicherer, Katja Herzog, Michael Pflanz, Markus Wieland, Philipp Rüger, Steffen Kecke, Heiner Kuhlmann, and Reinhard Töpfer. An automated field phenotyping pipeline for application in grapevine research. *Sensors*, 15(3):4823–4836, 2015.

[44] HP Schwarz, Ph Rüger, A Kicherer, and R Töpfer. Development of an autonomous driven robotic platform used for ht-phenotyping in viticulture. *Mech. Eng. Lett. Szent István Univ*, 10:153–160, 2013.

[45] Anna Kicherer. Bivcolor benchmark, 2015. URL `https://www.openagrar.de/receive/openagrar_mods_00021925`.

[46] Anna Kicherer, Katja Herzog, Nele Bendel, Hans-Christian Klück, Andreas Backhaus, Markus Wieland, Johann Christian Rose, Lasse Klingbeil, Thomas Läbe, Christian Hohl, et al. Phenoliner: a new field phenotyping platform for grapevine research. *Sensors*, 17(7):1625, 2017.

[47] Laura Zabawa and Anna Kicherer. Segmentation of wine berries, Mar 2021. URL `https://www.openagrar.de/receive/openagrar_mods_00067631`.

[48] Tamara L Berg, Alexander Sorokin, Gang Wang, David Alexander Forsyth, Derek Hoiem, Ian Endres, and Ali Farhadi. It's all about the data. *Proceedings of the IEEE*, 98(8):1434–1452, 2010.

[49] Shijie Hao, Yuan Zhou, and Yanrong Guo. A brief survey on semantic segmentation with deep learning. *Neurocomputing*, 406:302–321, 2020.

[50] Kah Phooi Seng, Li-Minn Ang, Leigh M Schmidtke, and Suzy Y Rogiers. Computer vision and machine learning for viticulture technology. *IEEE Access*, 6: 67494–67510, 2018.

[51] Anna Kicherer and J. Christian Rose. Riesling grapevine canopy images for 3d reconstruction, Oct 2016. URL `https://www.openagrar.de/receive/openagrar_mods_00022533`.

[52] Eli Stevens, Luca Antiga, and Thomas Viehmann. *Deep learning with PyTorch*. Manning Publications, 2020.

[53] Vincent Dumoulin and Francesco Visin. A guide to convolution arithmetic for deep learning. *arXiv preprint arXiv:1603.07285*, 2016.

[54] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pages 448–456. PMLR, 2015.

[55] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[56] O. Ronneberger, P.Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation. In *Medical Image Computing and Computer-Assisted Intervention (MICCAI)*, volume 9351 of *LNCS*, pages 234–241. Springer, 2015. URL `http://lmb.informatik.uni-freiburg.de/Publications/2015/RFB15a`. (available on arXiv:1505.04597 [cs.CV]).

[57] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *International Conference on Learning Representations*, 2015.

[58] Vijay Badrinarayanan, Alex Kendall, and Roberto Cipolla. Segnet: A deep convolutional encoder-decoder architecture for image segmentation. *IEEE transactions on pattern analysis and machine intelligence*, 39(12):2481–2495, 2017.

[59] Liang-Chieh Chen, Yukun Zhu, George Papandreou, Florian Schroff, and Hartwig Adam. Encoder-decoder with atrous separable convolution for semantic image segmentation. In *Proceedings of the European conference on computer vision (ECCV)*, pages 801–818, 2018.

[60] Anindya Apriliyanti Pravitasari, Nur Iriawan, Mawanda Almuhayar, Taufik Azmi, Kartika Fithriasari, Santi Wulan Purnami, Widiana Ferriastuti, et al. Unet-vgg16 with transfer learning for mri-based brain tumor segmentation. *Telkomnika*, 18 (3):1310–1318, 2020.

[61] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4510–4520, 2018.

[62] Andres Milioto and Cyrill Stachniss. Bonnet: An open-source training and deployment framework for semantic segmentation in robotics using cnns. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 7094–7100. IEEE, 2019.

[63] Connor Shorten and Taghi M Khoshgoftaar. A survey on image data augmentation for deep learning. *Journal of big data*, 6(1):1–48, 2019.

[64] Upov Ipgri. Oiv. 1997. descriptors for grapevine (vitis spp.). international union for the protection of new varieties of plants, geneva, switzerland/office international de la vigne et du vin, paris, france/international plant genetic resources institute, rome, italy. *This publication is available to download in portable document format from URL: http://www. cgiar. org/ipgri/IPGRI UPOV OIV Via delle Sette Chiese*, 142(34):4.

[65] Ayala-Ramirez Victor, HGC Carlos, Perez-Garcia Arturo, and ESY Raul. Circle detection on images using genetic algorithms. *Pattern Recognit. Lett*, 27(6):652–657, 2006.

[66] Cuneyt Akinlar and Cihan Topal. Edcircles: A real-time circle detector with a false detection control. *Pattern Recognition*, 46(3):725 – 740, 2013. ISSN 0031-3203. doi: https://doi.org/10.1016/j.patcog.2012.09.020. URL `http://www.sciencedirect.com/science/article/pii/S0031320312004268`.

[67] Miguel R González, Miguel E Martínez, María Cosío-León, Humberto Cervantes, and Carlos A Brizuela. Multiple circle detection in images: a simple evolutionary algorithm approach and a new benchmark of images. *Pattern Analysis and Applications*, 24(4):1583–1603, 2021.

[68] Teh-Chuan Chen and Kuo-Liang Chung. An efficient randomized algorithm for detecting circles. *Computer vision and image understanding*, 83(2):172–191, 2001.

[69] Erik Cuevas, Felipe Sención-Echauri, Daniel Zaldivar, and Marco Pérez-Cisneros. Multi-circle detection on images using artificial bee colony (abc) optimization. *Soft Computing*, 16(2):281–296, 2012.

[70] Erik Cuevas, Fernando Wario, Daniel Zaldivar, and Marco Pérez-Cisneros. Circle detection on images using learning automata. *Artificial Intelligence, Evolutionary Computing and Metaheuristics*, pages 545–570, 2013.

[71] Na Dong, Chun-Ho Wu, Wai-Hung Ip, Zeng-Qiang Chen, Ching-Yuen Chan, and Kai-Leung Yung. An opposition-based chaotic ga/pso hybrid algorithm and its application in circle detection. *Computers & Mathematics with Applications*, 64 (6):1886–1902, 2012.

[72] Alan López and Francisco J Cuevas. Automatic multi-circle detection on images using the teaching learning based optimisation algorithm. *IET Computer Vision*, 12(8):1188–1199, 2018.

[73] Jianping Wu, Ke Chen, and Xiaohui Gao. Fast and accurate circle detection using gradient-direction-based segmentation. *JOSA A*, 30(6):1184–1192, 2013.

[74] Jim R Parker. *Algorithms for image processing and computer vision*. John Wiley & Sons, 2010.

[75] John Canny. A computational approach to edge detection. *IEEE Transactions on pattern analysis and machine intelligence*, 8(6):679–698, 1986.

[76] Saket Bhardwaj and Ajay Mittal. A survey on various edge detector techniques. *Procedia Technology*, 4:220–226, 2012.

[77] Cuneyt Akinlar and Cihan Topal. Edpf: a real-time parameter-free edge segment detector with a false detection control. *International Journal of Pattern Recognition and Artificial Intelligence*, 26(01):1255002, 2012.

[78] Bing Zhou and Yang He. Fast circle detection using spatial decomposition of hough transform. *International Journal of Pattern Recognition and Artificial Intelligence*, 31(03):1755006, 2017.

[79] Bodi Yuan and Min Liu. Power histogram for circle detection on images. *Pattern Recognition*, 48(10):3268–3280, 2015.

[80] Hervé Chauris, Imen Karoui, Pierre Garreau, Hans Wackernagel, Philippe Craneguy, and L Bertino. The circlet transform: A robust tool for detecting features with circular shapes. *Computers & geosciences*, 37(3):331–342, 2011.

[81] Hanqing Zhang, Krister Wiklund, and Magnus Andersson. A fast and robust circle detection method using isosceles triangles sampling. *Pattern Recognition*, 54:218–228, 2016.

[82] Li-qin Jia, Cheng-zhang Peng, Hong-Min Liu, and Zhi-Heng Wang. A fast randomized circle detection algorithm. In *Image and Signal Processing (CISP), 2011 4th International Congress on*, volume 2, pages 820–823. IEEE, 2011.

[83] John Henry Holland. *Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence*. MIT press, 1992.

[84] David A. Coley. *An Introduction to Genetic Algorithms for Scientists and Engineers*. World Scientific, har/dskt edition, 1997. ISBN 9810236026,9789810236021,9789812386359.

[85] Fabrice Mairesse, Tadeusz Sliwa, Stéphane Binczak, and Yvon Voisin. Subpixel determination of imperfect circles characteristics. *Pattern Recognition*, 41(1):250–271, 2008.

[86] Farid Garcia-Lamont, Jair Cervantes, Asdrúbal López, and Lisbeth Rodriguez. Segmentation of images by color features: A survey. *Neurocomputing*, 292:1–27, 2018.

[87] Opencv: Morphological transformations. `https://docs.opencv.org/master/d9/d61/tutorial_py_morphological_ops.html`, .

[88] Opencv: Hough circle transform. `https://docs.opencv.org/3.4/d4/d70/tutorial_hough_circle.html`, . (Accessed on 05/22/2022).

[89] Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. How transferable are features in deep neural networks? *arXiv preprint arXiv:1411.1792*, 2014.

[90] Mateusz Buda, Atsuto Maki, and Maciej A Mazurowski. A systematic study of the class imbalance problem in convolutional neural networks. *Neural Networks*, 106:249–259, 2018.

[91] Steve Lawrence, Ian Burns, Andrew Back, Ah Chung Tsoi, and C Lee Giles. Neural network classification and prior class probabilities. In *Neural networks: Tricks of the trade*, pages 295–309. Springer, 2012.

[92] Randy Bullock. Least-squares circle fit. *Developmental testbed center*, 3, 2006.

[93] Tom Fawcett. An introduction to roc analysis. *Pattern recognition letters*, 27(8): 861–874, 2006.

[94] Sylvain Arlot, Alain Celisse, et al. A survey of cross-validation procedures for model selection. *Statistics surveys*, 4:40–79, 2010.

[95] 3.1. cross-validation: evaluating estimator performance — scikit-learn 0.22.2 documentation. `https://scikit-learn.org/stable/modules/cross_validation.html`.

[96] S. García, D. Molina, M. Lozano, and F Herrera. A study on the use of non-parametric tests for analyzing the evolutionary algorithms' behaviour: a case study on the cec'2005 special session on real parameter optimization. *J Heuristics*, 15:617–644, 2009.

[97] MR González-Márquez, CA Brizuela, ME Martínez-Rosas, and H Cervantes. Grape bunch detection using a pixel-wise classification in image processing. In *2020 IEEE International Autumn Meeting on Power, Electronics and Computing (ROPEC)*, volume 4, pages 1–6. IEEE, 2020.

[98] Dorothy Cheng and Edmund Y Lam. Transfer learning u-net deep learning for lung ultrasound segmentation. *arXiv preprint arXiv:2110.02196*, 2021.

[99] Chirag Balakrishna, Sarshar Dadashzadeh, and Sara Soltaninejad. Automatic detection of lumen and media in the ivus images using u-net with vgg16 encoder. *arXiv preprint arXiv:1806.07554*, 2018.

[100] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 2961–2969, 2017.

[101] Maria Cecilia do Nascimento Nunes. *Color atlas of postharvest quality of fruits and vegetables*. John Wiley & Sons, 2009.

[102] David M Woebbecke, George E Meyer, Kenneth Von Bargen, and David A Mortensen. Plant species identification, size, and enumeration using machine vision techniques on near-binary images. In *Optics in Agriculture and Forestry*, volume 1836, pages 208–219. International Society for Optics and Photonics, 1993.

[103] David M Woebbecke, George E Meyer, Kenneth Von Bargen, and David A Mortensen. Color indices for weed identification under various soil, residue, and lighting conditions. *Transactions of the ASAE*, 38(1):259–269, 1995.

[104] George E Meyer, Timothy W Hindman, and Koppolu Laksmi. Machine vision detection parameters for plant species identification. In *Precision agriculture and biological quality*, volume 3543, pages 327–335. International Society for Optics and Photonics, 1999.

[105] Takashi Kataoka, Toshihiro Kaneko, Hiroshi Okamoto, and S Hata. Crop growth estimation system using machine vision. In *Proceedings 2003 IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM 2003)*, volume 2, pages b1079–b1083. IEEE, 2003.

[106] George E Meyer, Joao Camargo Neto, David D Jones, and Timothy W Hindman. Intensified fuzzy clusters for classifying plant, soil, and residue regions of interest from color images. *Computers and electronics in agriculture*, 42(3):161–180, 2004.

[107] E Raymond Hunt, Michel Cavigelli, Craig ST Daughtry, James E Mcmurtrey, and Charles L Walthall. Evaluation of digital photography from model aircraft for remote sensing of crop biomass and nitrogen status. *Precision Agriculture*, 6 (4):359–378, 2005.

[108] T Hague, ND Tillett, and H Wheeler. Automated crop and weed monitoring in widely spaced cereals. *Precision Agriculture*, 7(1):21–32, 2006.

[109] Xavier P Burgos-Artizzu, Angela Ribeiro, Maria Guijarro, and Gonzalo Pajares. Real-time image processing for crop/weed discrimination in maize fields. *Computers and Electronics in Agriculture*, 75(2):337–346, 2011.

[110] Maria Guijarro, Gonzalo Pajares, Isabel Riomoros, PJ Herrera, XP Burgos-Artizzu, and Angela Ribeiro. Automatic segmentation of relevant textures in agricultural images. *Computers and Electronics in Agriculture*, 75(1):75–83, 2011.

[111] José Miguel Guerrero, Gonzalo Pajares, Martín Montalvo, Juan Romeo, and María Guijarro. Support vector machines for crop/weeds identification in maize fields. *Expert Systems with Applications*, 39(12):11149–11155, 2012.

[112] Loris Nanni, Alessandra Lumini, and Sheryl Brahnam. Survey on lbp based texture descriptors for image classification. *Expert Systems with Applications*, 39 (3):3634–3641, 2012.

[113] G Lowe. Sift-the scale invariant feature transform. *Int. J*, 2(91-110):2, 2004.

[114] Jyotismita Chaki, Nilanjan Dey, Luminiţa Moraru, and Fuqian Shi. Fragmented plant leaf recognition: Bag-of-features, fuzzy-color and edge-texture histogram descriptors with multi-layer perceptron. *Optik*, 181:639–650, 2019.

[115] Andrew R Webb. *Statistical pattern recognition*. John Wiley & Sons, 2003.

[116] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(Nov):2579–2605, 2008.

[117] Girish Chandrashekar and Ferat Sahin. A survey on feature selection methods. *Computers & Electrical Engineering*, 40(1):16–28, 2014.

[118] Hooman H Rashidi, Nam K Tran, Elham Vali Betts, Lydia P Howell, and Ralph Green. Artificial intelligence and machine learning in pathology: the present landscape of supervised methods. *Academic pathology*, 6:2374289519873088, 2019.

[119] Heng-Da Cheng, X_ H_ Jiang, Ying Sun, and Jingli Wang. Color image segmentation: advances and prospects. *Pattern recognition*, 34(12):2259–2281, 2001.

[120] Jun Wang and Jean-Daniel Zucker. Solving multiple-instance problem: A lazy learning approach. 2000.

[121] Richard O Duda, Peter E Hart, and David G Stork. *Pattern classification*. John Wiley & Sons, 2012.

[122] Roger Penrose. A generalized inverse for matrices. In *Mathematical proceedings of the Cambridge philosophical society*, volume 51, pages 406–413. Cambridge University Press, 1955.

[123] Bernhard Scholkopf and Alexander J Smola. *Learning with kernels: support vector machines, regularization, optimization, and beyond*. MIT press, 2001.

[124] Antanas Verikas, Adas Gelzinis, and Marija Bacauskiene. Mining data with random forests: A survey and results of new tests. *Pattern recognition*, 44(2):330–349, 2011.