Universidad Autónoma de Yucatán

Facultad de Matemáticas

Utilización del aprendizaje profundo para analizar automáticamente imágenes obtenidas por el ensayo cometa (electroforesis de ADN de células únicas)

T E S I S

QUE PARA OPTAR POR EL GRADO DE:

Maestro en Ciencias de la Computación

PRESENTA:

Ismel Rizo Blanco

ASESORES DE TESIS:
Dr. Carlos Brito Loeza
Dra. Anabel Martín González

Mérida, Yucatán, 2023

A la Facultad de Matemáticas.

Reconocimientos

Quiero agradecer:

- A mis padres, Axinia Blanco y Orlando Díaz, quienes siempre han estado a mi lado brindándome su amor y apoyo incondicional para seguir adelante en cada etapa de mi vida. Gracias por creer en mí y por ser mi fuente constante de inspiración.
- A mis tutores, Dra. Anabel Martín González y Dr. Carlos Brito Loeza, por su dedicación, paciencia y orientación durante el desarrollo de esta tesis. Su conocimiento y experiencia han sido invaluables para alcanzar los objetivos propuestos.
- A mis amigos, quienes han sido una parte fundamental en mi camino. Su compañía, consejos y risas han hecho que cada día sea especial y significativo.
- A todas aquellas personas que de alguna u otra forma contribuyeron en este proceso, brindando su colaboración y apoyo.

Resumen

La electroforesis en gel de células individuales, también conocida como ensayo Cometa, es una técnica que permite evaluar el daño en el Ácido Desoxirribonucleico (ADN) de células individuales. Debido a su asequibilidad, facilidad de uso y confiabilidad, esta técnica se aplica ampliamente en campos como el monitoreo de la salud humana, la evaluación de la genotoxicidad y la investigación de enfermedades degenerativas como el cáncer y la diabetes. Durante el ensayo Cometa, cada célula es sometida a rotura de sus membranas y electroforesis, luego de lo cual da lugar a una estructura que se asemeja a la de un cometa, con una región compacta que representa el ADN no dañado, conocida como la cabeza del cometa, y otra región que consiste en fragmentos de ADN altamente dañados, denominada la cola del cometa.

El daño celular ocasiona una amplia variabilidad en el área, longitud y contenido de ADN de los cometas, tanto en sus cabezas como en sus colas. Esta diversidad resalta la importancia de identificar y analizar minuciosamente cada célula de manera individual para obtener las características específicas necesarias en diversas aplicaciones.

Los resultados de este ensayo requieren un análisis por una o más personas expertas, quienes deberán emplear tiempo para clasificar detalladamente los cometas en las imágenes microscópicas obtenidas con la posibilidad de que existan sesgos involuntarios y muchas veces sin conocer a que tratamiento corresponde cada conteo. Por lo cual, el objetivo de esta investigación es desarrollar un sistema computacional, basado en la utilización de una red neuronal convolucional para la detección y clasificación automática de cometas en imágenes procesadas por el ensayo Cometa de acuerdo al nivel de daño de ADN que presenten.

Índice general

Índice de figuras

Índice de tablas

Capítulo 1

Introducción

El ácido desoxirribonucleico, comúnmente conocido como ADN, es una molécula esencial que almacena y transmite la información genética en los organismos vivos. Es considerado la molécula de la vida debido a su papel fundamental en la herencia genética y la determinación de las características heredadas de un organismo [1]. Su estructura en forma de doble hélice contiene las instrucciones necesarias para el desarrollo, funcionamiento y reproducción de todos los seres vivos.

En el campo de la genética y la biología molecular, la evaluación del daño en el ADN ha sido objeto de gran interés y relevancia. El ADN, como la molécula portadora de la información genética, se encuentra expuesto a diversos factores genotóxicos [2], citotóxicos, mutagénicos y cancerígenos [3] que pueden ocasionar alteraciones en su estructura y funcionalidad, lo cual tiene implicaciones significativas en diferentes áreas, como la toxicología [4], la biotecnología [5] y la genética [6].

En la década de los 80, los métodos convencionales para evaluar el daño en el ADN incluían la electroforesis alcalina y la microscopía de fluorescencia. Estos permitían detectar roturas de cadena en el ADN, pero presentaban limitaciones en términos de sensibilidad y capacidad para evaluar el daño a nivel individual de la célula [7]. Además, requerían grandes cantidades de ADN y eran laboriosos en términos de procesamiento y análisis.

El ensayo Cometa, también conocido como electroforesis alcalina de células individuales, fue introducido por primera vez por Ostling y Johanson [8] en 1984 como un método revolucionario para la evaluación directa y cuantitativa del daño en el ADN a nivel individual de la célula. Este método, realizado en condiciones alcalinas, se destacó por su versatilidad [9] detectando daño en el ADN sin importar su origen o causa y tambien sobresalió por su alta sensibilidad y rapidez para identificar y evaluar el grado de daño en el ADN causado por diferentes motivos, como afectaciones en algún sistema de reparación originado por el daño metabólico o endógeno, radiación, productos químicos y otros factores ambientales [10].

Esta técnica provoca la migración de las moléculas de ADN a través de una matriz de agarosa, en donde el ADN se separa en dos regiones distintas, formando una estructura similar a la de un cometa del espacio (ver Figura 1.1). La cabeza del cometa se refiere a la región en la que se encuentra el ADN intacto o con daños mínimos. Por otro lado, la cola del cometa está compuesta por fragmentos de ADN dañados que se han separado

de la cabeza durante la migración [11].

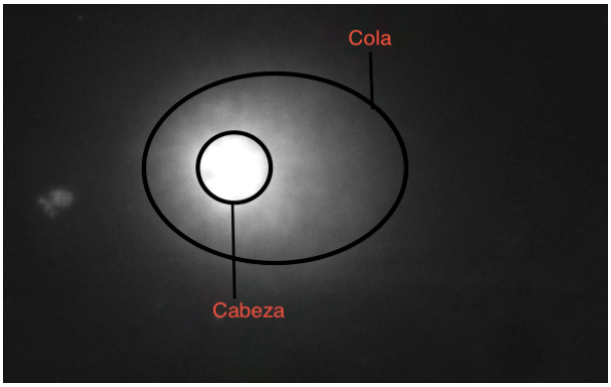


Figura 1.1: Regiones compuestas por una célula procesada por el ensayo Cometa

Los parámetros obtenidos de las células, luego de aplicar esta técnica, son de gran utilidad en el estudio de enfermedades degenerativas como el cáncer y la diabetes [12]. Estos parámetros se podrían calcular de forma manual a través de un microscopio, pero en los últimos años se han propuesto diversos sistemas informáticos capaces de automatizar la obtención de dichos parámetros.

1.1. Antecedentes

En las últimas décadas ha aumentado la popularidad de la técnica del ensayo Cometa para la detección de daños en el ADN, ya que es un examen rápido, versátil, sencillo y económico [13]. Para aprovechar al máximo el potencial de esta técnica, se han desarrollado diversos sistemas informáticos destinados a facilitar la recopilación, análisis y visualización de los resultados obtenidos a través del ensayo Cometa.

Uno de los primeros sistemas en procesar imágenes de ADN generadas por el ensayo Cometa fue CometScore, el cual sólo se puede utilizar en el sistema operativo Windows. CometScore (2007) [14] es una herramienta de software libre creada con el objetivo de puntuar cometas producidos por el proceso de electroforesis en gel, de células únicas. Esta herramienta fue una de las primeras creadas con el objetivo de procesar imágenes generadas por la técnica del ensayo Cometa para conocer qué tan dañado puede estar un ADN, sólo puede ser utilizada en el sistema operativo Windows. Una vez terminado el procesamiento de la imagen, el sistema retorna una hoja de cálculo con los parámetros individuales de cada Cometa que identificó en la imagen introducida.

OpenComet(2014) es una herramienta de software libre, desarrollada específicamente para abordar las dificultades en el análisis automático de imágenes en la técnica del ensayo Cometa [15]. OpenComet se implementa como un *plug-in* para la popular plataforma de procesamiento de imágenes, ImageJ [16]. El software ImageJ puede mostrar, procesar y analizar imágenes, y es principalmente diseñado para su uso con imágenes microscópicas. Este sistema tiene un algoritmo que se divide en dos partes, una encar-

gada de detectar los cometas denominada *comet finding* y otra que se ocupa de detectar las cabezas de los cometas llamada *head finding*.

El sistema demuestra un buen rendimiento en el análisis de imágenes; sin embargo, se ha observado que el área designada como cabeza del cometa tiende a ser significativamente mayor que su tamaño real. Además, se ha notado que el sistema en ocaciones elimina cometas que podrían haber sido reconocidos y existen casos, donde reconoció cometas que se encontraban en otro plano de forma incorrecta [17], lo que puede afectar la precisión de las mediciones. El software y el código fuente de OpenComet están disponibles en `https://cometbio.org`, así como las instrucciones para instalar y usar la herramienta. OpenComet es un software independiente y multiplataforma y, por lo tanto, funciona en Windows, Mac OS X y distribuciones del sistema Linux [15].

CometQ (2016) es una herramienta automatizada que surge para la detección y cuantificación de daños en el ADN mediante el análisis de imágenes de ensayo Cometa [18]. En este sistema se emplean cuatro diferentes etapas: clasificación, segmentación, partición y cuantificación y utiliza la técnica llamada agrupamiento difuso modificado (*modified fuzzy clustering*) para la segmentación de cometas. La automatización en la selección del método de segmentación, adaptado a las características de la imagen, ha demostrado producir resultados efectivos en la diferenciación entre los cometas y el fondo. No obstante, la clasificación de las diversas partes del cometa mediante el algoritmo de agrupamiento difuso presenta ciertos desafíos, ya que las fronteras que delimitan las regiones de cada clase tienden a mostrarse altamente irregulares. Este sistema es de código abierto y fue desarrollado en MATLAB.

HiComet (2018) es una herramienta computacional con el propósito de facilitar el análisis de datos de alto rendimiento del ensayo Cometa. Dada una imagen ruidosa con varios cometas colocados arbitrariamente, la herramienta puede reconocer cometas normales y dañados de una manera totalmente automatizada [19]. La metodología utilizada por este sistema consta de cuatro pasos principales: preprocesamiento, binarización, filtrado y corrección de superposición y por último caracterización y clasificación. Este sistema representa una de las pocas soluciones capaces de segmentar cometas que presentan superposición, pero esta capacidad está limitada a cometas que exhiben simetría en relación con el eje horizontal. A pesar de su potencial, lamentablemente, no existe una guía de usuario ni información sobre cómo ejecutar el código que se encuentra en el repositorio de GitHub. Además, no hay una versión independiente disponible, y el autor del documento de referencia ha afirmado que HiComet ya no se mantiene en la actualidad. En consecuencia, su ejecución se ve obstaculizada por la falta de documentación y mantenimiento [20].

Deep Comet(2020) es un método que utiliza aprendizaje profundo para la segmentación de cometas [21]. Este método se basa en el uso de redes neuronales convolucionales a partir del modelo Mask R-CNN [22], desarrollada por Facebook AI Research y el *framework* Pytorch [23] para su implementación. Luego de realizar la segmentación, asigna una puntuación a cada cometa encontrado a partir de las siguientes características: el porcentaje de ADN en la cola, el momento de la cola (longitud de la cola multiplicada por el porcentaje de ADN en la cola) y el momento de Olive (distancia entre los centros

de la cabeza y la cola por la tasa de ADN en la cola). El método propuesto presenta buenos resultados en la segmentación y puntuación de cometas con respecto a otros sistemas como el OpenComet [15] y el HiComet [19].Sin embargo, es notable que en las imágenes proporcionadas, la diversidad de cometas es limitada, y en la mayoría de los casos, la cabeza del cometa tiende a ubicarse en el extremo izquierdo.
Namuduri et al., (2021) presentan un nuevo enfoque para cuantificar el grado de daño del ADN, combinando aprendizaje profundo [24] con un método riguroso y completo para optimizar los hiperparámetros con la ayuda de pruebas estadísticas [25]. La red neuronal presentada implica un uso novedoso de la arquitectura VGG19 y Multi Task Learning(MTL) para calcular los intervalos de confianza. Esta arquitectura se comparó con MTL-VGG19 con respecto a los intervalos de confianza(CI), obteniendo los siguientes resultados para 30 ejecuciones de pruebas: VGG19(1.89) y MTL-VGG19(1.36). Los resultados obtenidos revelaron una mejora notable en la rapidez y precisión del análisis de imágenes, lo que permite un procesamiento más eficiente de grandes conjuntos de datos. Sin embargo, a pesar de los avances logrados, el sistema aún presenta un desafío en cuanto a la segmentación precisa del ADN en las imágenes de los cometas. La variabilidad en la apariencia de los cometas, así como la presencia de ruido y otros artefactos, pueden afectar la exactitud de la segmentación y, en consecuencia, influir en las mediciones posteriores.

1.2. Objetivos

1.2.1. Objetivo General

El objetivo general de este proyecto es la implementación de un sistema computacional basado en la utilización de una red neuronal convolucional para la detección y clasificación de células procesadas por ensayo cometa de acuerdo al nivel de daño de ADN que presenten.

1.2.2. Objetivos Específicos

Los objetivos específicos de este proyecto son los siguientes:

- Crear la base de datos a utilizar por la red neuronal para su entrenamiento y validación.
- Implementar y entrenar la red neuronal convolucional para la detección y clasificación de células en imágenes procesadas por el ensayo cometa.
- Evaluar el desempeño del modelo de la red neuronal diseñada.
- Comparar el modelo propuesto con otros métodos existentes.
- Desarrollar una aplicación web para integrarla con el modelo propuesto.

1.3. Contribuciones

- La principal contribución de este trabajo es el desarrollo de un modelo de detección automática de células procesadas por el ensayo Cometa basado en redes neuronales convolucionales, específicamente *You Only Look Once (YOLO)* versión 5.
- El trabajo también realiza la evaluación y comparación del rendimiento del modelo propuesto con respecto a otros métodos de detección de células procesadas por el ensayo Cometa.
- Finalmente, se implemena la integración del modelo propuesto en una página web, lo que permitiría a los investigadores acceder al modelo de manera remota para obtener resultados sobre la detección automática de células procesadas por el ensayo Cometa.

Capítulo 2

Marco Teórico

En este capítulo, se presentarán las bases teóricas y conceptuales necesarias para entender el enfoque de este trabajo de tesis. En primer lugar, se explicará la técnica del ensayo Cometa, su principio de funcionamiento y su aplicación en el campo de la genotoxicidad. Luego, se profundiza en los fundamentos de las ecuaciones diferenciales ordinarias. Posteriormente, se presenta una descripción detallada de las redes neuronales convolucionales y se explican los conceptos necesarios para entender su funconamiento. Finalmente, se presentan las métricas de desempeño utilizadas para evaluar la calidad del modelo.

2.1. Ensayo Cometa

El ensayo Cometa, también conocido como electroforesis alcalina de células individuales, fue introducido por primera vez por Ostling y Johanson en 1984 como una técnica microelectroforética para la visualización directa del daño del ADN en individuos [8]. Para llevar a cabo esta técnica, las células extraídas al individuo o provenientes de cultivos celulares se suspenden en agarosa y se depositan sobre un portaobjetos. Luego, se realiza una lisis celular para liberar el ADN y se somete a un campo eléctrico que provoca que el ADN se desenrolle y migre hacia el ánodo o polo positivo, formando una cola en forma de cometa. El proceso se muestra en la figura 2.1.

Esta técnica permite clasificar el tipo de daño en diferentes categorías en función de la morfología y extensión de la cola observada en los cometas. Se ha establecido una clasificación en cinco clases principales para describir los diferentes tipos de daño en el ADN [11].

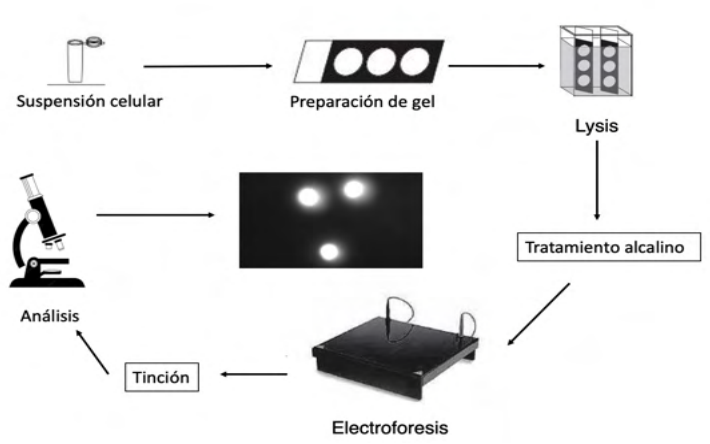


Figura 2.1: Pasos para realizar la técnica del ensayo Cometa

A continuación, se describen las clases definidas según el daño que presentan las células procedas po rel ensayo Cometa:

1. Clase 1: Sin daño o daño mínimo. En esta clase, no se observa ningún daño en el ADN o se observa un daño muy leve en el ADN. La imagen del cometa puede mostrar una cabeza compacta sin ninguna extensión de la cola o una cabeza compacta con una pequeña extensión de la cola y en caso de existir halo, es simétrico.

Figura 2.2: Clase 1

2. Clase 2: Daño moderado. En esta clase, se observa un daño moderado en el ADN. La imagen del cometa muestra una cabeza más pequeña en comparación con la clase 1 y una cola más extendida. El diámetro de la cabeza debe ser menor que la longitud de la cola.

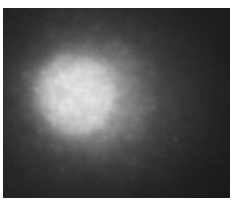


Figura 2.3: Clase 2

3. Clase 3: Daño importante. En esta clase, se observa un daño significativo en el ADN. La imagen del cometa muestra una cabeza pequeña, y restos del ADN en la cola. En esta clase el diámetro de la cabeza es igual a la longitud de la cola, lo que implica que el diámetro de la cabeza equivale a la mitad del largo total.

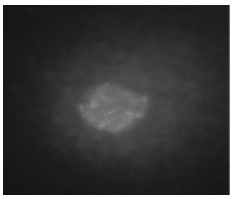


Figura 2.4: Clase 3

4. Clase 4: Daño grave. En esta clase, se observa un daño grave en el ADN. La imagen del cometa muestra una pequeña cabeza y una cola con mayor longitud. En esta clase el diámetro de la cabeza es inferior a la mitad de la longitud total del cometa.

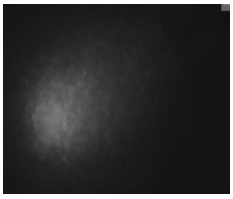


Figura 2.5: Clase 4

5. Clase 5: Daño máximo. En esta clase, se observa un daño máximo en el ADN. La imagen del cometa muestra una dispersión intensa de ADN en toda el área analizada, sin una estructura de cabeza y cola claramente definida.

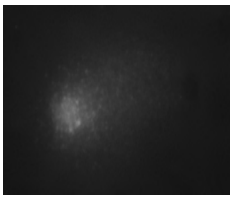


Figura 2.6: Clase 5

La clasificación en estas cinco clases permite una evaluación más detallada del daño en el ADN y ayuda a distinguir entre diferentes niveles de genotoxicidad. Esto es especialmente importante en estudios de toxicología y evaluación de riesgos, donde se requiere una comprensión precisa de la naturaleza y la gravedad del daño en el ADN.

2.2. Ecuaciones diferenciales ordinarias

Las ecuaciones diferenciales ordinarias son una parte fundamental del análisis numérico y se utilizan para describir el comportamiento de sistemas dinámicos en los que la función desconocida depende de una sola variable independiente. Según [26], una ecuación diferencial ordinaria se define como:

$$\frac{dy}{dx} = f(x, y) \tag{2.1}$$

donde $\frac{dy}{dx}$ representa la derivada de la función desconocida y con respecto a la variable independiente x, $f(x, y)$ es una función conocida que describe la relación entre las variables x e y.

El objetivo en las ecuaciones diferenciales ordinarias es encontrar una solución $y(x)$ que satisfaga la ecuación diferencial dada junto con ciertas condiciones iniciales. Para resolver estas ecuaciones, existen diversos métodos numéricos, como los métodos de Euler, de Heun, de Runge-Kutta, entre otros.

El análisis numérico de ecuaciones diferenciales ordinarias es fundamental en ciencias de la computación, ya que permite modelar y simular sistemas dinámicos en diversos campos, como la física, la biología, la economía y la ingeniería.

2.2.1. Método de Euler

Según [26], el Método de Euler es uno de los métodos numéricos más simples y ampliamente utilizados para resolver ecuaciones diferenciales ordinarias. Este método aproxima la solución de una ecuación diferencial en un intervalo dado, dividiendo el intervalo en pasos discretos y calculando el valor de la función desconocida en cada paso. El Método de Euler se basa en la aproximación de la derivada utilizando la fórmula del cociente incremental.

A continuación, se muestra el proceso para resolver ecuaciones diferenciales ordinarias utilizando el Método de Euler:

1. Dado un problema de ecuación diferencial ordinaria de primer orden (ver ecuación 2.1).

2. Elige un punto inicial x_0 y su correspondiente valor de la función desconocida y_0.

3. Divide el intervalo de interés en varios subintervalos de igual tamaño. Estos subintervalos están determinados por el tamaño del paso h que deseas utilizar.

4. Utiliza la fórmula del Método de Euler para actualizar la aproximación de la solución en cada paso. La fórmula de actualización es:

   $$y_{i+1} = y_i + h \cdot f(x_i, y_i) \tag{2.2}$$

   donde y_i es la aproximación de la solución en el punto x_i, h es el tamaño del paso y $f(x_i, y_i)$ es la función que describe la relación entre las variables x e y en ese punto.

5. Repite el paso 4 para cada subintervalo, utilizando el valor obtenido en el paso anterior para calcular el siguiente valor de la función.

6. Continúa el proceso hasta alcanzar el punto final deseado x_f.

Es importante destacar que el Método de Euler puede introducir errores acumulativos a medida que se realizan más iteraciones y el tamaño del paso h se vuelve más pequeño. Por lo tanto, su precisión puede ser limitada en comparación con otros métodos numéricos [26].

2.2.2. Método del Trapecio Explícito o Método de Heun

El Método del Trapecio Explícito [26], también conocido como Método de Heun, es un método numérico utilizado para resolver ecuaciones diferenciales ordinarias. El Método de Heun se basa en la aproximación de la solución mediante segmentos de recta trazados en el plano (t, y), donde t representa la variable independiente y y representa la variable dependiente. En lugar de utilizar la pendiente en el extremo izquierdo del intervalo, como en el Método de Euler, el Método del Trapecio Explícito utiliza el promedio de las pendientes en ambos extremos del intervalo.

A continuación, se muestra el proceso para resolver ecuaciones diferenciales ordinarias utilizando el Método de Heun:

1. Definir el intervalo de interés $[a, b]$ en el cual se desea aproximar la solución de la ecuación diferencial.

2. Dividir el intervalo $[a, b]$ en N subintervalos de igual tamaño, donde N es un número entero.

3. Calcular el tamaño del paso h mediante la fórmula $h = \frac{b-a}{N}$.

4. Establecer la condición inicial y_0 en el punto $t = a$.

5. Utilizando la fórmula del Método del Trapecio Explícito, calcular las aproximaciones $y_1, y_2, \ldots, y_N$ de la solución en los puntos $t_1, t_2, \ldots, t_N$, respectivamente. La fórmula es la siguiente:

$$y_{i+1} = y_i + \frac{h}{2}\left(f(t_i, y_i) + f\left(t_i + h, y_i + h \cdot f(t_i, y_i)\right)\right) \tag{2.3}$$

   donde y_i es la aproximación de la solución en el punto t_i, $t_{i+1} = t_i + h$ es el siguiente punto en el intervalo, y $f(t, y)$ es la función que describe la relación entre las variables t y y en la ecuación diferencial.

Al repetir estos pasos para todos los subintervalos, se obtiene una aproximación de la solución de la ecuación diferencial en el intervalo $[a, b]$.

El Método del Trapecio Explícito ofrece una mejora con respecto al Método de Euler al considerar el promedio de las pendientes en ambos extremos del intervalo, lo cual

proporciona una mayor precisión en la aproximación de la solución. Sin embargo, cabe destacar que este método puede requerir un mayor número de cálculos en comparación con el Método de Euler [26].

2.2.3. Método de Runge-Kutta de orden cuatro (RK4)

El Método de Runge-Kutta de orden cuatro (RK4) [26] es un método numérico utilizado para aproximar la solución de ecuaciones diferenciales ordinarias. Es uno de los métodos más utilizados debido a su precisión y eficiencia. El RK4 utiliza una combinación ponderada de pendientes en diferentes puntos dentro de cada intervalo de paso para calcular las aproximaciones sucesivas de la solución.

A continuación, se muestra el proceso para resolver ecuaciones diferenciales ordinarias utilizando el Método de RK4:

1. Definir el intervalo de interés $[a, b]$ en el cual se desea aproximar la solución de la ecuación difrencial ordinaria.
2. Dividir el intervalo $[a, b]$ en N subintervalos de igual tamaño, donde N es un número entero.
3. Calcular el tamaño del paso h mediante la fórmula $h = \frac{b-a}{N}$.
4. Establecer la condición inicial w_0 en el punto $t = a$.
5. Para cada $i = 0, 1, \ldots, N-1$, realizar los siguientes cálculos:
   a. Calcular los valores de las pendientes:

$$s_1 = f(t_i, w_i)$$

$$s_2 = f\left(t_i + \frac{h}{2}, w_i + \frac{h}{2}s_1\right)$$

$$s_3 = f\left(t_i + \frac{h}{2}, w_i + \frac{h}{2}s_2\right)$$

$$s_4 = f(t_i + h, w_i + hs_3)$$

   b. Actualizar la aproximación de la solución:

$$w_{i+1} = w_i + \frac{h}{6}\left(s_1 + 2s_2 + 2s_3 + s_4\right)$$

   c. Actualizar el valor de t_{i+1} utilizando $t_{i+1} = t_i + h$.
6. Repetir los pasos 5a, 5b y 5c para cada subintervalo, obteniendo así las aproximaciones sucesivas de la solución en los puntos $t_1, t_2, \ldots, t_N$.

7. La aproximación final de la solución de la ecuación diferencial ordinaria en el intervalo $[a, b]$ estará dada por los valores $w_0, w_1, \ldots, w_N$ obtenidos en los pasos anteriores.

El Método de Runge-Kutta de orden cuatro ofrece una mayor precisión en comparación con métodos como el Método de Euler y el Método del Trapecio Explícito. Sin embargo, también implica un mayor costo computacional debido al cálculo adicional de las pendientes s_2, s_3, y s_4. Su precisión y eficiencia lo convierten en una elección popular en el análisis numérico de ecuaciones diferenciales [26].

2.3. Redes neuronales convolucionales

Las redes neuronales convolucionales (CNN: *convolutional neural networks*, por sus siglas en inglés) son un tipo de redes neuronales artificiales, donde las neuronas artificiales o nodos corresponden a campos receptivos de una manera muy similar a las neuronas en la corteza visual primaria de un cerebro biológico [27]. Estas redes son muy efectivas para realizar tareas de visión computacional [28] y procesamiento de imágenes [29].

Las CNN se han utilizado con éxito en una amplia gama de áreas. En el campo de la medicina, han sido utilizadas para el diagnóstico de enfermedades a partir de imágenes médicas [30], como la detección de cáncer en mamografías [31] o la identificación de anomalías en resonancias magnéticas [32]. En la industria automotriz, las CNN son fundamentales en sistemas de conducción autónoma [33] para la detección de peatones [34], señales de tráfico y obstáculos en tiempo real [35]. Además, se aplican en la seguridad y vigilancia [36], el reconocimiento de escritura a mano [37], el procesamiento de imágenes satelitales [38], entre otras tareas.

Para poder entender como las CNN realizan este tipo de tareas, en los siguientes subepígrafes se profundizará en los principales conceptos y su funcionamiento.

2.3.1. Kernel

Un kernel, también conocido como filtro, es una matriz de coeficientes utilizada para realizar operaciones de convolución en una imagen [39]. En la figura 2.7, se observa un ejemplo de un kernel con coeficientes genéricos y una dimensión de 3×3 .

Figura 2.7: Kernel genérico de 3×3

Los kernels desempeñan un papel fundamental en el procesamiento de imágenes y en las CNN ya que permiten extraer características relevantes de una imagen. Existen diversos tipos de kernels, algunos de ellos incluyen los kernels de suavizado [40], que se emplean para reducir el ruido y suavizar la imagen como el kernel de promedio [41] (ver en la figura 2.8a); los kernels de detección de bordes [42], que resaltan los bordes y cambios bruscos en la imagen como el kernel de Sobel [43] (ver en la figura 2.8b) y el de Prewit [44] (ver en la figura 2.8c); y los kernels de convolución personalizados [45], que se diseñan según las necesidades particulares de la tarea y adaptan el proceso de convolución a características específicas de la imagen o de la red neuronal.

La elección del tipo de kernel depende del objetivo del procesamiento y de las características que se desean resaltar o detectar en la imagen.

1/9	1/9	1/9
1/9	1/9	1/9
1/9	1/9	1/9

(a) Kernel de promedio

-1	0	1
-2	0	2
-1	0	2

(b) Kernel de Sobel

1	1	1
0	0	0
-1	-1	-1

(c) Kernel de Prewitt

Figura 2.8: Ejemplos de diferentes tipos de kernel

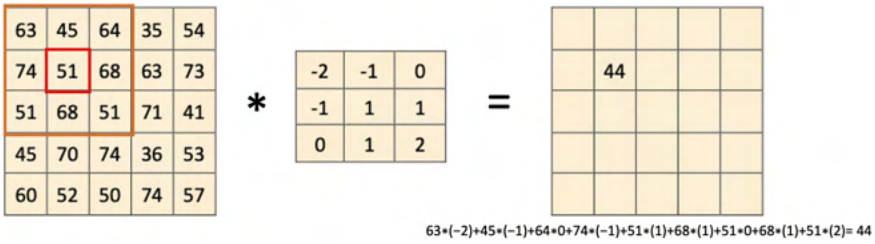
2.3.2. Convolución

La convolución es una operación matemática que combina dos funciones para producir una tercera, que representa cómo una de las funciones influye en la otra. Este proceso se aplica mediante la superposición de un kernel o filtro en una imagen de entrada, realizando una multiplicación punto a punto y sumando los resultados para obtener un valor representativo en la ubicación correspondiente de la salida [46]. Esto permite extraer características como bordes, texturas y formas en diferentes partes de la imagen. La operación de convolución se puede representar mediante la ecuación 2.4.

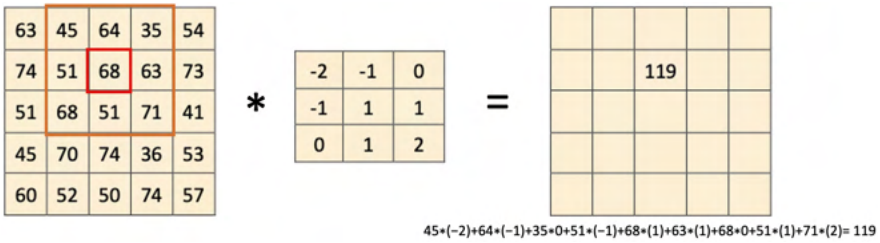
$$(f * g)(x, y) = \sum_i \sum_j f(i, j) \cdot g(x - i, y - j) \tag{2.4}$$

Donde f representa la imagen de entrada, g es el kernel y (x, y) son las coordenadas de la ubicación actual en la salida.

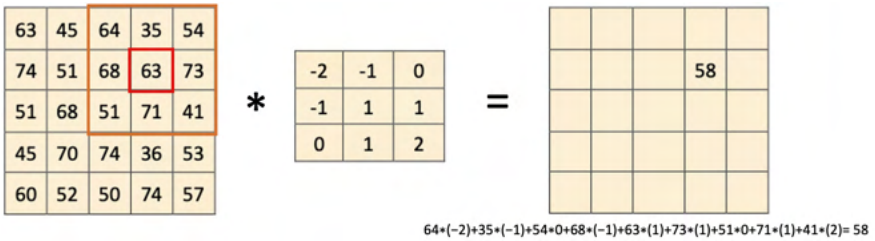
La representación gráfica de la convolución se puede observar en la figura 2.9.



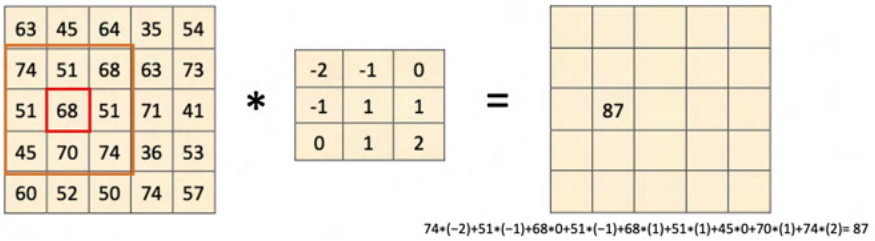
(a) Paso 1



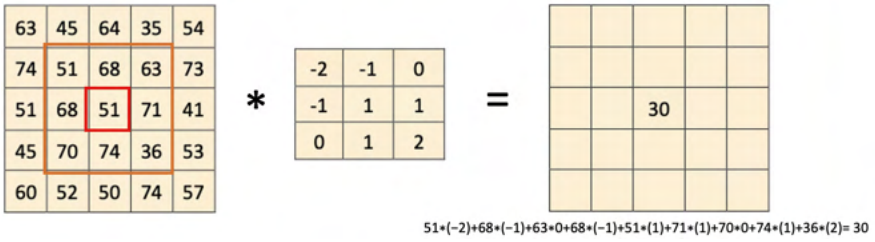
(b) Paso 2



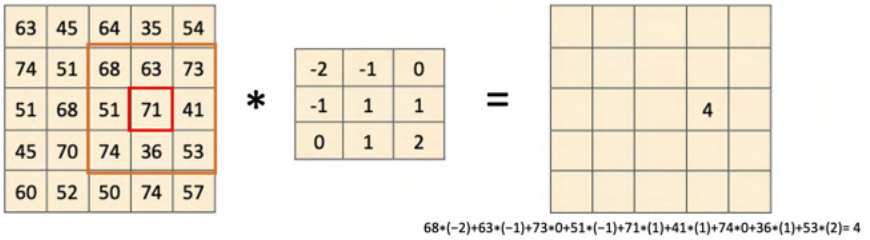
(c) Paso 3



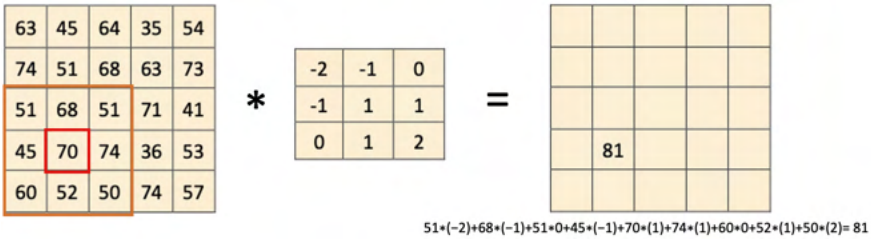
(d) Paso 4



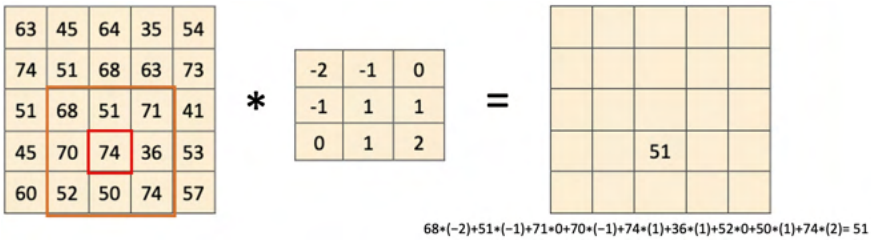
(e) Paso 5



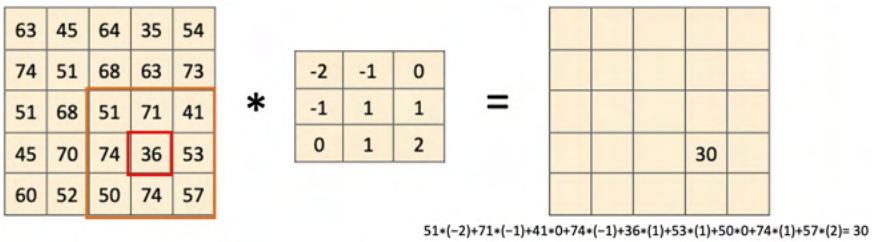
(f) Paso 6



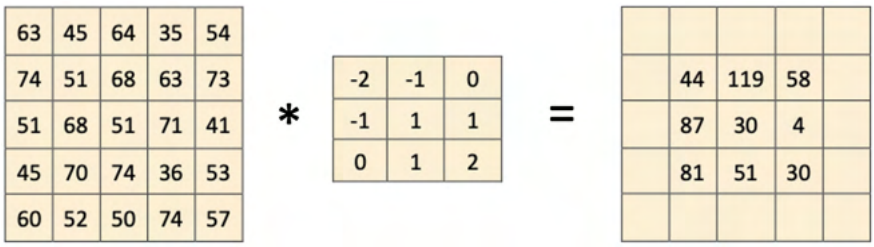
(g) Paso 7



(h) Paso 8



(i) Paso 9



(j) Resultado parcial

Figura 2.9: Proceso de convolución

Para poder aplicar también la convolución en los píxeles del borde de la imagen existen varias alternativas, algunas de las cuales son:

1. Completar con ceros los valores de alrededor.
2. Repetir los valores de la imagen original en el borde.
3. Completar con los valores de la parte simétrica opuesta de la imagen original.

En la figura 2.10 se muestra el resultado final de la convolución luego de repetir los valores de la imagen original en el borde .

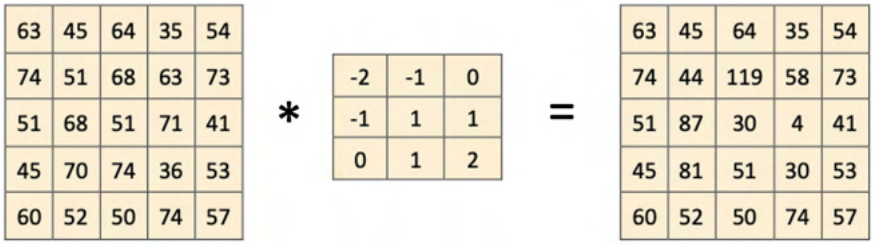


Figura 2.10: Resultado final

2.3.3. Función de activación

Una función de activación es una función matemática que se aplica a la salida de una neurona en una red neuronal artificial. Su propósito principal es introducir no linealidad en el modelo, lo que permite a la red neuronal aprender y representar relaciones y patrones más complejos en los datos [47]. A continuación, se describen algunas funciones de activación comunes utilizadas en las redes neuronales convolucionales.

2.3.3.1. ReLU (Rectified Linear Unit)

La función de activación ReLU es una de las más ampliamente utilizadas debido a su simplicidad y eficiencia computacional [48]. En la ecuación 2.5 y en la figura 2.11 se define la función de activación ReLU:

$$f(x) = \text{máx}(0, x) \tag{2.5}$$

La función ReLU mapea los valores negativos a cero y mantiene los valores positivos sin cambios. Esta función es conocida por ayudar a resolver el problema de desvanecimiento del gradiente y acelerar el proceso de entrenamiento de las redes neuronales.

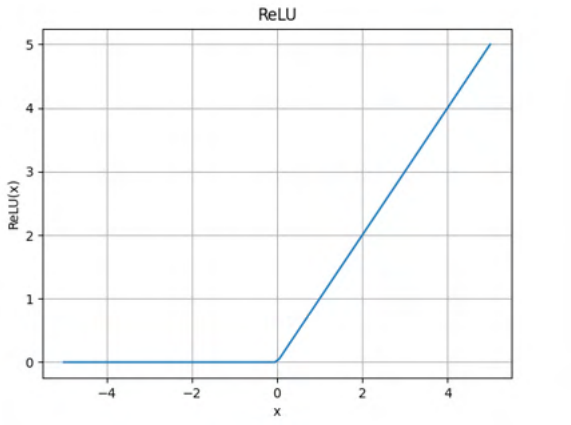


Figura 2.11: Función de activación ReLU

2.3.3.2. SiLU (Sigmoid-weighted Linear Unit)

La función de activación SiLU, también conocida como Swish, es una función no lineal propuesta recientemente que ha demostrado mejor rendimiento en comparación con otras funciones de activación tradicionales [49]. En la ecuación 2.6 y en la figura 2.12 se define la función de activación SiLU:

$$f(x) = \frac{x}{1 + e^{-x}} \tag{2.6}$$

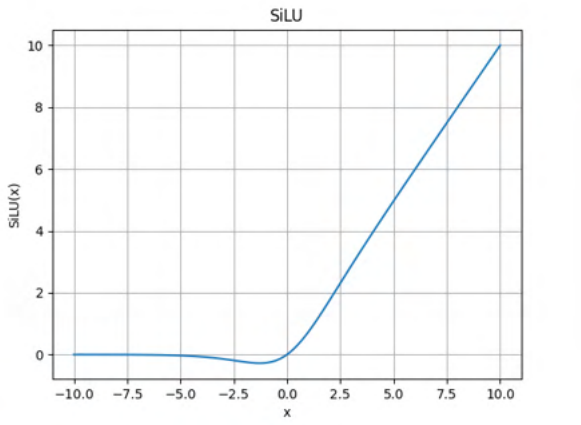


Figura 2.12: Función de activación SiLu

La función SiLU combina una función lineal ponderada con una función sigmoide. Dado un valor de entrada x, se aplica la función lineal x y luego se normaliza utilizando la función sigmoide $\sigma(x) = \frac{1}{1+e^{-x}}$. Esta función es continua y diferenciable en todos los puntos, lo que la hace adecuada para el entrenamiento de redes neuronales.

2.3.3.3. Softmax

La función de activación Softmax se utiliza comúnmente en la capa de salida de las redes neuronales clasificadoras para producir una distribución de probabilidades sobre las clases [50]. En la ecuación 2.7 se define la función de activación Softmax:

$$f(x_i) = \frac{e^{x_i}}{\sum_{j=1}^{C} e^{x_j}} \tag{2.7}$$

Donde x_i representa la entrada para la clase i, y C es el número total de clases.

La función Softmax calcula la exponencial de cada elemento de entrada, y luego normaliza los resultados dividiendo por la suma de todas las exponenciales. Esto produce una distribución de probabilidades sobre las clases, donde cada valor representa la probabilidad de pertenencia a una clase específica.

2.3.4. Función de costo

La función de costo o pérdida es una medida utilizada para evaluar qué tan bien se está desempeñando un modelo de redes neuronales en comparación con los valores reales del conjunto de datos [51]. La elección adecuada de la función de costo es crucial para el proceso de aprendizaje de la red. Una función de costo comúnmente utilizada en problemas de clasificación es la entropía cruzada (*cross-entropy*, por sus siglas en inglés), en la ecuación 2.8 se puede observar su definición.

$$L(y, \hat{y}) = -\sum_{i=1}^{N} y_i \log(\hat{y}_i) \tag{2.8}$$

Donde y es el vector de valores verdaderos y $\hat{y}$ es el vector de valores predichos por el modelo. La entropía cruzada penaliza las discrepancias entre los valores verdaderos y los valores predichos. Cuanto mayor sea la discrepancia, mayor será la pérdida resultante.

La entropía cruzada es una función de pérdida efectiva para problemas de clasificación y se utiliza ampliamente en el entrenamiento de redes neuronales. Su derivada en relación a los valores predichos $\hat{y}_i$ facilita el cálculo del gradiente durante el proceso de retropropagación, lo que permite actualizar los pesos de la red de manera eficiente.

2.3.5. Descenso de gradiente

El descenso de gradiente es un algoritmo de optimización utilizado en el entrenamiento de redes neuronales y en otros problemas de aprendizaje automático. Su objetivo es encontrar los valores óptimos de los parámetros de un modelo mediante la minimización de una función de pérdida [52].

El descenso de gradiente se basa en la idea de ajustar iterativamente los parámetros en la dirección opuesta al gradiente de la función de pérdida. Esto implica que, en cada iteración, se calcula el gradiente de la función de pérdida con respecto a los parámetros y se actualizan los parámetros en la dirección que conduce a una reducción de la pérdida.

El algoritmo del descenso de gradiente se puede describir mediante la siguiente actualización de los parámetros:

$$\theta = \theta - \alpha \nabla J(\theta) \tag{2.9}$$

Donde θ representa los parámetros del modelo, α es la tasa de aprendizaje (un hiperparámetro que controla la velocidad de convergencia) y $\nabla J(\theta)$ es el gradiente de la función de pérdida J con respecto a los parámetros θ.

2.3.6. Descenso de gradiente estocástico

El descenso de gradiente estocástico (SGD, por sus siglas en inglés) es una variante del algoritmo de descenso de gradiente que se utiliza ampliamente en el entrenamiento de redes neuronales [53]. El SGD realiza actualizaciones de parámetros para cada muestra de entrenamiento de forma individual o en pequeños grupos conocidos como mini lotes.

El algoritmo del SGD se puede describir mediante la siguiente actualización de los parámetros para una sola muestra de entrenamiento:

$$\theta = \theta - \alpha \nabla J(\theta; x^{(i)}, y^{(i)}) \tag{2.10}$$

Donde θ representa los parámetros del modelo, α es la tasa de aprendizaje y $\nabla J(\theta; x^{(i)}, y^{(i)})$ es el gradiente de la función de pérdida J con respecto a los parámetros θ calculado para la muestra de entrenamiento $x^{(i)}$ y su correspondiente etiqueta $y^{(i)}$.

2.3.7. Entrenamiento

Una vez que se ha configurado el modelo de la red neuronal convolucional, seleccionado la función de costo apropiada tras experimentar con varias opciones para abordar el problema en cuestión, así como el método de optimización para ajustar los pesos, se inicia el proceso de entrenamiento del modelo. En esta etapa, se definen una serie de componentes esenciales para llevar a cabo el entrenamiento de manera efectiva:

- **Subconjuntos de la base de datos:** Subconjunto de entrenamiento, validación y prueba.
- **Batch size o tamaño de lote:** Número de muestras que se utilizan en cada iteración durante el proceso de entrenamiento.
- **Época:** Número de veces que se procesan todos los lotes de entrenamiento completos durante el entrenamiento.
- **Tasa de aprendizaje:** Hiperparámetro que determina la cantidad de actualización aplicada a los pesos del modelo en cada iteración del algoritmo de optimización.

2.3.8. Aumentación de datos

La técnica de aumento de datos surge con el objetivo de aumentar la cantidad y diversidad del conjunto de datos de entrenamiento [54]. Esto se realiza mediante la aplicación de transformaciones a las imágenes existentes, lo que permite generar nuevas muestras que conservan las características relevantes de los datos originales.

Algunos ejemplos de estas transformaciones son:

- **Aumentación geométrica:** Se aplican transformaciones como rotaciones, traslaciones y cambios de escala a las imágenes. Estas transformaciones permiten introducir variaciones en la posición y orientación de los objetos, lo que ayuda a que el modelo sea más robusto frente a diferentes situaciones [55].
- **Aumentación de color:** Se aplican cambios en el brillo, contraste, saturación y tonalidad de las imágenes. Estas transformaciones permiten introducir variaciones en la apariencia de los objetos, lo que ayuda a que el modelo sea más invariante a cambios en la iluminación y las condiciones de color [56].
- **Aumentación de recorte:** Se realizan recortes aleatorios en las imágenes, lo que permite enfocar la atención del modelo en regiones específicas de interés. Esto ayuda a que el modelo sea más preciso al detectar y clasificar objetos en diferentes tamaños y contextos [57].

2.4. Red neuronal convolucional *You Only Look Once (YOLO)*

You Only Look Once (Yolo) es una red neuronal convolucional enfocada en la detección y clasificación de objetos, propuesta por Joseph Redmon en el año 2015 [58]. Para realizar las tareas de detección y clasificación muchos sistemas actuales de detección funcionan deslizando varias ventanas de diferentes dimensiones por toda la imagen hasta que encuentra un objeto y luego lo inserta a un clasificador, este proceso puede ser muy engorroso y con una alta complejidad computacional [59]. Yolo propone un nuevo enfoque para realizar dichas tareas ya que las maneja como un simple problema de regresión, es decir, directamente de los píxeles de la imagen y utilizando una serie de convoluciones predice las coordenadas de los cuadros delimitadores y las probabilidades de las clases, por lo que no tiene un alto costo computacional y se logra optimizar el rendimiento de la red para la detección y clasificación de objetos.

YOLO divide la imagen de entrada en una cuadrícula de tamaño $S \times S$. Cada celda de esta cuadrícula cuenta con cinco variables fundamentales: x, y, w, h y confianza. Las variables x y y representan las coordenadas del centro del objeto detectado en esa celda, w es el ancho y h es el largo de la caja delimitadora encargada de encerrar el objeto encontrado, y la variable de confianza muestra la probabilidad de que en esa celda exista el centro de un objeto. Cada celda contiene también la probabilidad de

cada clase a la que puede pertenecer el objeto detectado. Los cálculos de estas variables sólo se realizan en las celdas que contengan una alta confianza. YOLO puede detectar múltiples cuadros delimitadores por celda de cuadrícula, en caso que el centro de dos o más objetos se encuentren dentro de la misma celda.

2.4.1. YOLO v3

YOLO v3 es una versión mejorada del modelo original YOLO, presentada en el año 2018 por Joseph Redmon y Ali Farhadi [60]. Su objetivo principal es abordar las limitaciones de las versiones anteriores y mejorar la precisión de la detección de objetos en imágenes.

Para calcular las coordenadas absolutas del cuadro delimitador en la imagen completa, se utilizan las siguientes fórmulas:

1. Coordenada x del centro del cuadro delimitador:

$$x = \sigma(t_x) + c_x \tag{2.11}$$

   Donde:

   - σ es la función de activación sigmoide, que se utiliza para asegurar que t_x esté en el rango de 0 a 1.
   - t_x es el valor predicho de la coordenada x del cuadro delimitador (relativo al tamaño de la celda).
   - c_x es el índice de la celda en el eje horizontal de la cuadrícula.

2. Coordenada y del centro del cuadro delimitador:

$$y = \sigma(t_y) + c_y \tag{2.12}$$

   Donde:

   - σ es la función de activación sigmoide.
   - t_y es el valor predicho de la coordenada y del cuadro delimitador (relativo al tamaño de la celda).
   - c_y es el índice de la celda en el eje vertical de la cuadrícula.

3. Ancho del cuadro delimitador:

$$w = p_w \cdot e^{t_w} \tag{2.13}$$

   Donde:

   - p_w es el ancho previamente definido del cuadro delimitador de referencia (por ejemplo, el ancho de la celda).

- t_w es el valor predicho del ancho del cuadro delimitador (relativo al tamaño de la celda).

4. Altura del cuadro delimitador:

$$h = p_h \cdot e^{t_h} \tag{2.14}$$

Donde:

- p_h es la altura previamente definida del cuadro delimitador de referencia (por ejemplo, la altura de la celda).
- t_h es el valor predicho de la altura del cuadro delimitador (relativo al tamaño de la celda).

Es importante tener en cuenta que estos cálculos se realizan para cada celda en la cuadrícula, y los valores resultantes representan las coordenadas absolutas del cuadro delimitador en la imagen completa. Estas coordenadas permiten ubicar con precisión los objetos detectados dentro de la imagen. Para una mejor visualización ver figura 2.13.

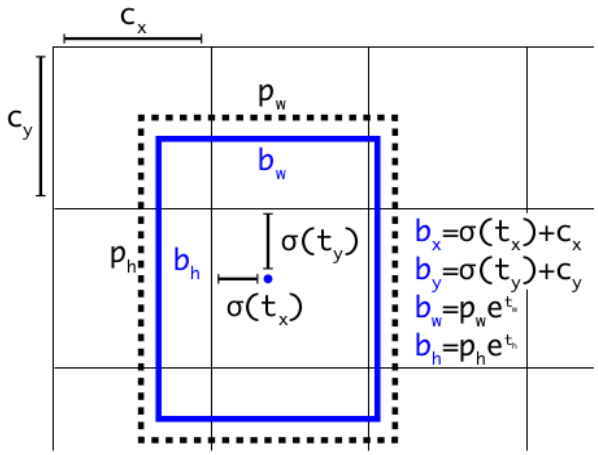


Figura 2.13: Cálculo de las coordenadas absolutas de los cuadros delimitadores

2.4.2. YOLO v5

YOLO v5 es una versión más reciente y optimizada de la serie YOLO, presentada por Glenn Jocher y desarrollada por la empresa Ultralytics en mayo de 2020 [61]. En YOLO v5, la fórmula para predecir las coordenadas de la caja se actualizó para reducir la sensibilidad de la cuadrícula y evitar que el modelo prediga dimensiones de caja ilimitadas.

Las fórmulas para calcular el cuadro delimitador previsto son las siguientes:

1. Coordenada x del centro del cuadro delimitador:

$$x = (2 \cdot \sigma(t_x) - 0.5) + c_x \tag{2.15}$$

Donde:

- σ es la función de activación sigmoide, que se utiliza para asegurar que t_x esté en el rango de 0 a 1.
- t_x es el valor predicho de la coordenada x del cuadro delimitador (relativo al tamaño de la celda).
- c_x es el índice de la celda en el eje horizontal de la cuadrícula.

2. Coordenada y del centro del cuadro delimitador:

$$y = (2 \cdot \sigma(t_y) - 0.5) + c_y \tag{2.16}$$

Donde:

- σ es la función de activación sigmoide.
- t_y es el valor predicho de la coordenada y del cuadro delimitador (relativo al tamaño de la celda).
- c_y es el índice de la celda en el eje vertical de la cuadrícula.

3. Ancho del cuadro delimitador:

$$w = p_w \cdot (2 \cdot \sigma(t_w))^2 \tag{2.17}$$

Donde:

- p_w es el ancho previamente definido del cuadro delimitador de referencia (por ejemplo, el ancho de la celda).
- σ es la función de activación sigmoide.
- t_w es el valor predicho del ancho del cuadro delimitador (relativo al tamaño de la celda).

4. Altura del cuadro delimitador:

$$h = p_h \cdot (2 \cdot \sigma(t_h))^2 \tag{2.18}$$

Donde:

- p_h es la altura previamente definida del cuadro delimitador de referencia (por ejemplo, la altura de la celda).
- σ es la función de activación sigmoide.

- t_h es el valor predicho de la altura del cuadro delimitador (relativo al tamaño de la celda).

YOLOv5 en su arquitectura cuenta con 4 partes fundamentales [61]:

- Entrada de imágenes (*Input*): Imagen
- Columna vertebral (*Backbone*): New CSP-Darknet53
- Cuello (*Neck*): SPPF
- Cabeza (*Head*): YOLOv3 Head

En la figura 2.14 se puede observar a detalle la arquitectura de la red neuronal convolucional YOLOv5.

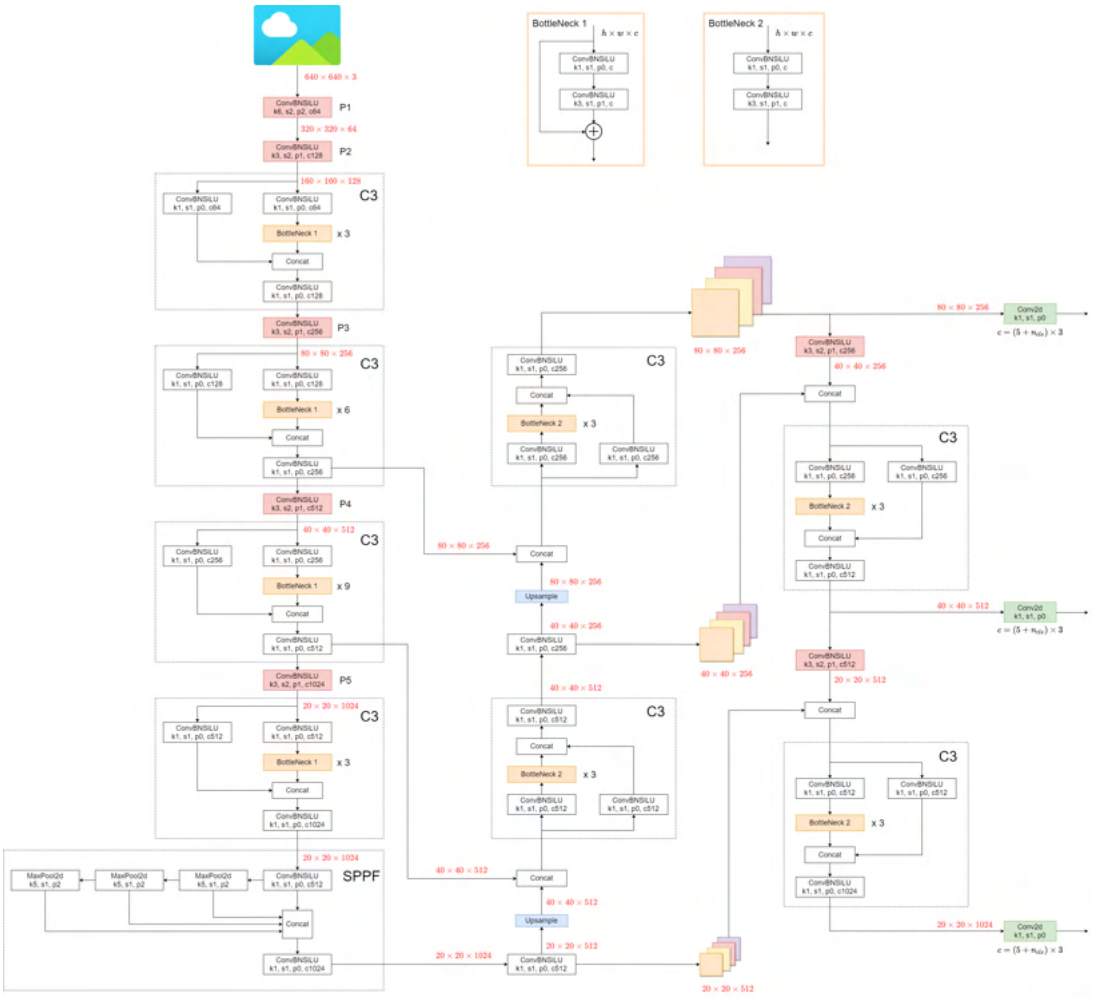


Figura 2.14: Arquitectura de YOLOv5 [61]

2.4.3. Intersección sobre la Unión

La métrica Intersección sobre la Unión (IoU: *Intersection over Union*, por sus siglas en inglés), también conocida como Jaccard Index, es una medida comúnmente utilizada para evaluar la superposición o coincidencia entre dos cajas delimitadoras (*bounding boxes*, por sus siglas en inglés) en problemas de detección de objetos [62]. El IOU se calcula dividiendo el área de intersección entre las dos cajas por su área de unión, como se muestra en la ecuación 2.19.

$$\text{IoU} = \frac{\text{Área de intersección}}{\text{Área de unión}} \tag{2.19}$$

En la figura 2.15 se puede observar diferentes ejemplos de superposiciones de dos cajas delimitadoras y su resultado al calcular su IOU.

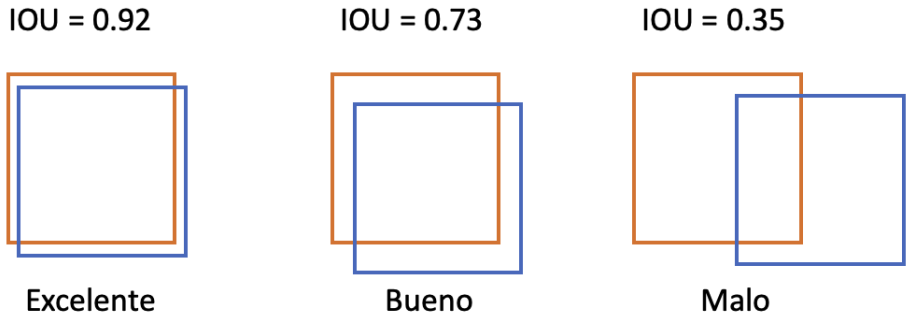


Figura 2.15: Ejemplos de superposiciones de dos cajas delimitadoras

El valor del IOU oscila entre 0 y 1, donde 0 indica que no hay superposición entre las cajas delimitadoras y 1 indica una superposición completa o coincidencia perfecta. Un IOU mayor indica una mayor similitud entre las cajas y una mejor alineación de los objetos detectados. Se utiliza como criterio de comparación para determinar si una detección es un verdadero positivo o un falso positivo, según si el IOU supera un umbral predefinido.

2.4.4. No Maximum Suppression (NMS)

No Maximum Suppression (NMS) es un algoritmo utilizado en problemas de detección de objetos para eliminar detecciones redundantes y seleccionar las más relevantes [63]. Su objetivo principal es evitar la duplicación de detecciones y mejorar la precisión de los resultados finales.

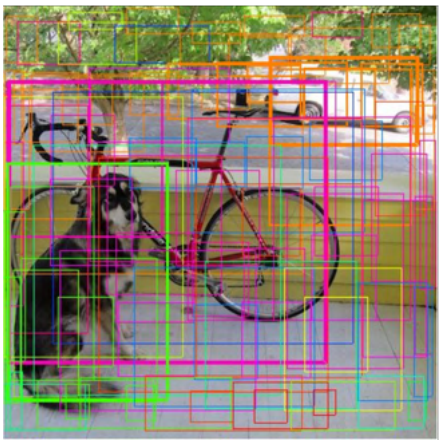
El Algoritmo 2.1 muestra el procedimiento de NMS utilizado para filtrar las detecciones superpuestas [64].

Una vez aplicado el algoritmo de NMS sólo permanecen los *bounding boxes* supervivientes. Ver figura 2.16.

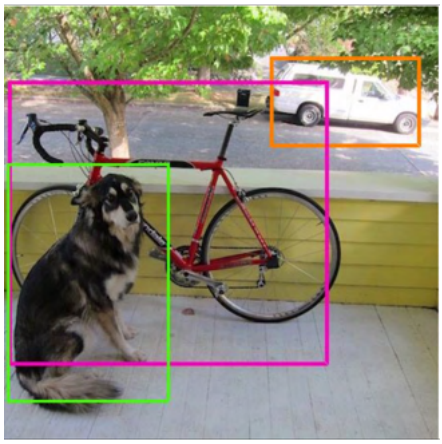
Algoritmo 2.1: Algoritmo de *No Maximum Suppression*

Entrada: Imágenes con sus respectivas detecciones
Salida: Detecciones filtradas

1. Ordenar todas las detecciones de mayor a menor, basado en el valor de probabilidad de clase C_i asignado;
2. Para cada detección d, calcular su Intersección sobre la Unión (IoU) con las detecciones que tienen un valor de probabilidad menor;
3. Eliminar aquellos bounding boxes que superen un umbral t de IoU;
4. Repetir el paso anterior con la siguiente detección si existen detecciones restantes;
5. Continuar con la siguiente imagen;



(a) Imagen antes de aplicar NMS



(b) Imagen luego de aplicar NMS

Figura 2.16: Aplicación del algoritmo *No Maximum Suppression*

2.5. Métricas de desempeño

En esta sección se presentan las principales métricas de desempeño utilizadas para evaluar la calidad y precisión de los modelos de detección y clasificación de objetos.

2.5.1. Matriz de confusión

La matriz de confusión es una herramienta fundamental en la evaluación del desempeño de un modelo de detección y clasificación. Permite visualizar y analizar la cantidad de verdaderos positivos (TP), falsos positivos (FP), verdaderos negativos (TN) y falsos negativos (FN) obtenidos por el modelo. Esta matriz es especialmente útil para comprender el grado de acierto y error en las predicciones del modelo [65].

En la figura 2.17 se muestra un ejemplo visual de una matriz de confusión, donde:

- TN (True Negative): Número de casos negativos correctamente clasificados como negativos.
- FP (False Positive): Número de casos negativos incorrectamente clasificados como positivos.
- FN (False Negative): Número de casos positivos incorrectamente clasificados como negativos.
- TP (True Positive): Número de casos positivos correctamente clasificados como positivos.

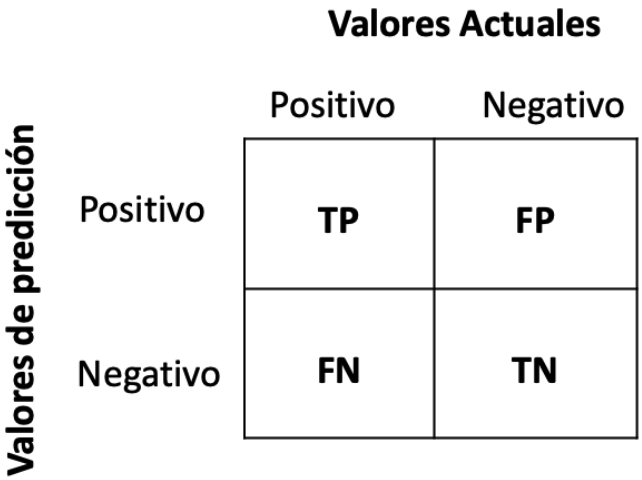


Figura 2.17: Ejemplo visual de una matriz de confusión.

2.5.2. Precisión

La precisión es una métrica de desempeño utilizada en problemas de clasificación que mide la proporción de casos positivos correctamente identificados por el modelo entre todos los casos clasificados como positivos [66]. La ecuación 2.20 muestra como calcular la precisión.

$$\text{Precisión} = \frac{TP}{TP + FP} \tag{2.20}$$

Donde:

- TP (True Positive): Número de casos positivos correctamente clasificados como positivos.
- FP (False Positive): Número de casos negativos incorrectamente clasificados como positivos.

2.5.3. Exhaustividad

La exhaustividad (*Recall*, por sus siglas en inglés), también conocido como sensibilidad o tasa de verdaderos positivos, es una métrica utilizada para evaluar el rendimiento de un modelo de clasificación o detección. Representa la proporción de casos positivos que han sido correctamente identificados por el modelo en relación con todos los casos positivos presentes en el conjunto de datos [67]. La ecuación 2.21 muestra como calcular el recall.

$$\text{Recall} = \frac{TP}{TP + FN} \tag{2.21}$$

Donde:

- TP (True Positive): Número de casos positivos correctamente clasificados como positivos.
- FN (False Negative): Número de casos positivos incorrectamente clasificados como negativos.

2.5.4. F1-Score

El F1-Score es una métrica utilizada para evaluar el rendimiento de un modelo de clasificación o detección, que combina la precisión y el recall en una única medida. Representa la media armónica entre la precisión y el recall, lo que proporciona una visión equilibrada del desempeño del modelo [68]. La ecuación 2.22 muestra como calcular el F1-Score.

$$\text{F1-Score} = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \tag{2.22}$$

Donde:

- Precision: Es la proporción de casos positivos correctamente identificados en relación con todos los casos clasificados como positivos.
- Recall: Es la proporción de casos positivos correctamente identificados en relación con todos los casos positivos presentes en el conjunto de datos.

2.5.5. Mean Average Precision (mAP)

El mAP es una métrica utilizada para evaluar el rendimiento de los modelos de detección de objetos. Mide la precisión promedio de detección de objetos en diferentes niveles de umbral de confianza [69]. El mAP se calcula utilizando la curva de precisión-recall y la ecuación 2.23 muestra como calcularlo.

$$\text{mAP} = \frac{1}{N} \sum_{i=1}^{N} \text{AP}_i \tag{2.23}$$

Donde:

- n es el número de clases o categorías.
- AP_i es la precisión promedio (*Average Precision*) para la clase i y podemos ver como calcular el AP en la ecuación 2.24 [70].

$$\text{AP} = \sum_{r=1}^{R} (P(r) \cdot \Delta r) \tag{2.24}$$

Donde:

- R representa los valores de recall.
- $p(r)$ representa la precisión correspondiente a cada valor de recall.
- Δr representa el cambio en el valor de recall.

El mAP se puede calcular utilizando diferentes valores de umbral de confianza. Dos métricas comunes son mAP@0.5 y mAP@0.5-0.95. La métrica mAP@0.5 se refiere al promedio de las precisiones para un umbral de confianza de 0.5 y mAP@0.5-0.95 se refiere al promedio de las precisiones para un rango de umbrales de confianza desde 0.5 hasta 0.95 con incrementos de 0.05.

Capítulo 3

Metodología

En esta sección se presenta el proceso a llevar a cabo para el desarrollo de este trabajo. Se detalla el procedimiento para la preparación de la base de datos compuesta por imágenes procesadas por el ensayo Cometa. Luego, se definen los módulos que componen la arquitectura del modelo propuesto para la detección y clasificación de células en imágenes procesadas por el ensayo Cometa. Finalmente se describe como se realizó el entrenamiento y la evaluación de dicho modelo.

3.1. Funcionamiento general del sistema

El sistema propuesto para la detección y clasificación de células en imágenes procesadas por el ensayo Cometa se compone por 3 módulos, cada uno es encargado de realizar una tarea en espécifico. La entrada del sistema puede ser una imagen en escala de grises o una imagen RGB, y la salida es la misma imagen de entrada, pero con contornos añadidos que delimitan los cometas y los clasifica según la clase a la que pertenecen.

En la figura 3.1, se puede observar un diagrama representativo del sistema desarrollado.

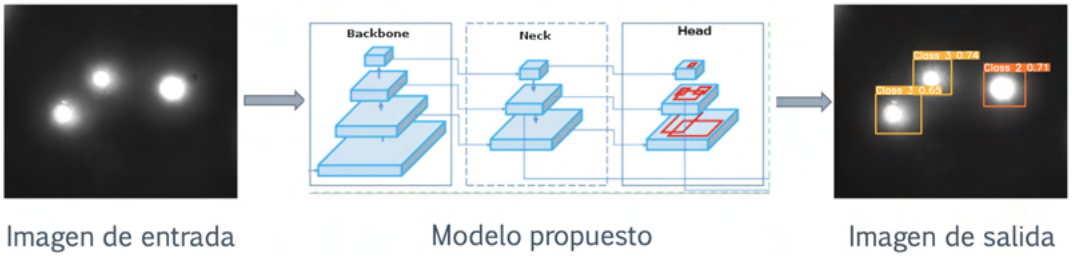


Figura 3.1: Esquema del sistema de detección

A continuación se explica el funcionamiento de cada módulo del sistema:

- ***Backbone***: Es la parte inicial de la red convolucional y se encarga de extraer características de la imagen de entrada. El backbone suele estar compuesto por

varias capas convolucionales, como convoluciones 2D, convoluciones de separable espacial, entre otras.

- ***Neck***: Su función principal es fusionar y combinar las características extraídas por el módulo *backbone*. Suele estar formado por capas convolucionales adicionales que ayudan a reducir la dimensionalidad y extraer características más abstractas y de alto nivel.
- ***Head***: Es el último módulo de la red y se encarga de realizar las predicciones finales, como la detección y clasificación de objeto.

3.2. Base de datos

Para realizar este estudio se dispone de una base de datos compuesta por un conjunto de 236 imágenes que han sido sometidas al análisis del ensayo Cometa. Estas imágenes capturan células en un formato de 728×960 píxeles en escala de grises, proporcionando una representación detallada y precisa de las estructuras celulares.

La base de datos fue armada desde Uruguay, etiquetando uno a uno cada cometa. Las imágenes del Grupo de Biofisicoquímica (dirigido por el Prof RD Peluffo) del Departamento de Ciencias Biológicas (DCB) en el Centro Universitario Regional Litoral Nortev(CENUR LN) de la Universidad de la República, Salto, fueron etiquetadas por la Lic Natalia Ibargoyen en tanto imágenes del Depto de Genética del Instituto de Investigaciones Biológicas Clemente Estable perteneciente al Ministerio de Educación y Cultura (IIBCE, MEC) fueron etiquetadas por la Lic Valentina Perini. El conocimiento y experiencia de estos grupos y la supervisión de Laura Lafon-Hughes garantizan la calidad y relevancia de las imágenes, estableciendo una sólida base para el desarrollo y evaluación de nuestro modelo propuesto.

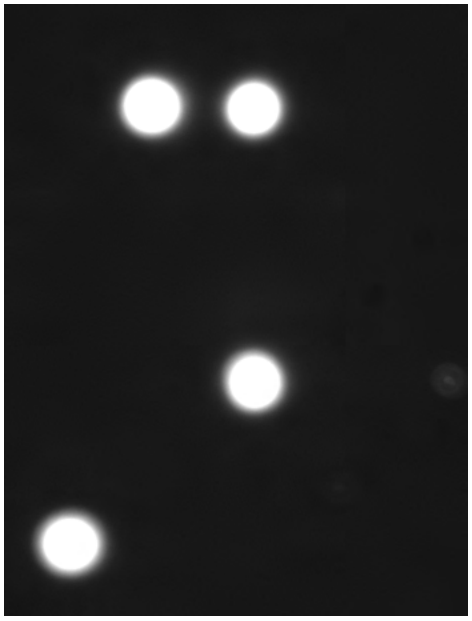
La base de datos se divide en tres conjuntos fundamentales: entrenamiento, validación y prueba. En este contexo, se emplearán respectivamente 189, 23 y 24 imágenes, representando proporciones del 80 %, 10 % y 10 % de la base de datos original. Esta partición estratégica no solo respalda la efectividad del proceso de entrenamiento y evaluación del modelo propuesto, sino que también permite que el modelo pueda generalizar y hacer predicciones confiables en escenarios del mundo real.

En la tabla 3.1 se muestra la distribución de cometas por clases en las imágenes existentes en la base de datos.

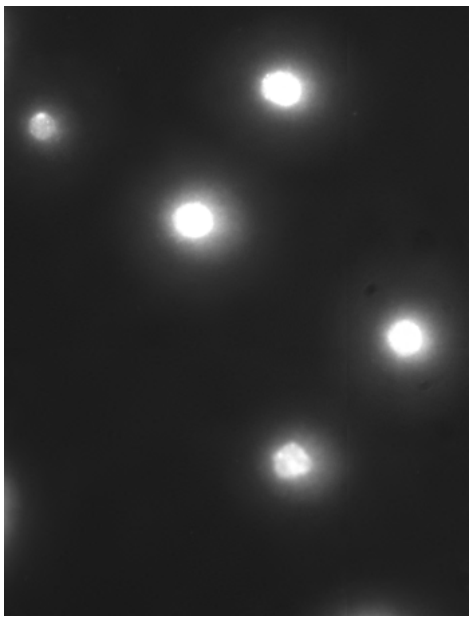
Clases de cometas	Entrenamiento	Validación	Prueba	Total
Clase 1	65	11	6	**82**
Clase 2	188	20	22	**230**
Clase 3	177	36	23	**236**
Clase 4	64	11	4	**79**
Clase 5	11	2	1	**14**
Total	**505**	**80**	**56**	**641**

Tabla 3.1: Distribución del número de cometas por clase en las imágenes de la base de datos

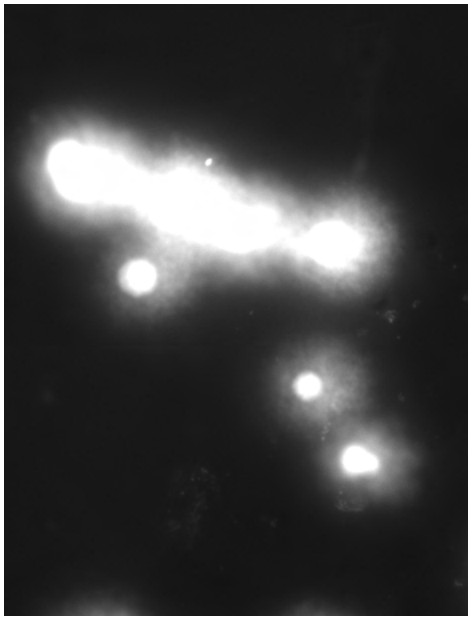
A continuación se muestra una selección ilustrativa de imágenes que capturan diversos aspectos celulares y sus características únicas. Estos ejemplos ofrecen una visión tangible de cómo se visualiza el ADN y las células en diferentes condiciones y contextos.



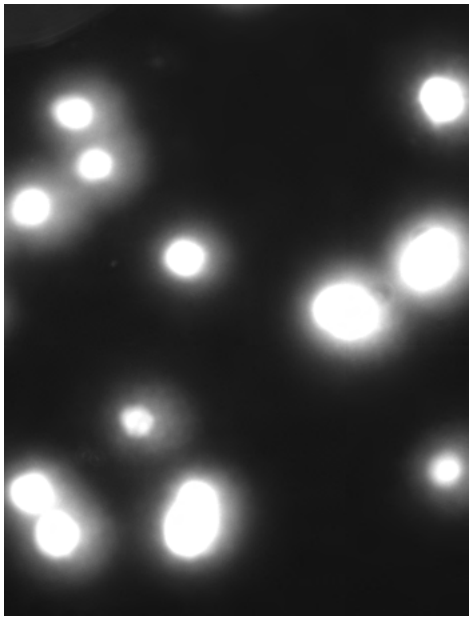
(a) Ejemplo 1



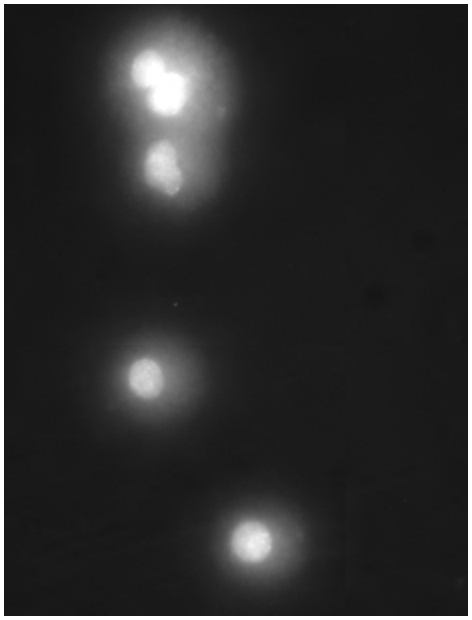
(b) Ejemplo 2



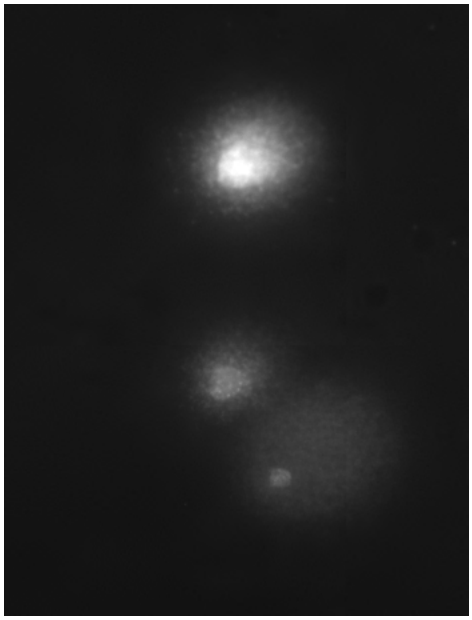
(a) Ejemplo 3



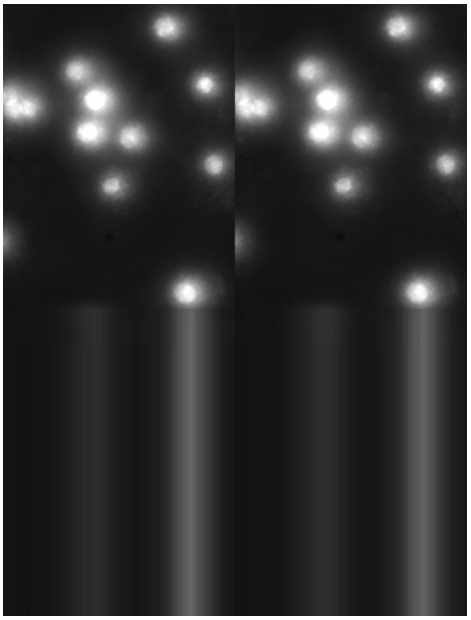
(b) Ejemplo 4



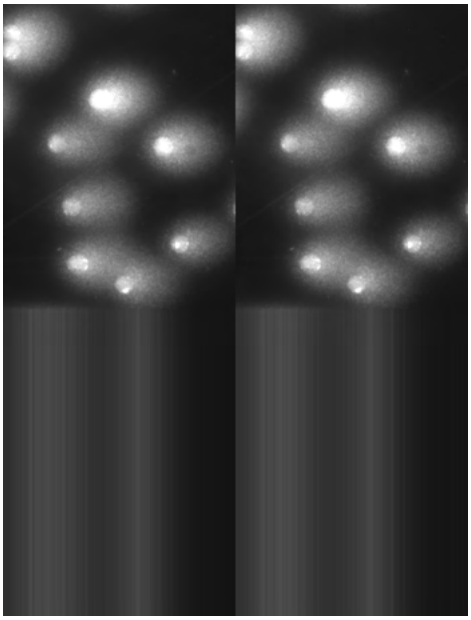
(a) Ejemplo 5



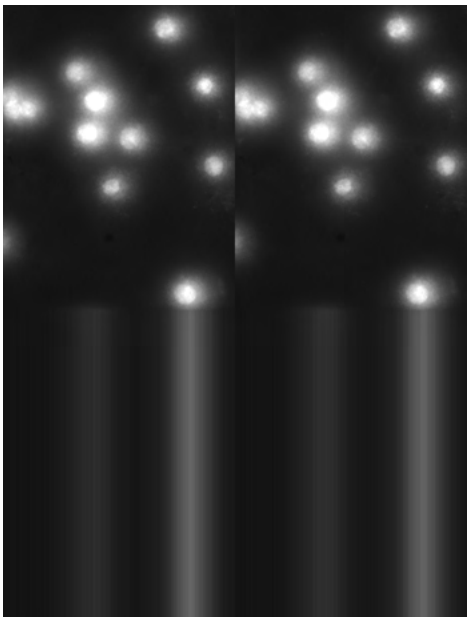
(b) Ejemplo 6



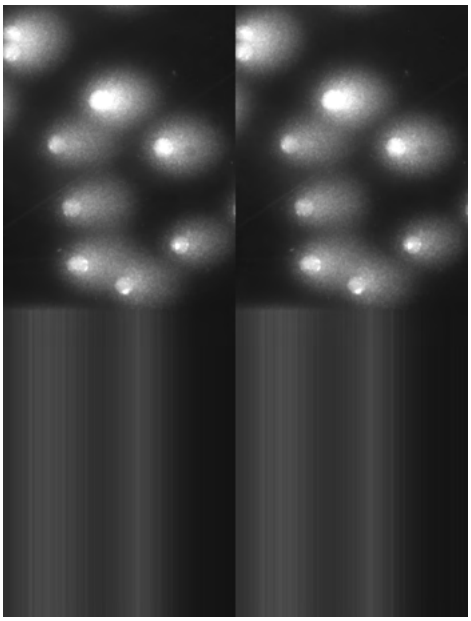
(a) Ejemplo 7



(b) Ejemplo 8



(a) Ejemplo 9



(b) Ejemplo 10

3.2.1. Ajuste y etiquetado manual de la base de datos

Para poder entrenar la CNN es necesario etiquetar cada cometa encontrado en las imágenes de entrenamiento, según la clase a la que pertenece como se puede observar en la figura 3.7. Este proceso se realizó utilizando la herramienta informática LabelImg [71].

El ajuste y etiquetado manual de la base de datos desempeña un papel fundamental en la preparación de los datos para el entrenamiento de la Red Neuronal Convolucional. Esta etapa implica el proceso de anotar meticulosamente cada instancia de cometa identificado en las imágenes de entrenamiento, asignando a cada una la etiqueta de su clase correspondiente, como se puede observar en la figura 3.7.

Para llevar a cabo este procedimiento de manera eficiente y precisa, hemos empleado la herramienta informática especializada conocida como LabelImg [71]. Esta herramienta proporciona una interfaz amigable que permite a los etiquetadores Lic. Natalia Ibargoyen y Lic. Valentina Perini trazar cuidadosamente los límites de cada cometa presente en las imágenes. Además de su facilidad de uso, LabelImg ofrece la capacidad de asignar las clases adecuadas a cada objeto anotado, permitiendo la creación de un conjunto de datos etiquetado y bien estructurado.

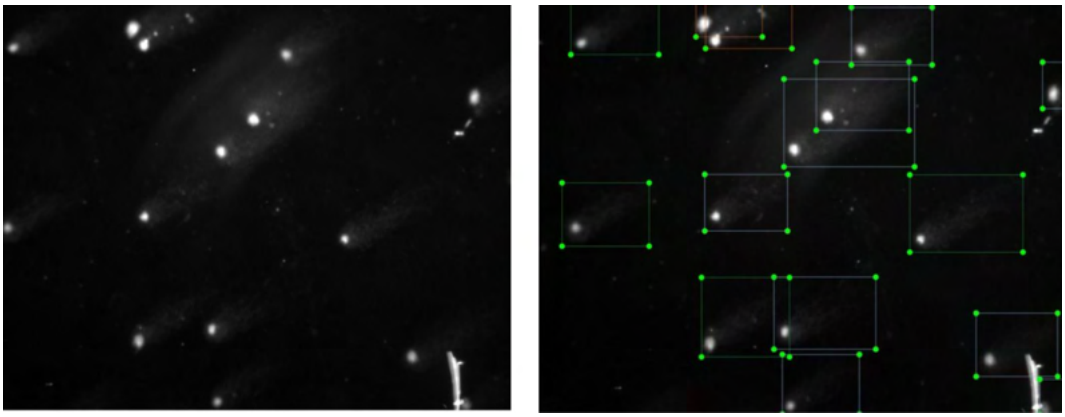


Figura 3.7: Imagen original (izquierda) e imagen procesada por herramienta LabelImg (derecha)

Luego de realizar la marcación de los cometas, esta herramienta lanza como resultado un fichero con extensión .txt (ver figura 3.8) para cada imagen con la siguiente estructura:

- Columna 1 : Clase a la que pertenece el objeto.
- Columna 2 : Centro del objeto con respecto al eje de coordenadas horizontal.
- Columna 3 : Centro del objeto con respecto al eje de coordenadas vertical.
- Columna 4: Distancia desde el centro hasta el borde horizontal.
- Columna 5: Distancia desde el centro hasta el borde vertical.

archivo.txt

```
3 0.427198 0.282292 0.142052 0.096983
3 0.700549 0.279167 0.198813 0.130500
3 0.137363 0.707292 0.174201 0.111042
3 0.160714 0.807292 0.173429 0.102217
3 0.692308 0.596875 0.258354 0.148779
4 0.781593 0.738542 0.335162 0.164421
4 0.501374 0.905208 0.195346 0.109040
4 0.105769 0.515625 0.168063 0.124006
```

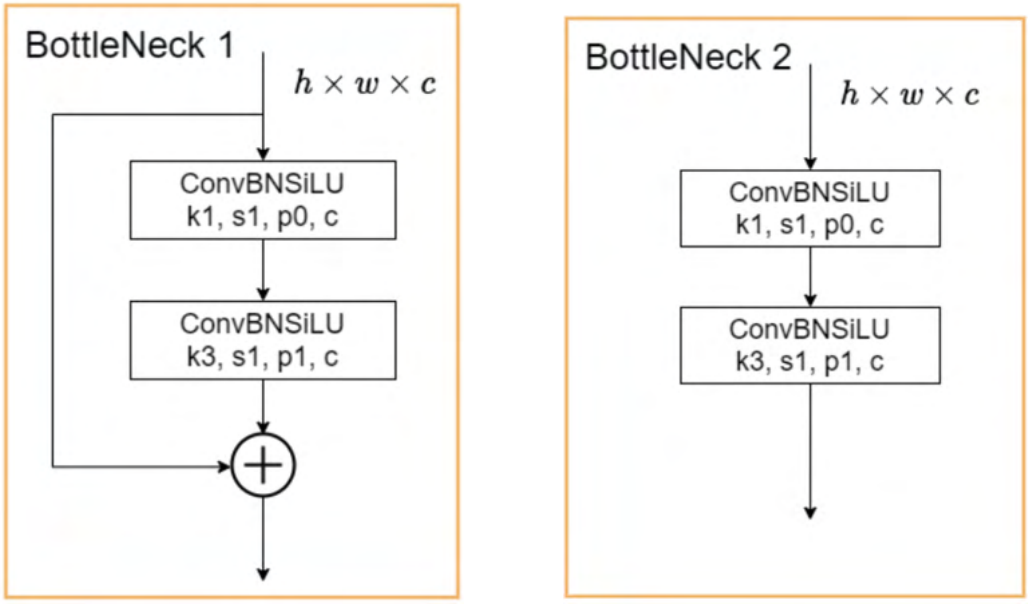
Figura 3.8: Ejemplo de archivo.txt generado por la herramienta LabelImg

3.3. Arquitectura del modelo propuesto

La red neuronal propuesta para el sistema de detección y clasificación de células en imágenes procesadas por el ensayo Cometa se basa en la arquitectura de YOLOv5 (ver figura 2.14), una red convolucional reconocida por su capacidad para realizar tareas de detección de objetos de manera precisa y eficiente.

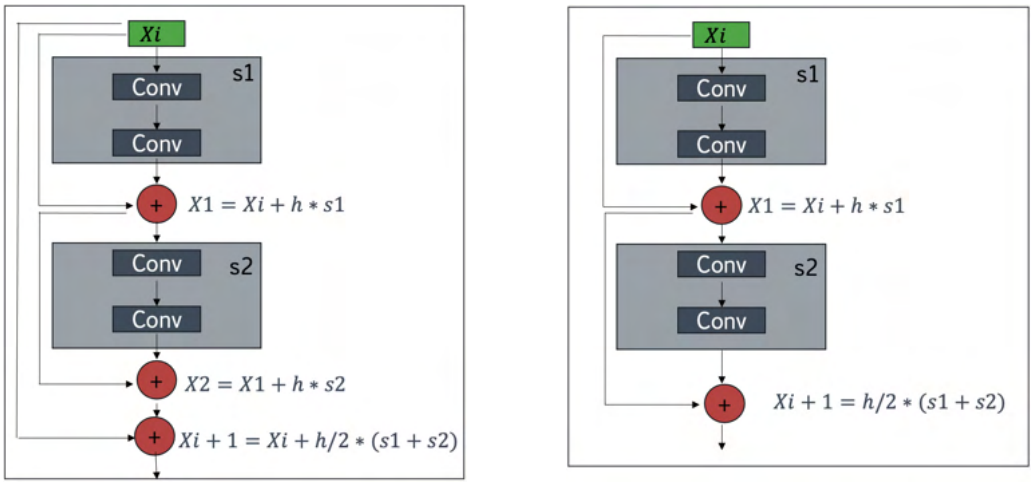
El cambio principal que se introduce en el sistema propuesto es en los módulos denominados *BottleNeck* 1 (ver figura 3.9a) y *BottleNeck* 2 (ver figura 3.9b), encontrados en el *Backbone* y *Head* respectivamente. En particular, se exploran dos enfoques alternativos: el Método de Heun y el Método de Runge-Kutta de cuarto orden (RK4). Estos métodos, ampliamente reconocidos en el campo de las ecuaciones diferenciales ordinarias, se adaptan y aplican específicamente al proceso de detección de células dañadas en el contexto de la red neuronal convolucional YOLOv5. Para referirnos a estos modelos propuestos, los denominaremos YOLOv5_Heun y YOLOv5_RK4 .

Originalmente esta red realiza el procesamiento de los datos en dirección hacia adelante en los módulos de *BottleNeck* 1 y 2 basándose en el Método de Euler utilizando la ecuación 2.2, el cambio realizado fue sustituir esta fórmula por las ecuaciones correspondientes a los métodos de Heun 2.3 y Runge-Kutta de orden cuatro 5 como se puede observar en las figuras 3.10 y 3.11.



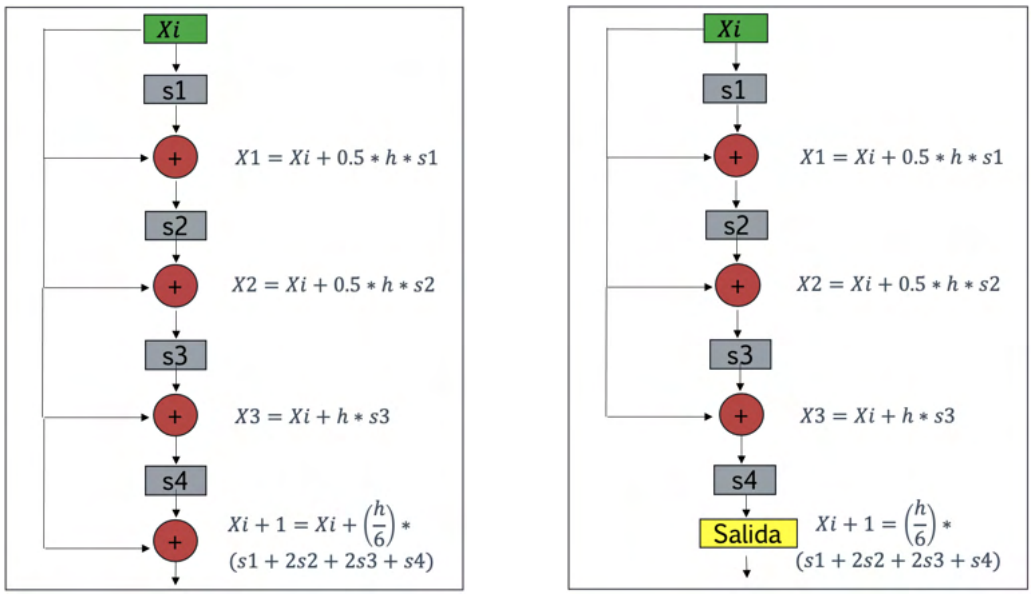
(a) Bottleneck 1 **(b)** Bottleneck 2

Figura 3.9: Bottleneck 1 y Bottleneck 2 de la arquitectura del modelo Yolov5



(a) Bottleneck_Heun 1 **(b)** Bottleneck_Heun 2

Figura 3.10: Cambios realizados en el modelo Yolov5_Heun



(a) Bottleneck_RK4 1 **(b)** Bottleneck_RK4 2

Figura 3.11: Cambios realizados en el modelo Yolov5_RK4

3.4. Entrenamiento del modelo propuesto

Se implementó la red neuronal convolucional utilizando el lenguaje de programación Python junto a la librería PyTorch, tomando como base el código de YOLOv5 [61]. Este desarrollo se llevó a cabo en una máquina virtual de Google Colaboratory, que ofrece acceso gratuito a recursos computacionales, incluyendo GPU de modelos K80, T4, P4 y P100 de NVIDIA, durante periodos definidos.

Las imágenes utilizadas en el entrenamiento fueron redimensionadas por la CNN a una resolución de 640x640 píxeles. La base de datos de entrenamiento, mencionada previamente, consta de 236 imágenes etiquetadas. En el proceso de entrenamiento, se aplicó la técnica de *transfer learning*, que implica cargar los pesos originales de la red preentrenada. Se empleó un *learning rate* de 0.01, y para la capa final se utilizó una función de activación lineal, mientras que todas las demás capas se beneficiaron de la función *Sigmoid Linear Unit* (SiLU). Como función de error a optimizar, se eligió el error cuadrático medio, ya que es fácil de optimizar y pondera de manera equitativa los errores en cajas delimitadoras tanto grandes como pequeñas.

3.5. Evaluación del modelo propuesto

Una de las métricas utilizadas para evaluar modelos de detección es la llamada Intersección sobre la Unión (IoU: *Intersection over Union*). Esta métrica es utilizada para saber que tanto interseccionan dos cajas delimitadoras, y basado en el resultado

ejecutar acciones, tales como eliminar alguna caja delimitadora o considerarlos como falsas detecciones. Luego de calcular esta métrica se utiliza el algoritmo *No Maximum Suppression* (NMS) con el objetivo de eliminar redundancia de recuadros que hayan detectado el mismo objeto.

Otras de las métricas que maneja el modelo es *mean Average Precision* (mAP), la cual es empleada para medir el rendimiento de la red neuronal, es decir, que tan bien el modelo realiza correctamente las detecciones así como las clasificaciones de los objetos sobre las imágenes de prueba o validación. El mAP es definida como la media de los valores AP (*Average Precision*) de cada clase del modelo. Este resultado AP por clase es dado por el área de bajo de la curva precisión/exhaustividad (*precision/recall* (PR), en inglés).

En todas las épocas de entrenamiento se verificará el resultado del mAP con un valor de IoU de 0.5 con el objetivo de controlar el aumento de esta métrica, y en caso de que no suceda un aumento considerable entre una época y la siguiente realizar un ajuste de los hiper-parámetros del modelo para obtener mejor resultado.

Al terminar el entrenamiento de esta red neuronal se realizará una comparación con respecto a otros modelos de redes neuronales convolucional, utilizando las mismas métricas de desempeño.

Capítulo 4

Resultados

En esta sección se realizará un análisis detallado de los resultados genereales obtenidos y por cada clase de daño, basándose en las métricas de desempeño: mAP, precisión (P), recall (R) y F1-Score, que permitirán evaluar la precisión y confiabilidad del modelo en la identificación correcta de las células dañadas. Finalmente, se presentarán los resultados de detección y clasificación de células en imágenes procesadas por el ensayo Cometa, realizando una comparación entre los modelos de detección: Scaled-YOLOv4 [72], YOLOv5_s [61], YOLOv5_Heun 3.3 y YOLOv5_Rk4 3.3. Para realizar esta comparación, se utilizan tablas e imágenes que muestran las métricas de desempeño y los resultados visuales obtenidos para cada modelo en las diferentes clases de cometas.

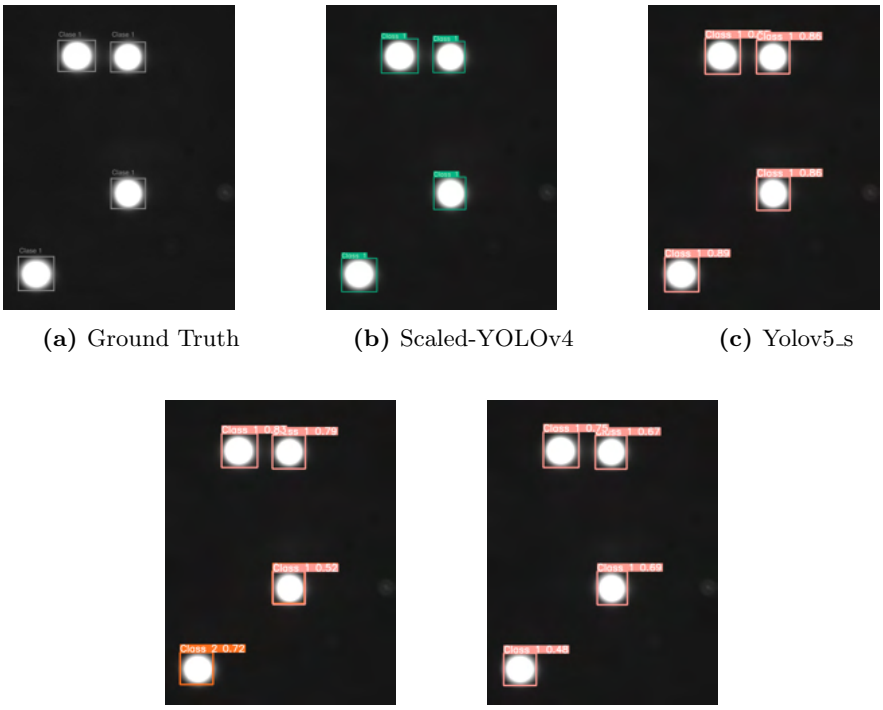
.

4.1. Detecciones Clase 1

En esta sección se presentan los resultados obtenidos para la clase 1 mediante el análisis de un conjunto de validación compuesto por 11 instancias de células clasificadas como pertenecientes a dicha clase. En la tabla 4.1 se observa una comparación de los resultados obtenidos por los diferentes modelos para la clase 1 y en la figura 4.1 se muestran las imágenes resultantes luego de ejecutar los modelos con la misma imagen de entrada y la imagen resultante deseada (*Ground Truth*).

Modelos	**P**	**R**	**F1_score**	**mAP50**	**mAP50-95**
Yolov5_s	0.730	0.727	0.728	0.788	0.375
Yolov5_Heun (h =0.25)	1.000	0.815	**0.898**	0.875	0.470
Yolov5_RK4 (h =0.25)	**0.936**	**0.818**	0.873	**0.889**	0.470
Scaled-YOLOv4	0.673	**0.818**	0.738	0.878	**0.519**

Tabla 4.1: Comparación de diferentes modelos para la clase 1



(a) Ground Truth **(b)** Scaled-YOLOv4 **(c)** Yolov5_s

(d) Yolov5_Heun **(e)** Yolov5_RK4

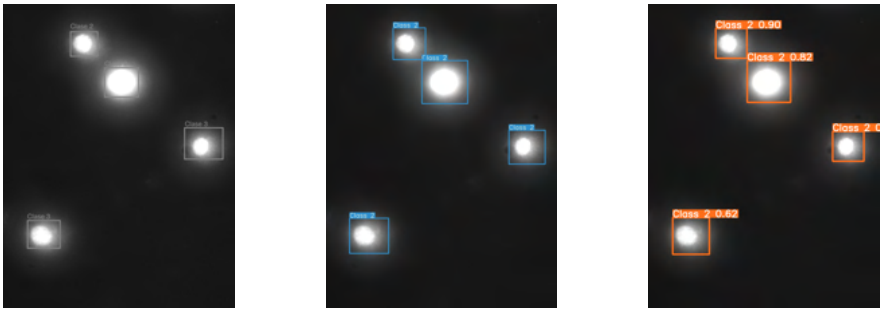
Figura 4.1: Comparación de imágenes para la clase 1

4.2. Detecciones Clase 2

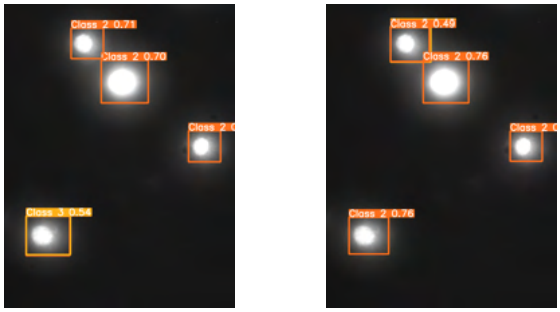
En esta sección se presentan los resultados obtenidos para la clase 2 mediante el análisis de un conjunto de validación compuesto por 20 instancias de células clasificadas como pertenecientes a dicha clase. En la tabla 4.2 se precisa una comparación de los resultados obtenidos por los diferentes modelos para la clase 2 y en la figura 4.2 se muestran las imágenes resultantes luego de ejecutar los modelos con la misma imagen de entrada y la imagen resultante deseada (*Ground Truth*).

Modelos	**P**	**R**	**F1_Score**	**mAP50**	**mAP50-95**
Yolov5_s	0.453	0.900	0.603	0.728	0.422
Yolov5_Heun (h =0.25)	**0.527**	0.850	**0.651**	0.767	0.425
Yolov5_RK4 (h =0.25)	0.465	0.950	0.624	**0.798**	**0.484**
Scaled-YOLOv4	0.315	**1.000**	0.479	0.796	0.475

Tabla 4.2: Comparación de diferentes modelos para la clase 2



(a) Ground Truth **(b)** Scaled-YOLOv4 **(c)** Yolov5_s



(d) Yolov5_Heun **(e)** Yolov5_RK4

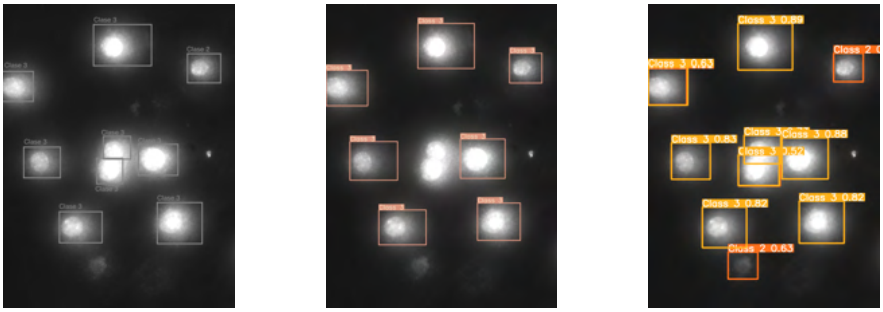
Figura 4.2: Comparación de imágenes para la clase 2

4.3. Detecciones Clase 3

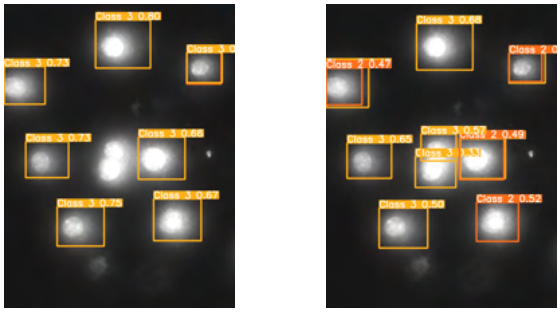
En esta sección se presentan los resultados obtenidos para la clase 3 mediante el análisis de un conjunto de validación compuesto por 36 instancias de células clasificadas como pertenecientes a dicha clase. En la tabla 4.3 se observa una comparación de los resultados obtenidos por los diferentes modelos para la clase 3 y en la figura 4.3 se muestran las imágenes resultantes luego de ejecutar los modelos con la misma imagen de entrada y la imagen resultante deseada (*Ground Truth*).

Modelos	**P**	**R**	**F1_Score**	**mAP50**	**mAP50-95**
Yolov5_s	**0.774**	0.665	**0.715**	0.787	0.450
Yolov5_Heun (h =0.25)	0.580	0.750	0.654	**0.799**	**0.474**
Yolov5_RK4 (h =0.25)	0.548	**0.861**	0.670	0.764	0.433
Scaled-YOLOv4	0.425	**0.861**	0.569	0.761	0.389

Tabla 4.3: Comparación de diferentes modelos para la clase 3



(a) Ground Truth **(b)** Scaled-YOLOv4 **(c)** Yolov5_s



(d) Yolov5_Heun **(e)** Yolov5_RK4

Figura 4.3: Comparación de imágenes para la clase 3

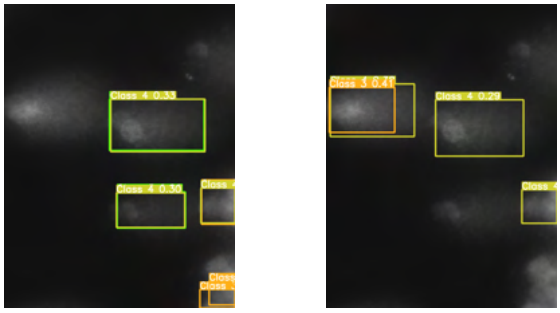
4.4. Detecciones Clase 4

En esta sección se presentan los resultados obtenidos para la clase 4 mediante el análisis de un conjunto de validación compuesto por 11 instancias de células clasificadas como pertenecientes a dicha clase. En la tabla 4.4 se establece una comparación de los resultados obtenidos por los diferentes modelos para la clase 4 y en la figura 4.4 se muestran las imágenes resultantes luego de ejecutar los modelos con la misma imagen de entrada y la imagen resultante deseada (*Ground Truth*).

Modelos	**P**	**R**	**F1_Score**	**mAP50**	**mAP50-95**
Yolov5_s	**0.729**	0.491	0.587	**0.633**	**0.375**
Yolov5_Heun (h =0.25)	0.666	0.362	0.469	0.581	0.348
Yolov5_RK4 (h =0.25)	0.332	0.545	0.413	0.421	0.247
Scaled-YOLOv4	0.450	**0.909**	**0.602**	0.594	0.284

Tabla 4.4: Comparación de diferentes modelos para la clase 4

(a) Ground Truth **(b)** Scaled-YOLOv4 **(c)** Yolov5_s



(d) Yolov5_Heun **(e)** Yolov5_RK4

Figura 4.4: Comparación de imágenes para la clase 4

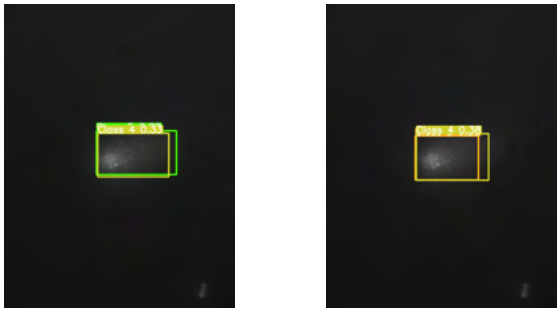
4.5. Detecciones Clase 5

En esta sección se presentan los resultados obtenidos para la clase 5 mediante el análisis de un conjunto de validación compuesto por 2 instancias de células clasificadas como pertenecientes a dicha clase. En la tabla 4.5 se muestra una comparación de los resultados obtenidos por los diferentes modelos para la clase 5 y en la figura 4.5 se observan las imágenes resultantes luego de ejecutar los modelos con la misma imagen de entrada y la imagen resultante deseada (*Ground Truth*).

Modelos	**P**	**R**	**F1_Score**	**mAP50**	**mAP50-95**
Yolov5_s	**0.420**	**1.000**	**0.592**	0.745	0.422
Yolov5_Heun (h =0.25)	0.329	**1.000**	0.495	**0.828**	**0.447**
Yolov5_RK4 (h =0.25)	0.269	0.500	0.350	0.606	0.342
Scaled-YOLOv4	0.119	**1.000**	0.213	0.235	0.128

Tabla 4.5: Comparación de diferentes modelos para la clase 5

(a) Ground Truth **(b)** Scaled-YOLOv4 **(c)** Yolov5_s



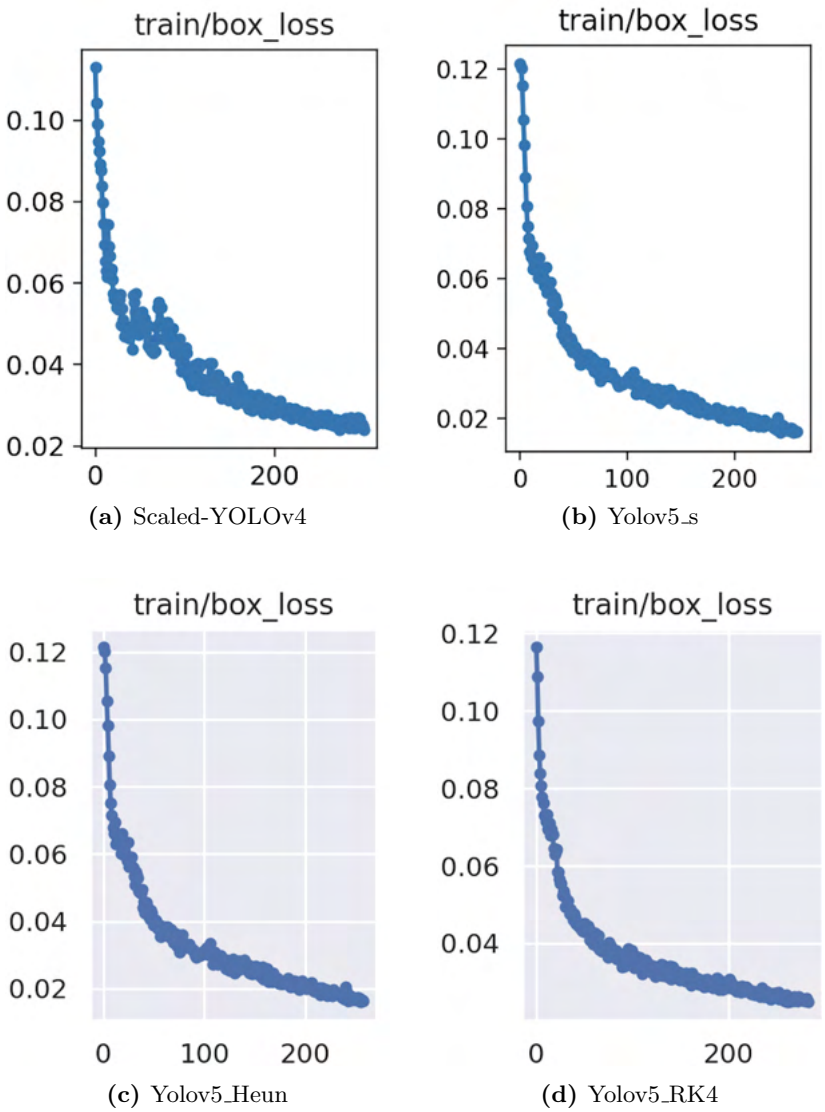
(d) Yolov5_Heun **(e)** Yolov5_RK4

Figura 4.5: Comparación de imágenes para la clase 5

4.6. Análisis y comparación de los resultados generales

Luego de entrenar los 4 modelos por 300 épocas se obtuvieron las gráficas que muestran los resultados de las métricas: pérdida de caja *(box_loss)*, pérdida de objeto *(obj_loss)* y pérdida de clasificación *(cls_loss)*, las cuales se representan en la figura 4.6, figura 4.7 y figura 4.8 respectivamente.

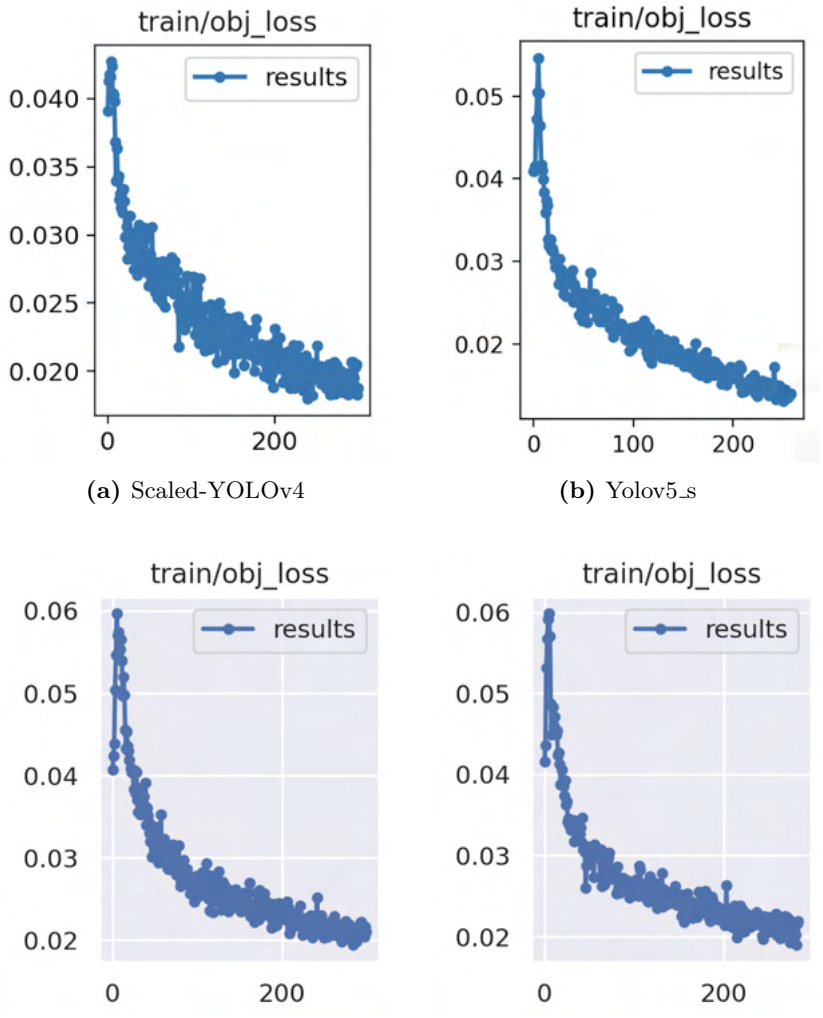
Como se puede constatar todos los modelos presentan una disminución gradual en las pérdidas a medida que avanza el entrenamiento, aunque los que tienen como base a Yolov5 convergen rápidamente de manera similar.



(a) Scaled-YOLOv4 **(b)** Yolov5_s

(c) Yolov5_Heun **(d)** Yolov5_RK4

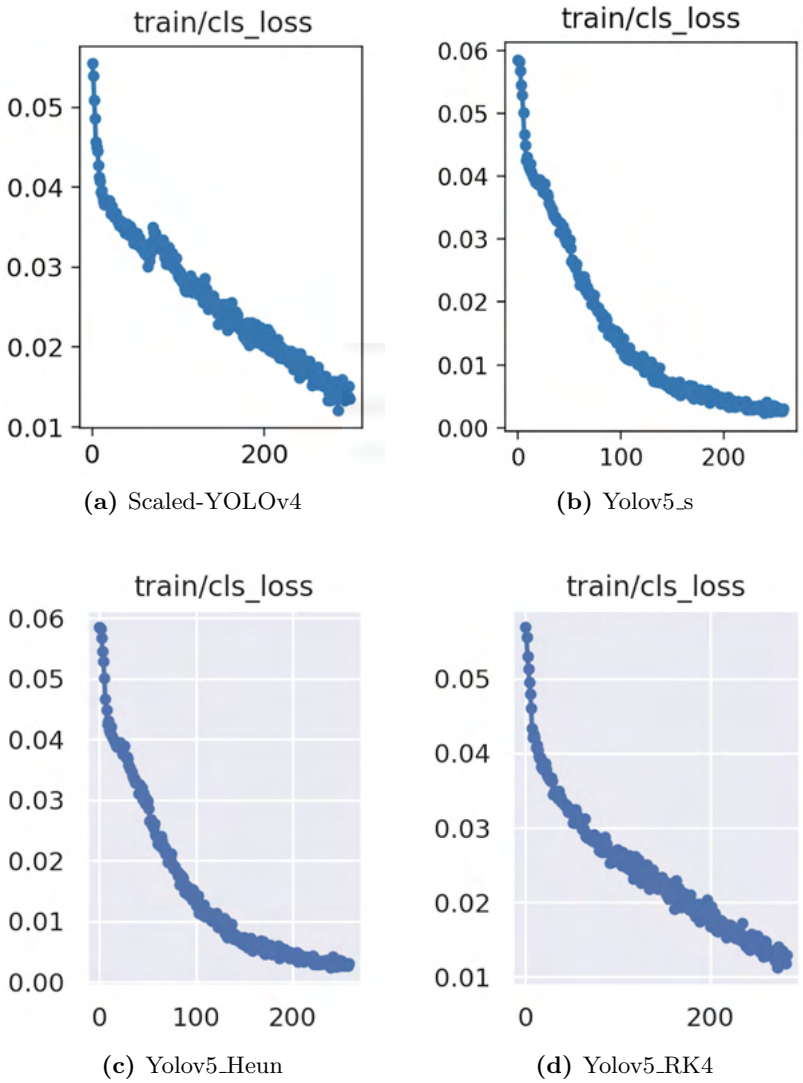
Figura 4.6: Comparación de la pérdida promedio de las predicciones de cajas durante el entrenamiento



(a) Scaled-YOLOv4 **(b)** Yolov5_s

(c) Yolov5_Heun **(d)** Yolov5_RK4

Figura 4.7: Comparación de la pérdida promedio de las detecciones de objetos durante el entrenamiento



(a) Scaled-YOLOv4 **(b)** Yolov5_s

(c) Yolov5_Heun **(d)** Yolov5_RK4

Figura 4.8: Comparación de la pérdida promedio de la clasificación de objetos durante el entrenamiento

En la Tabla 4.6 se presentan los resultados obtenidos después de entrenar los modelos propuestos, utilizando un conjunto de validación compuesto por 80 instancias de células clasificadas en alguna de las cinco clases de cometas. La cual incluye diversas métricas de evaluación que nos permiten comparar y determinar cuál de los modelos presenta un mejor rendimiento en la tarea de detección y clasificación de células en imágenes procesadas por el ensayo Cometa.

Modelos	**P**	**R**	**F1 Score**	**mAP50**	**mAP50-95**
Yolov5_s	**0.621**	0.757	**0.682**	0.736	0.409
Yolov5_Heun (h =0.25)	0.620	0.756	0.681	**0.770**	**0.433**
Yolov5_RK4 (h =0.25)	0.510	0.735	0.602	0.695	0.396
Scaled-YOLOv4	0.375	**0.918**	0.532	0.653	0.359

Tabla 4.6: Comparación de diferentes modelos para todas las clases

Al analizar la tabla de resultados generales, revelan que el modelo Yolov5_Heun en términos de la métrica *F1 Score* alcanza un valor de 0.681, el cual es muy cercano al valor más alto obtenido por el modelo Yolov5_s (0.682). Esto indica que ambos modelos tienen un equilibrio similar entre precisión y recall en la detección y clasificación de células. Sin embargo, al examinar las métricas de mAP50 y mAP50-95, el modelo Yolov5_Heun supera a los otros modelos, lo cual sugiere que Yolov5_Heun logra una alta precisión en la detección de objetos a través de diferentes umbrales de confianza.

Capítulo 5

Conclusiones

En el presente trabajo se propuso un sistema computacional basado en una red neuronal convolucional para la detección y clasificación automática de las células procesadas por el ensayo Cometa. Se realizó una revisión exhaustiva de la literatura relacionada con la detección y clasificación de células dañadas, así como el uso de redes neuronales convolucionales en este contexto.

Se seleccionó como punto de partida para esta investigación la red neuronal convolucional YOLOv5 y se propuso un cambio principal en el sistema, que consiste en la incorporación de una fórmula diferente en el *forward* del módulo *bottleneck*. Se exploraron dos enfoques alternativos, el Método de Heun y el Método de Runge-Kutta de cuarto orden (RK4).

Los resultados obtenidos muestran que el modelo propuesto Yolov5_Heun logró un mejor desempeño en la detección y clasificación de células procesadas por el ensayo cometa, evidenciado por sus valores de F1-Soce (0.681), mAP50 (0.770) y mAP50-95 (0.433).

5.1. Trabajo futuro

A pesar que el método propuesto en esta investigación ha logrado resultados positivos en la detección y clasificación de cometas, aún existe un amplio campo para el desarrollo y refinamiento de técnicas y enfoques que posibiliten una detección y clasificación más precisa de estos cometas.

Es importante considerar la exploración de otras arquitecturas de redes neuronales para ser evaluadas y determinar si ofrecen un rendimiento superior en términos de precisión y velocidad en la detección de cometas en imágenes de microscopio.

Otro trabajo a futuro sería la expansión de la base de datos utilizada en este estudio, incorporando imágenes de cometas con variadas formas y diversos grados de daño, en particular los de categoría 5 que para este primer trabajo quedó sub-representada. Esto permitirá desarrollar un modelo más versátil capaz de reconocer cometas de diferentes características.

Bibliografía

[1] James D Watson and Francis HC Crick. Molecular structure of nucleic acids: a structure for deoxyribose nucleic acid. *Nature*, 171(4356):737–738, 1953. 11

[2] André P. Schuch and Carlos Frederico Martins Menck. The genotoxic effects of dna lesions induced by artificial uv-radiation and sunlight. *Journal of photochemistry and photobiology. B, Biology*, 99 3:111–6, 2010. URL https://api.semanticscholar.org/CorpusID:9379639. 11

[3] Rinat Islamov, A. Aitynova, and Aizhan Zhussupova. Genetic toxicology and genetically active environmental factors. *International Journal of Biology*, 11:36–46, 2018. URL https://api.semanticscholar.org/CorpusID:91675811. 11

[4] Felipe de Araújo Nascimento, Daniela de Melo Silva, Hugo Nunes, and Michelle Rocha-Parise. Evaluation of dna damage and toxicological methodology development: A bibliometric study. *Human & Experimental Toxicology*, 39, 02 2020. doi: 10.1177/0960327120903481. 11

[5] Caroline Lanier, Nicolas Manier, Damien Cuny, and Annabelle Deram. The comet assay in higher terrestrial plant model: Review and evolutionary trends. *Environmental pollution (Barking, Essex : 1987)*, 207:6–20, 08 2015. doi: 10.1016/j.envpol.2015.08.020. 11

[6] Andem Andem. Review on comet assay: A reliable tool for assessing dna damage in animal models. *JOURNAL OF CURRENT RESEARCH IN SCIENCE*, 1:405–427, 06 2013. 11

[7] Daryl W. Fairbairn, Peggy L. Olive, and Kim L. O'Neill. The comet assay: a comprehensive review. *Mutation Research/Reviews in Genetic Toxicology*, 339(1):37–59, 1995. ISSN 0165-1110. doi: https://doi.org/10.1016/0165-1110(94)00013-3. URL https://www.sciencedirect.com/science/article/pii/0165111094000133. 11

[8] O. Ostling and K. J. Johanson. Microelectrophoretic study of radiation-induced dna damages in individual mammalian cells. 123:291–298, 1984. ISSN 0006-291X. doi: 10.1016/0006-291x(84)90411-x. 11, 16

[9] Eugenia Cordelli, Margherita Bignami, and Francesca Pacchierotti. Comet assay: a versatile but complex tool in genotoxicity testing. *Toxicology research*, 10 1:68–78, 2021. URL https://api.semanticscholar.org/CorpusID:231988256. 11

[10] Ramazan Uzen, Ayşe Gaye Tomatır, Nurhan Cucer, and Hamiyet Donmez-Altuntas. Comet assay and applications. *International Journal of Scientific and Technological Research*, 2019. URL https://api.semanticscholar.org/CorpusID:198270495. 11

[11] R R Tice, E Agurell, D Anderson, B Burlinson, A Hartmann, H Kobayashi, Y Miyamae, E Rojas, J C Ryu, and Y F Sasaki. Single cell gel/comet assay: guidelines for in vitro and in vivo genetic toxicology testing. *Environ Mol Mutagen*, 35(3):206–221, 2000. ISSN 0893-6692 (Print); 0893-6692 (Linking). doi: 10.1002/(sici)1098-2280(2000)35:3⟨206::aid-em8⟩3.0.co;2-j. 12, 16

[12] Francis Collins, George Gray, and Judith Bucher. Toxicology: Transforming environmental health protection. *Science (New York, N.Y.)*, 319:906–7, 03 2008. doi: 10.1126/science.1154619. 12

[13] Aleksandra Fucic. [the comet assay method: a new approach in genotoxicology research]. *Arhiv za higijenu rada i toksikologiju*, 48 4:413–9, 1997. URL https://api.semanticscholar.org/CorpusID:43360351. 12

[14] CometScore. http://rexhoover.com/, 2017. Consultado el 30 de noviembre de 2022. 12

[15] Benjamin M. Gyori, Gireedhar Venkatachalam, P.S. Thiagarajan, David Hsu, and Marie-Veronique Clement. Opencomet: An automated tool for comet assay image analysis. *Redox Biology*, 2:457–465, 2014. ISSN 2213-2317. doi: https://doi.org/10.1016/j.redox.2013.12.020. URL https://www.sciencedirect.com/science/article/pii/S2213231714000032. 12, 13, 14

[16] Caroline Schneider, Wayne Rasband, and Kevin Eliceiri. Nih image to imagej: 25 years of image analysis. *Nature Methods*, 9, 07 2012. doi: 10.1038/nmeth.2089. 12

[17] M. Perini Buenahora. Comparación del nivel de daño genético inducido por bleomicina, en presencia o ausencia de un pool de inhibidores de enzimas de reparación en células vero, 2021. 13

[18] Sreelatha Ganapathy, Aparna Muraleedharan, Puthumangalathu Savithri Sathidevi, Parkash Chand, and Ravi Philip Rajkumar. Cometq: An automated tool for the detection and quantification of dna damage using comet assay image analysis. *Computer Methods and Programs in Biomedicine*, 133:143–154, 2016. ISSN 0169-2607. doi: https://doi.org/10.1016/j.cmpb.2016.05.020. URL https://www.sciencedirect.com/science/article/pii/S0169260716301006. 13

[19] Taehoon Lee, Sungmin Lee, Woo Young Sim, Yu Jung, Sunmi Han, Joong-Ho Won, Hyeyoung Min, and Sungroh Yoon. HiComet: A high-throughput comet analysis tool for large-scale DNA damage assessment. *BMC Bioinformatics*, 19, 02 2018. doi: 10.1186/s12859-018-2015-7. 13, 14

[20] Attila Beleon, Sara Pignatta, Chiara Arienti, Antonella Carbonaro, Peter Horvath, Giovanni Martinelli, Gastone Castellani, Anna Tesei, and Filippo Piccinini. Cometanalyser: A user-friendly, open-source deep-learning microscopy tool for quantitative comet assay analysis. *Computational and Structural Biotechnology Journal*, 20:4122–4130, 2022. ISSN 2001-0370. doi: https://doi.org/10.1016/j.csbj.2022.07.053. URL https://www.sciencedirect.com/science/article/pii/S2001037022003336. 13

[21] Yiyu Hong, Hyo-Jeong Han, Hannah Lee, Donghwan Lee, Junsu Ko, Zhen-Yu Hong, Ji-Young Lee, Ju-Hyung Seok, Hee Seon Lim, Woo-Chan Son, and Insuk Sohn. Deep learning method for comet segmentation and comet assay image analysis. *Scientific Reports*, 10(1):18915, November 2020. doi: 10.1038/s41598-020-75592-7. URL https://pubmed.ncbi.nlm.nih.gov/33144610/. 13

[22] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross B. Girshick. Mask r-cnn. *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 2980–2988, 2017. 13

[23] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zach DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. *PyTorch: An Imperative Style, High-Performance Deep Learning Library*. Curran Associates Inc., Red Hook, NY, USA, 2019. 13

[24] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org. 14

[25] Srikanth Namuduri, Prateek Mehta, Lise Barbe, Stephanie Lam, Zohreh Faghihmonzavi, Steve Finkbeiner, and Shekhar Bhansali. Faster deep ensemble averaging for quantification of DNA damage from comet assay images with uncertainty estimates, 2021. 14

[26] T. Sauer. *Numerical Analysis*. Pearson, 2018. ISBN 9780134696454. URL https://books.google.com.mx/books?id=2rKgtAEACAAJ. 19, 20, 21, 22

[27] Raquel Flórez López and José Miguel Fernández Fernández. *Las redes neuronales artificiales*. Netbiblo, 2008. 22

[28] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521 (7553):436–444, 2015. 22

[29] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255, 2009. doi: 10.1109/CVPR.2009.5206848. 22

[30] Syed Muhammad Anwar, Muhammad Majid, Adnan Qayyum, Muhammad Awais, Majdi Alnowami, and Muhammad Khurram Khan. Medical image analysis using convolutional neural networks: a review. *Journal of medical systems*, 42:1–13, 2018. 22

[31] Leila Abdelrahman, Manal Al Ghamdi, Fernando Collado-Mesa, and Mohamed Abdel-Mottaleb. Convolutional neural networks for breast cancer detection in mammography: A survey. *Computers in biology and medicine*, 131:104248, 2021. 22

[32] Andre Esteva, Brett Kuprel, Roberto A Novoa, Justin Ko, Susan M Swetter, Helen M Blau, and Sebastian Thrun. Corrigendum: Dermatologist-level classification of skin cancer with deep neural networks. *Nature*, 546(7660):686, June 2017. ISSN 0028-0836. doi: 10.1038/nature22985. URL https://doi.org/10.1038/nature22985. 22

[33] Yuchi Tian, Kexin Pei, Suman Jana, and Baishakhi Ray. Deeptest: Automated testing of deep-neural-network-driven autonomous cars. In *Proceedings of the 40th international conference on software engineering*, pages 303–314, 2018. 22

[34] Mate Szarvas, Akira Yoshizawa, Munetaka Yamamoto, and Jun Ogata. Pedestrian detection with convolutional neural networks. In *IEEE Proceedings. Intelligent Vehicles Symposium, 2005.*, pages 224–229. IEEE, 2005. 22

[35] Yihui Wu, Yulong Liu, Jianmin Li, Huaping Liu, and Xiaolin Hu. Traffic sign detection based on convolutional neural networks. In *The 2013 international joint conference on neural networks (IJCNN)*, pages 1–7. IEEE, 2013. 22

[36] Pejman Rasti, Tonis Uiboupin, Sergio Escalera, and Gholamreza Anbarjafari. Convolutional neural network super resolution for face recognition in surveillance monitoring. In *Articulated Motion and Deformable Objects: 9th International Conference, AMDO 2016, Palma de Mallorca, Spain, July 13-15, 2016, Proceedings 9*, pages 175–184. Springer, 2016. 22

[37] Najwa Altwaijry and Isra Al-Turaiki. Arabic handwriting recognition system using convolutional neural network. *Neural Computing and Applications*, 33(7):2249–2261, 2021. 22

[38] Emmanuel Maggiori, Yuliya Tarabalka, Guillaume Charpiat, and Pierre Alliez. Convolutional neural networks for large-scale remote-sensing image classification. *IEEE Transactions on geoscience and remote sensing*, 55(2):645–657, 2016. 22

[39] Hiroyuki Takeda, Sina Farsiu, and Peyman Milanfar. Kernel regression for image processing and reconstruction. *IEEE Transactions on image processing*, 16(2): 349–366, 2007. 22

[40] Matt P Wand and M Chris Jones. *Kernel smoothing*. CRC press, 1994. 23

[41] Sucharita Ghosh. *Kernel smoothing: Principles, methods and applications*. John Wiley & Sons, 2018. 23

[42] Pietro Perona. Steerable-scalable kernels for edge detection and junction analysis. In *Computer Vision—ECCV'92: Second European Conference on Computer Vision Santa Margherita Ligure, Italy, May 19–22, 1992 Proceedings 2*, pages 3–18. Springer, 1992. 23

[43] O Rebecca Vincent, Olusegun Folorunso, et al. A descriptive algorithm for sobel image edge detection. In *Proceedings of informing science & IT education conference (InSITE)*, volume 40, pages 97–107, 2009. 23

[44] Tanvir Ahmed Abbasi and Mohammad Usaid Abbasi. A novel architecture for prewitt edge detector. *Journal of Active & Passive Electronic Devices*, 4(3), 2009. 23

[45] Yin Cui, Feng Zhou, Jiang Wang, Xiao Liu, Yuanqing Lin, and Serge Belongie. Kernel pooling for convolutional neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2921–2930, 2017. 23

[46] Fernando Giménez Palomares, Juan Serrá, and Elena Alemany. Aplicación de la convolución de matrices al filtrado de imágenes. *Modelling in Science Education and Learning*, 9:97, 01 2016. doi: 10.4995/msel.2016.4524. 23

[47] Andrinandrasana David Rasamoelina, Fouzia Adjailia, and Peter Sincak. A review of activation function for artificial neural network. *2020 IEEE 18th World Symposium on Applied Machine Intelligence and Informatics (SAMI)*, pages 281–286, 2020. 26

[48] Guifang Lin and Wei Shen. Research on convolutional neural network based on improved relu piecewise activation function. *Procedia Computer Science*, 131:977–984, 2018. 26

[49] Prajit Ramachandran, Barret Zoph, and Quoc V. Le. Searching for activation functions. *ArXiv*, abs/1710.05941, 2018. 27

[50] Ioannis Kouretas and Vassilis Paliouras. Simplified hardware implementation of the softmax activation function. *2019 8th International Conference on Modern Circuits and Systems Technologies (MOCAST)*, pages 1–4, 2019. 28

[51] Shie Mannor, Dori Peleg, and Reuven Y. Rubinstein. The cross entropy method for classification. *Proceedings of the 22nd international conference on Machine learning*, 2005. 28

[52] Suriya Gunasekar, Jason D. Lee, Daniel Soudry, and Nathan Srebro. Implicit bias of gradient descent on linear convolutional networks. In *Neural Information Processing Systems*, 2018. 28

[53] Silvére Bonnabel. Stochastic gradient descent on riemannian manifolds. *IEEE Transactions on Automatic Control*, 58:2217–2229, 2011. 29

[54] Alhussein Fawzi, Horst Samulowitz, Deepak S. Turaga, and Pascal Frossard. Adaptive data augmentation for image classification. *2016 IEEE International Conference on Image Processing (ICIP)*, pages 3688–3692, 2016. 30

[55] Francisco López de la Rosa, José L. Gómez-Sirvent, Roberto Sánchez-Reolid, Rafael Morales, and Antonio Fernández-Caballero. Geometric transformation-based data augmentation on defect classification of segmented images of semiconductor materials using a resnet50 convolutional neural network. *Expert Systems with Applications*, 206:117731, 2022. ISSN 0957-4174. doi: https://doi.org/10.1016/j.eswa.2022.117731. URL https://www.sciencedirect.com/science/article/pii/S0957417422010120. 30

[56] David Tellez, Geert Litjens, Péter Bándi, Wouter Bulten, John-Melle Bokhorst, Francesco Ciompi, and Jeroen Van Der Laak. Quantifying the effects of data augmentation and stain color normalization in convolutional neural networks for computational pathology. *Medical image analysis*, 58:101544, 2019. 30

[57] Junlin Han, Pengfei Fang, Weihao Li, Jie Hong, Mohammad Ali Armin, Ian Reid, Lars Petersson, and Hongdong Li. You only cut once: Boosting data augmentation with a single cut. In *International Conference on Machine Learning*, pages 8196–8212. PMLR, 2022. 30

[58] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection, 2015. URL https://arxiv.org/abs/1506.02640. 30

[59] Pierre Sermanet, Koray Kavukcuoglu, Soumith Chintala, and Yann LeCun. Pedestrian detection with unsupervised multi-stage feature learning, 2013. 30

[60] Joseph Redmon and Ali Farhadi. Yolov3: An incremental improvement. *CoRR*, abs/1804.02767, 2018. URL http://arxiv.org/abs/1804.02767. 31

[61] Glenn Jocher, Ayush Chaurasia, Alex Stoken, Jirka Borovec, NanoCode012, Yonghye Kwon, Kalen Michael, TaoXie, Jiacong Fang, imyhxy, Lorna, (Zeng Yifu), Colin Wong, Abhiram V, Diego Montes, Zhiqiang Wang, Cristi Fati, Jebastin Nadar, Laughing, UnglvKitDe, Victor Sonck, tkianai, yxNONG, Piotr Skalski, Adam Hogan, Dhruv Nair, Max Strobel, and Mrinal Jain. ultralytics/yolov5: v7.0 - YOLOv5 SOTA Realtime Instance Segmentation. Zenodo, November 2022. doi: 10.5281/zenodo.7347926. URL https://doi.org/10.5281/zenodo.7347926. 32, 34, 35, 49, 51

[62] Apurva Badithela, Tichakorn Wongpiromsarn, and Richard M. Murray. Evaluation metrics for object detection for autonomous systems, 2022.

[63] Rasmus Rothe, Matthieu Guillaumin, and Luc Van Gool. Non-maximum suppression for object detection by passing messages between windows. In *Asian Conference on Computer Vision*, 2014.

[64] Jan Hendrik Hosang, Rodrigo Benenson, and Bernt Schiele. A convnet for non-maximum suppression. *ArXiv*, abs/1511.06437, 2015.

[65] José Manuel Sánchez Muñoz. Análisis de calidad cartográfica mediante el estudio de la matriz de confusión. *Pensamiento Matemático*, 6:9–26, 2016.

[66] Ricardo Borja-Robalino, Antonio Monleon-Getino, and Jose Benedé. Estandarización de métricas de rendimiento para clasificadores machine y deep learning. 02 2020.

[67] Rafael Padilla, Sergio L. Netto, and Eduardo A. B. da Silva. A survey on performance metrics for object-detection algorithms. *2020 International Conference on Systems, Signals and Image Processing (IWSSIP)*, pages 237–242, 2020.

[68] Quentin Hennocq, Thomas Bongibault, Matthieu Bizière, Ombline Delassus, Maxime Douillet, Valérie Cormier-Daire, Jeanne Amiel, Stanislas Lyonnet, Sandrine Marlin, Marlène Rio, Arnaud Picard, Roman Hossein Khonsari, and Nicolas Garcelon. An automatic facial landmarking for children with rare diseases. *American Journal of Medical Genetics Part A*, 191:1210 – 1221, 2023.

[69] Q. Rahman, Niko Sunderhauf, and Feras Dayoub. Per-frame map prediction for continuous performance monitoring of object detection during deployment. *2021 IEEE Winter Conference on Applications of Computer Vision Workshops (WACVW)*, pages 152–160, 2020.

[70] Vladimir Y. Mariano, Junghye Min, Jin Hyeong Park, Rangachar Kasturi, David Mihalcik, Huiping Li, David S. Doermann, and Thomas Drayer. Performance evaluation of object detection algorithms. *Object recognition supported by user interaction for service robots*, 3:965–969 vol.3, 2002.

[71] Tzutalin. Labelimg, Git code, 2015. URL https://github.com/tzutalin/labelImg.

[72] Chien-Yao Wang, Alexey Bochkovskiy, and Hong-Yuan Mark Liao. Scaled-yolov4: Scaling cross stage partial network. *CoRR*, abs/2011.08036, 2020. URL https://arxiv.org/abs/2011.08036.